CONTROLLED LINK ESTABLISHMENT ATTACKS ON DISTRIBUTED SENSOR NETWORKS AND COUNTERMEASURES

DISSERTATION

Submitted to the University Graduate School
University of Technology, Sydney

In fulfilment of the requirements for the degree of
Doctor of Philosophy in Engineering and Information Technology

BY

THANH DAI TRAN

B.Eng., Hanoi University of Science and Technology, 2005
M.Eng., Kyung Hee University, 2007

Sydney, December 2010

CERTIFICATE OF AUTHORSHIP/ORIGINALITY

I certify that the work in this thesis has not previously been submitted for a degree nor has it been submitted as part of requirements for a degree except as fully acknowledged within the text.

I also certify that the thesis has been written by me. Any help that I have received in my research work and the preparation of the thesis itself has been acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

Signature of Candidate

……

To the memory of my dearest grandmother and grandfather

Abstract

For over a decade, the boom in research, development, and application of distributed sensor networks (DSNs) has enabled their pervasion in many aspects of human life. In such networks, collaboration among sensor nodes plays a key role in resolving distributed tasks. Typically, traditional cryptographic protections such as encryption and authentication are utilised to secure this collaboration against malicious attacks.

Unfortunately, this secured collaboration is undermined by an attack named Controlled Link Establishment Attack (CLEA). To launch CLEA, the attacker first captures and compromises a limited number of nodes to extract their secret information. Next, the attacker repetitively utilises the compromised nodes and secret information to create overwhelming controlled links with legitimate nodes. These controlled links are then used to subvert network-wide cooperative efforts or gain the control of the network.

This thesis comprises two parts: CLEA investigation and development of new countermeasures against CLEA. The investigation involves (i) identifying and characterising CLEA based on the examination of actual instances (ii) undertaking a literature review of existing key establishment schemes for DSNs and pinpointing their vulnerability to CLEA (iii) performing a comprehensive survey of existing countermeasures applicable to defend against CLEA, and (iv) studying the feasibility of CLEA. The conclusion drawn from this investigation is that although CLEA is a real and serious threat, no sufficiently robust and efficient countermeasures have been found in the literature to defeat the attack.

The development starts with a study of related works that can be used as building blocks for new countermeasures followed by their description. The proposed countermeasures can be classified into either protection-based approach or detection-based approach. Following the first approach, three schemes focusing on protecting key establishment schemes by leveraging a cryptographic one-way hash chain are developed. Following the second approach, three schemes are introduced. The first two schemes are capable

of detecting and stamping out CLEA attempts from the beginning. The final scheme is even more powerful than the previous ones with the ability to identify and revoke the source of the attack.

Finally, thorough evaluations of the proposed schemes in respect of security features and performance overheads are carried out through intensive analyses, simulations, implementation, and extensive comparison with other schemes. The findings from these evaluations indicate that the proposed schemes achieve robust and effective prevention, detection, and revocation capability against CLEA with reasonable overheads. In comparison, the protection-based schemes are more performance efficient but less security effective than the detection-based schemes. They are all suitable for use in the current generation of sensor nodes.

Acknowledgements

A major research project like this is never entirely the work of the author. Over my doctoral candidature, I have been receiving support and encouragement from different people in various ways. My thesis would not have been completed without these support and encouragement.

First and foremost, I wish to express my deepest gratitude towards my supervisor, Dr. Johnson Agbinya, who has been my supervisor since the beginning of my study. Johnson has always been a great mentor and above all good friend, giving me tremendously valuable suggestions, thoughtful guidance, and continuous support. His excellent ideas and feedback from numerous face-to-face discussions and group meetings have assisted me in selecting my thesis topic, formulating research questions, developing new methodology for these questions, publishing my work, and completing this thesis. His research expertise and community network as well as his financing for my publications and conference attendance have been very helpful. His dedication to research and vastness of knowledge have set a standard I long to meet for the years to come. It is my great honour to receive his guidance and direction towards the accomplishment of this dissertation.

My sincere thanks are extended to my co-supervisor, Dr. Adel Ali Al-Jumaily and my previous co-supervisor, A/Prof. Elaine Lawrence, for their advice and support throughout my thesis.

I am indebted to the Faculty of Engineering and Information Technology (FEIT), UTS in general and Prof. Hung Nguyen in particular for having offered me a Faculty of Engineering Postgraduate Research Scholarship. Without this scholarship, my dream of studying in Australia would not have been fulfilled. I would like to extend further my thanks to the Vice Chancellor's Conference Funding and FEIT Travel Funding committees for their generous financial support of my conference trips.

My keen appreciation goes to Ms. Phyllis Agius, Research Administration Officer, for her dedicated assistance. She has always been my first point of contact whenever I had difficulties with paperwork and administrative details.

Moving toward more personal acknowledgments, I owe special gratitude to my closest friend Richard Turnell for his true companionship, unwavering support, and warm encouragement. He has made me feel at home and devoted countless hours to improving my English skills, proofreading my paper drafts, teaching me how to drive cars, and above all helping me absorb and value Australian culture. He has been my point of reference for difficulties I faced in my daily life and acted as my main source of hilarity, vitality, and exhilaration. Without his effort, I would not have concentrated on my study and finished it on time.

My heartfelt gratitude towards my beloved fiancee, Giang, cannot be expressed in words. Her eternal love, understanding, and sharing are always a powerful source of inspiration and energy for my study. She has helped me keep balance in my mind and fetched me happiness in life when I lost track of my research and felt disappointed and stressed. I do not hesitate to say that, with her by my side, I am not alone on this so-called ‘lonely journey’.

Last, but most important, is the dedication of this thesis to my dearest parents and younger sister for their never-ending love, unconditional support, and tireless sacrifices. It is no exaggeration to say that I am very lucky to have such marvellous family members. I hope they will read this work some day and be proud of me.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter we first introduce distributed sensor networks (DSNs) including their characteristics and current and participatory applications with the aim of emphasising the necessity of security measures to DSNs. Next, we narrow down the scope of this thesis by clearly defining the problem addressed in the thesis. We then present thesis objectives, design challenges, and thesis contributions and organisation. We then close the chapter by enumerating publications derived from the three-year research project.

1.1 Distributed Sensor Networks

Over the past few years, the rapid convergence of research trends from a number of different disciplines including digital circuitry, wireless communications, and the micro-electromechanical system (MEMS) to name but a few has opened the door to a new generation of large-scale distributed networks of miniature sensor nodes. These DSNs exhibit revolutionary approaches to traditional methods of environmental sensing and monitoring [1]. Furthermore, DSNs are expected to be an inspiring and enabling technology for a broad spectrum of novel applications in military and civilian areas. One of the initial-stage examples is CitiSense [2], a citywide network of thousands of cell phone-based sensor nodes, whose goal is to provide up-to-the-minute environmental information about where citizens live, work and play [3]. The most intriguing fantasies include concepts of “smart dust” - a term used to describe tiny digital sensors no bigger than grains of sand, scattered around the globe, collecting all sorts of information and communicating with powerful computer networks to monitor, measure and understand the physical world in new ways [4].

In essence a DSN, originally motivated by military applications, consists of a large number of spatially distributed autonomous sensor nodes capable of sensing and

monitoring a wide range of physical conditions such as temperature, light intensity, air pressure, humidity, sound levels, vibration, motion or speed, soil composition, chemical agents, and magnetic field strength [5] from their surroundings. Each sensor node is usually composed of one or a few sensing components, a processing component, which is able to carry out simple computation, a communication component, which is capable of wireless communicating with its neighbour nodes within short distances and a power supply unit. The size and cost of a sensor node may vary depending on the actual needs of the application. The size may vary from that of a shoe box [6] down to the size of a grain of dust as envisaged in [7]. The world's smallest sensor node ever developed is reported in [8] with the size of 2.5 by 3.5 by 1 millimeters which is smaller than Abraham Lincoln's head on a penny. Likewise, the cost can range from hundreds of dollars as for commercial off-the-shelf (COTS) nodes down to the projected price of a few cents.

The deployment of sensor nodes can be carried out in various forms according to different application scenarios. Nodes may be manually placed in the area of interest or rapidly deployed by a reconnaissance team near the paths of travel, or even randomly distributed from a vehicle or an airplane over enemy terrain. During the whole lifetime of a sensor network application, the deployment can happen once or may also be a continuous process with more nodes being added over time [6]. After the deployment, sensor nodes quickly self-organise together to form a wireless ad-hoc network.

Sensed data by sensor nodes is collected in many ways. It can be sent to one or few stationary control nodes (base stations) directly or through short-range multi-hop wireless routes consisting of sequential intermediate sensor nodes. The data can also be collected by mobile control nodes (e.g., unmanned aerial units, foot soldiers) that access sensor network at unpredictable locations and utilise the first encountered node as a conduit for the information accumulated by the network [9]. The control nodes may further process the collected data, disseminate control commands to sensor nodes, and function as a gateway to a traditional wired network.

1.1.1 Characteristics

DSNs exhibit unique characteristics in comparison with traditional sensing networks. Obviously most of them are advantageous features which play an important role in the proliferation of such networks. The following provides a summary of these features.

- Innovative tools for various applications: Due to a great number of advantages over traditional sensing and monitoring systems, DSNs have found their applications in many diverse areas such as agricultural and environmental monitoring, health care, industrial process control, security and surveillance to name but a few. They have also been envisaged as the enabling technology [10-12] to realise fictional ideas such as *the ambient intelligence* [13, 14].
- Extended coverage: The coverage and deployment flexibility of traditional sensing approaches are narrowly limited to a certain physical terrain due to the constraints of cost and manual deployment. In contrast, although the coverage of a single sensor node is small, the densely distributed nodes can work simultaneously and cooperatively to extend the coverage of the whole network. Furthermore, the network coverage can be adjusted conveniently by adding new nodes or moving nodes.
- Flexible deployment: DSNs can be deployed in harsh environments where human intervention is impossible or undesirable. They can also be deployed over unfriendly terrains where infrastructures are not available and/or traditional deployment fashion is not feasible. Moreover, DSNs are much easier to relocate than the traditional sensing networks. This makes DSNs very suitable for applications whose objectives change over the network lifetime.
- Reliability, flexibility and self-organisation: Although a single node is quite limited in terms of capability and reliability, densely deployed sensor nodes can provide high fault tolerance, and thus improve the robustness of the entire system. Since a sensor fails, its neighbouring nodes can substitute for that node to provide the same or similar information. Multiple routing paths are also available to help the system avoid the problem of communication link failures. The flexibility is exhibited through several facets such as the adjustable sensing coverage by node displacement or addition, trade-off between delay and

information accuracy by sensor node cooperation, balance of power consumption among collaborative sensor nodes.

- Improved monitoring ability and information quality: Data fusion from a great number of sensor nodes can reveal previously hidden phenomena in the physical world as well as provide more valuable insights to the end user than the high-accuracy but high-cost single-sited sensor network.
- Lower cost: DSNs are expected to be less expensive than their macro-sensor counterparts due to the reduced size of each sensor node, lower price and the ease of their deployment. In addition, the cost of administration and maintenance of DSNs is much lower than that of macro-sensor networks due to their self-configuration and self-maintenance features.
- Easy debugging and re-tasking: Owing to the over-the-air code dissemination and acquisition protocols for DSNs [15, 16], network administrators can update programs on sensor nodes, fix bugs, change network functionality, tune module parameters, and replace program module without having to collect the nodes from the field and physically attach them to a computer for burning process. Moreover, individual sensor nodes are able to fetch and install program modules from the network dynamically and on demand. This enables sensor nodes to self-program in order to adapt to changing tasks and evolving environments.

However, DSNs have several limitations, which should be considered when designing any protocols for these networks. Some of these limitations include:

- Limited resources: To reduce the cost and size of individual sensor nodes as well as the entire network, sensor nodes usually have severely limited power supply, computing power, communication bandwidth, and storage capacity. As a result, to prolong the network lifetime, any protocols designed for DSNs are required to be energy-efficient, communication-efficient, storage-efficient, and computationally simple.
- Unreliable wireless communication: Sensor nodes communicate with each other through low-bandwidth shared wireless channels which are far more unreliable than their wired counterparts. This unreliability can introduce a much higher rate of packet collision and loss, and communication latency.

- Random and dynamic topology: In most cases, it is difficult to know the topology of sensor networks a priori due to the random distribution of sensor nodes over inaccessible terrains. Furthermore, the topology and connectivity may frequently vary because of node failure, death, addition, or channel fading.
- Unattended condition: DSNs generally operate without human intervention. An individual node has to establish connections and maintain connectivity autonomously. Sensor nodes are not rescued from physical destruction, battery exhaustion, or malfunction. Therefore, they are usually considered to be expendable sensor nodes.
- Application-specific requirements: Since DSN applications are usually diverse, no unique protocol satisfies the requirements of all applications. A protocol designed and developed for one application needs to be re-designed and re-developed to fit into other applications. This problem prevents the reuse of developed resources and thus increases the cost of the entire application.

1.1.2 Applications

DSNs have been found to be suitable for use in a multitude of applications such as military, environment detection and monitoring, disaster prevention and relief, medical care, home intelligence, scientific exploration, interactive surroundings, and surveillance [1]. This thesis only focuses on the security aspects of DSNs; in support of which this section presents a few specific real-world applications in which security measures are a prerequisite for the success of these applications.

- Vehicle detection and tracking: Researchers from the University of California (UC) Berkeley at the Marine Corps Air/Ground Combat Center in Twentynine Palms of California carried out a well-known experiment [17] to show that a sensor network delivered by an unmanned aerial vehicle (UAV) can be used to detect and track the movement of vehicles. More specifically, they could deploy the sensor network onto a road from the UAV which then can establish a time-synchronised multi-hop communication network among the nodes on the ground, detect and track vehicles passing through the network, transfer vehicle track information from the ground network to the UAV, and transfer vehicle track information from the UAV to an observer at the base camp.

- Mobile antitank land mines: One example of such an application is the Defense Advanced Research Projects Agency funded project developed by a start-up named Sensoria to create mobile antitank land mines that automatically shift position to fill gaps after other mines have detonated [18].
- Counter sniper system: PinPtr [19], an acoustic system, takes advantage of sensor network technology to eliminate the problem of existing counter-sniper systems. Instead of using a few expensive acoustic sensors, the project used a low-cost ad-hoc acoustic sensor network to measure both the muzzle blast and shock wave to accurately determine the location of the shooter and the trajectory of the bullet. PintPtr was built on top of the UC Berkeley MICA2 mote platform [20] running TinyOS [21] together with a custom acoustic sensor daughtercard.
- Dirty bomb detection and localisation: Researchers from the Institute for Software Integrated Systems, Vanderbilt University and Oak Ridge National Laboratory conducted a research project to detect and localise dirty bombs using their radio interferometric positioning technique [22, 23]. In their IPSN 2006 conference demonstration, the dirty bomb detection and localisation process proceeds as follows: (i) Security guards walk around the Vanderbilt Football Stadium with a cell-phone connected to a radiation detector and a Crossbow XSM mote; (ii) The position of the guard is continuously tracked using a radio interferometric technique running on the motes; (iii) A camera automatically tracks the positions using the geolocation information from the mote network; (iv) When the radiation level crosses a threshold the detector sends an alarm and the camera zooms in on the position.
- Oil and gas exploration: HP and Shell announced their collaboration to develop a wireless sensing system to acquire high-resolution seismic data. By vastly improving the quality of seismic imaging, the new system will allow Shell to more easily and cost-effectively explore difficult oil and gas reservoirs [24].
- Hazardous material localisation and terrorist tracking: A new intelligent system (HAMLeT) [25], which is a network of highly-sensitive smell sensors, has been developed to help identify terrorists carrying explosives in security-sensitive areas such as airports, train stations, and governmental buildings. Sensitive electronic noses capture the smell of the explosives; the system processes the acquired data, correlates it with individuals' movements and ultimately tracks

down the suspects. Pursuing the same goal, a group of University of Michigan engineering undergraduate students have developed a system to detect suicide bombers carrying improvised explosive devices (IEDs) using the wireless sensor network technology .

All of the exemplary applications mentioned above are security-critical. None of them would function correctly if appropriate security measures are not taken. The absence of security mechanisms in these applications can pose further security threats to their users. For example, in the applications of detecting and localising dirty bombs and shooters, an attacker can eavesdrop and/or supply misleading information to hide the on-going attack. In the oil and gas exploration application, lacking security algorithms enables business rivals or disgruntled employees to steal commercially sensitive information through eavesdropping and/or intentionally disrupt the manufacturing process via sensed data modification and fabrication.

1.2 Problem Scope and Definition

In enormous DSN applications, security measures such as encryption and authentication play a paramount role in securing network operations due to the fact that DSNs have many characteristics that render themselves highly susceptible to the attacker. These adverse characteristics include the following.

- Constrained resources: As reviewed in the previous section, sensor nodes usually have severely limited processing capability, storage capacity, communication bandwidth, and power supply. Consequently, the well-established security protocols developed for wired and ad hoc networks can not be ported directly to DSNs due to their complexity and energy demands. Innovative security solutions taking the resource constraints into account are essential. The difficulty lies in the fact that a stronger security protocol uses more resources at the sensor nodes, which can result in performance degradation of applications. In contrast, weak security protocols can be broken easily by the attacker. In most cases, a trade-off must be made between security and performance.

- Hostile environments: DSNs are usually deployed in an environment open to adversaries, inclement weather without the support of any fixed infrastructure. It is difficult to perform continuous supervision after network deployment. Therefore, DSNs may face various attacks and failures.
- Unattended conditions: In general, DSNs operate in an unattended fashion for long periods of time. As a consequence, sensor nodes can be open to physical attacks. The attacker can capture unsupervised sensor nodes; tamper with the captured nodes' memory to extract all their secrets that are used in security protocols. The attacker can also place his own malicious nodes into the network to disrupt network operations.
- Wireless communication: Communication among sensor nodes is made possible via wireless channels. These channels are open in the sense that anyone with a wireless interface operating in the same frequency band can monitor or interface with normal protocol exchanges between the nodes of the DSN. This enables the attacker to blend into the networks.

As shown in the literature, solving the key establishment problem plays a key role in enabling such security measures [26-32]. This problem refers to how to efficiently provide a shared key for the purpose of securing the communication link between a pair/group of sensor nodes under adverse conditions such as resource constraints and lack of post-deployment network topology. Over the last few years, research has produced a considerable body of work in developing key establishment schemes using either symmetric-key cryptography [26-36] or public-key cryptography [37-40].

Unfortunately, the key establishment schemes can be a prime target for an attack termed *controlled link establishment attack* (CLEA) whose goal is to gain a significant portion or even full control of DSNs through the controlled link establishment. CLEA leads to the subversion of the great majority of processes which are only accomplished via collaborative efforts of sensor nodes in DSNs, such as data aggregation methods, routing protocols, distributed voting schemes, misbehaviour detection systems. The attack can be exemplified by the node replication attack (NRA) [9, 28], key-swapping collusion attack (KSCA) [41] or the combination of these two attacks due to their same final attack goal. Similar to NRA and KSCA, in order to launch CLEA, the attacker must first mount node capture attack [27, 28, 42, 43] by penetrating into the network to

capture a number of legitimate sensor nodes. Having full control of the captured nodes, the attacker can exploit their unprotected nature to extract their *secret information* [44] such as cryptographic keys or keying materials, cryptographic parameters, node identifiers (IDs). The attacker then abuses the compromised nodes and exposed secret information to establish controlled communication links with legitimate sensor nodes. The defining characteristics of CLEA are summarised as below.

- Node compromise – a prerequisite: In the context of this thesis, it is assumed that the network in general and individual sensor nodes are already protected by security primitives such as encryption and authentication services. This means that any entity which does not possess appropriate secret information has no right to participate in the normal operations of the network. Obviously, to gain this access right, the attacker has to compromise nodes to obtain the necessary secret information.
- Limited number of compromised nodes: This attack considers the situation that the attacker operates in a stealthy manner, attempting to avoid detection and thus evade human attention and intervention. Therefore, the attacker only has the capability to capture and compromise a limited number of legitimate sensor nodes. We assume that the number of compromised links obtained solely from the node compromise is too few to fulfil the attack goal.
- Repetitive usage of compromised nodes and secret information: The attacker distributes the secret information, e.g. compromised keying materials and node IDs, among the controlled nodes in order for them to establish controlled links with uncompromised nodes. From a limited number of compromised nodes, the attacker has to use the compromised nodes and secret information repetitively at different network locations in various permutations to increase the number of controlled links. By this mechanism, a single controlled node can present multiple legitimate IDs and use multiple keying materials. Alternatively, multiple controlled nodes can use a single legitimate ID and keying material. For example, in NRA one ID and the associated keying materials are loaded into multiple generic sensor nodes. Meanwhile, in KSCA the compromised IDs and keying materials are swapped among the fixed portion of controlled nodes to maximise their usability.

- Control of large portion of secured links: This feature is the direct result of repetitive usage of compromised nodes and secret information. Using this overwhelming control, the attacker can cause devastating impacts on a wide variety of crucial services that require cooperative efforts such as data aggregation mechanisms, routing protocols, distributed voting schemes, resource allocation algorithms, and misbehaviour detection systems.
- Means of attack – controlled nodes: It is required that the only means of launching CLEA is via the *controlled nodes* which can be for instance the compromised nodes themselves or the replicas of the compromised nodes. The controlled nodes can communicate and cooperate with each other under the coordination of the attacker. For example, the attacker can use out-of-band channels or mobile unmanned vehicles to link the controlled nodes together.

CLEA is superficially similar to the Sybil attack [45] in that single nodes can present multiple identities. However, in the Sybil attack these pseudo identities are generated randomly instead of legitimate identities being reused according to available secret information and thus the Sybil attacker does not have to compromise sensor nodes. Therefore, the Sybil attack is much less complicated and more straightforward to deal with than CLEA.

The focus of the research which is reported in this thesis is summarised in the following five research questions which stem from the previous discussion of DSNs and CLEA:

- Question 1: What key establishment schemes are vulnerable to CLEA?
- Question 2: What are existing hardware tamper proof techniques to protect sensor nodes from CLEA? What is their vulnerability?
- Question 3: Is it possible to implement CLEA on real-world sensor platforms?
- Question 4: Have there been any existing countermeasures against the attack? What are their strong and weak points?
- Question 5: What are more efficient approaches to thwarting the attack?

1.3 Thesis Objectives

The objectives of this thesis aim for addressing the preceding questions with two main targets. The first target is to demonstrate that CLEA is very feasible in DSNs. The second target is to develop a series of schemes to defend against CLEA by detecting and stopping the controlled nodes' attempt to establish controlled links with uncompromised nodes in a DSN. We aim to detect this attack in a distributed fashion in which any legitimate node in the network can be a detector. Since DSNs are usually left to operate in an attended manner for long periods of time, the attack should be detected and deterred at the first attempt because otherwise the attacker would be able to launch other malicious attacks using the compromised nodes and controlled links. Moreover, the designed schemes should be independent from specific assumptions such as a priori topological and deployment knowledge and undesirable requirements of additional functionalities and resources to be applicable to most, if not all DSNs. Finally, we would like to use symmetric-key cryptographic primitives in the proposed schemes to the most feasible extent in order to save network resources and thus prolong the network lifetime.

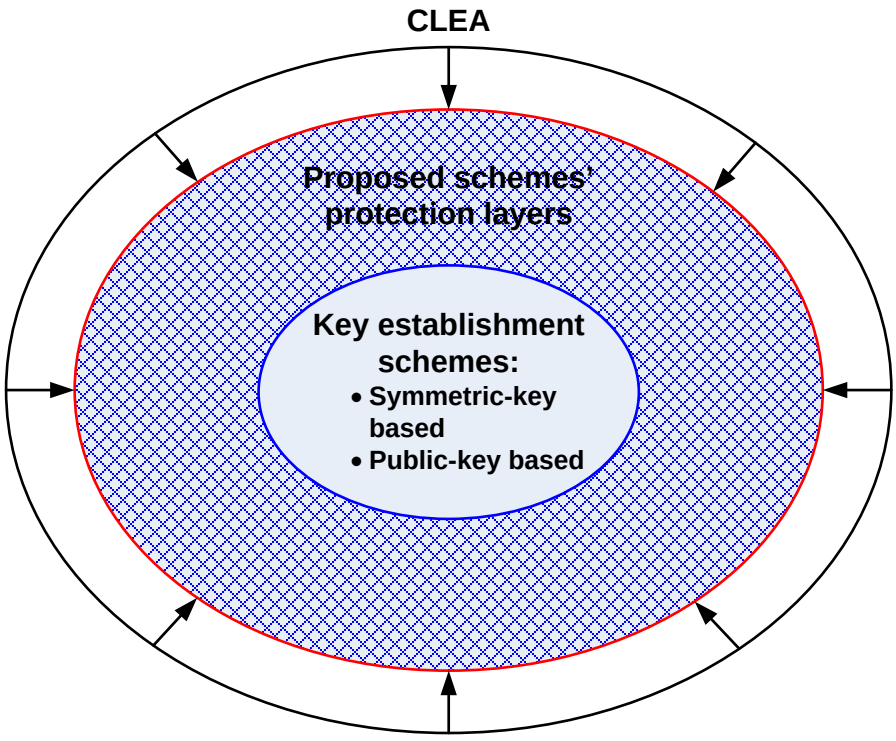


Figure 1.1: Basic concept of the thesis objectives

Figure 1.1 depicts the general concept of the thesis objective. In this concept, the key establishment schemes, either symmetric-key based or public-key based ones, are a target of CLEA when the attacker wants to overwhelm the network with controlled links. The proposed schemes provide protection layers against CLEA. There are two types of protection layers. The first layer is passive in the sense that its application to the key establishment schemes only prevents CLEA from establishing controlled links. The second layer is active in a manner that it not only prevents CLEA but also detects and revokes controlled nodes from normal network operations.

1.4 Design Challenges

Achieving the aforementioned objectives is a challenging task in DSNs due to the following factors.

i) Constrained resources: As discussed in the previous sections.

ii) Unshielded sensor nodes: Due to the fact that sensor nodes are designed to be low-cost, disposable, and massively deployed, in order to minimise the overall cost of the network, they can not afford costly tamper-resistant hardware. Therefore, while left unattended for a long period of time in a hostile environment, sensor nodes can be physically captured and subject to tampering [44, 46, 47]. This ease of node compromise obviously strengthens the incentive for the attacker to mount CLEA.

iii) Powerful attacker: The attacker considered in this thesis is powerful in the sense that the attacker has sufficient resources to randomly compromise a limited number of sensor nodes throughout the network in a distributed fashion without being detected. Furthermore, the attacker is capable of coordinating the controlled nodes via out-of-band channels, for example, to maximise the impact of the attack.

iv) Dynamic network: According to [27], DSNs are dynamic in the sense that they allow addition and deletion of sensor nodes after deployment in order to grow the network or replace failing and unreliable nodes. Hence, all of the DSN security algorithms are expected to provide network scalability to accommodate new added nodes. This also means that the DSN security algorithms are open to the controlled nodes as well.

v) Unknown deployment knowledge and post-deployment network topology: This factor implies that, before deployment, a sensor node is assumed not to have any knowledge about which nodes will be its neighbours and will have secure communication links with itself. Therefore, the node can not tell whether a request for secure link establishment is from a node geographically close to its location or from a distant node (controlled node).

1.5 Thesis Contributions

This dissertation reports on an investigation of CLEA and the principles of countermeasures against the attack by providing solutions to each of the research questions given in the previous section. To this end, we make the following contributions:

i) A comprehensive survey of the key establishment schemes which are vulnerable to CLEA to address question 1. It emphasises the widespread and devastating effect of CLEA on various key establishment schemes.

ii) A comprehensive survey of existing countermeasures against CLEA and the identification of their limitations to address the fourth question.

iii) A review of hardware tamper-proof technologies as well as their vulnerability to address the second question.

iv) A detailed description of how disclosed secret information can be utilised repetitively at different locations and the prototype implementation of CLEA in the form of KSCA to address the third question.

v) A new countermeasure against CLEA via cryptographic secret protection. This addresses the fifth question and consists of the following sub-contributions:

- One-way hash chain based scheme (OWHCBS): This scheme confronts CLEA by minimising the utility of compromised nodes or secret information to the attacker. This is achieved by making use of the combination of the multi-phase deployment nature of DSNs and an one-way hash chain (OWHC) [48]. Accordingly, secret information in sensor nodes of each phase is protected via encryption by one different key taken in a specific order from the OWHC. Regardless of their secret information being encrypted, sensor nodes of the

previous phase can still establish secure links with those of the later phase thanks to the property of the OWHC.

- Diversified one-way hash chain based scheme (DOWHCBS): As can be seen, the security of OWHCBS relies on the security of OWHC keys. If the attacker obtains one OWHC key, the attacker can reconstruct the previous OWHC keys of the chain and then use the exposed keys to launch CLEA. To improve the security of the OWHC based scheme, we develop the diversified OWHC based scheme in which the OWHC keys are diversified after secure link establishment. Owing to this diversification, the attacker finds it more difficult to recover the keys for CLEA attack.
- Hidden one-way hash chain based scheme (HOWHCBS): The security analysis reported here shows that DOWHCBS is still vulnerable to a sophisticated attack which can recover the OWHC keys from their diversified forms. To overcome this security threat, a new technique to hide the OWHC key in HOWHCBS is introduced in Chapter 6.

vi) A new countermeasure against CLEA via controlled node detection and revocation. This contribution also address the last question and comprises the following sub-contributions:

- Naïve detection scheme: This scheme is the simplest one of the three controlled node detection schemes. The basic idea is that each node maintains a counter to keep track of the node deployment and addition activity in its neighbourhood. This counter increases by 1 after a constant time interval ΔT. Each incoming node attempts to join the network by broadcasting a HELLO message which contains the newest constant counter value. The neighbouring nodes admit the incoming node only if their maintained counter values are equal to the one in the received HELLO message.
- Adaptive detection scheme: Examination of the naïve scheme exposes several problems. First, newly deployed legitimate nodes might fail to join the network when ΔT (see table 2.1) is small and/or because of counter de-synchronisation. Second, the attacker might have sufficient time to mount CLEA when ΔT is large. In the light of this examination, the adaptive detection scheme is proposed to tackle the problems.

- Extended adaptive detection scheme: Flaws in the adaptive scheme might cause it to be susceptible to sophisticated backward and forward CLEAs. This thesis shows how the scheme can be upgraded to the extended adaptive detection scheme to confront these attacks. The extended scheme provides much greater security in comparison with the adaptive scheme by trading off small performance.

1.6 The Organisation

The balance of this dissertation is organised as follows. The network model, node model, service model, security assumptions, and adversary model are discussed in Chapter 2. Chapter 3 introduces related work on the key establishment schemes for DSNs, NRA, KSCA, and corresponding countermeasures together with their limitations. Chapter 4 provides building blocks for the development of our proposed countermeasures. Chapter 5 demonstrates the feasibility of CLEA. Chapter 6 presents the details of our secret information protection based countermeasures against CLEA. Chapter 7 develops the per node deployment based detection schemes against CLEA. Finally, Chapter 8 summarises the thesis and concludes with recommendations for future work.

1.7 Publications Related to the Thesis

During the course of investigation for this dissertation, the following papers have been published or in press.

1.7.1 Book Chapters

Thanh Dai Tran and Johnson Ihyeh Agbinya, "Security and Key Establishment Schemes for Distributed Sensor Networks," in Biomedical and environmental sensing, E. B. J.I. Agbinya, Y. Hamam, F. Rocaries, S. K. Lal, Ed. Aalborg, Denmark: River Publishers, 2009, pp. 181-217.

Thanh Dai Tran and Johnson Ihyeh Agbinya, "Network Invading Attack on Key Pre-distribution Schemes for Distributed Sensor Networks and Countermeasures," in

Security of Self-Organizing Networks: MANET, WSN,WMN, VANET, A.-S. K. Pathan, Ed. USA: Auerbach Publications, 2010.

1.7.2 Journal Articles

Thanh Dai Tran and Johnson Ihyeh Agbinya, " Combating Key-swapping Collusion Attack on Random Pairwise Key Pre-distribution Schemes for Wireless Sensor Networks," Security and Communication Networks, Vol. 4, No. 2, pp. 109–121, 2011.

Thanh Dai Tran, Adel Al-Jumaily, and Johnson Ihyeh Agbinya, "Per Node Deployment Based Detection of Controlled Link Establishment Attack in Distributed Sensor Networks," Int. J. Sensor Networks, in Press, 2011.

1.7.3 Peer Reviewed Conference Papers

Thanh Dai Tran and Johnson Ihyeh Agbinya, " A Framework for Confronting Key-swapping Collusion Attack on Random Pairwise Key Pre-distribution Schemes for Distributed Sensor Networks," in Proc. 5th IEEE Int'l Conference on Mobile Ad Hoc and Sensor Systems (MASS'08), Atlanta, Georgia, USA, October 2008, pp. 815-820.

Thanh Dai Tran and Johnson Ihyeh Agbinya, "Non-threshold Deterministic Parwise Key Establishment Scheme for Clustered Distributed Sensor Networks," in Proc. of 3rd Int'l Conf. on Broadband Communications, Information Technology & Biomedical Applications, Pretoria, South Africa, November 2008, pp. 44-49.

Thanh Dai Tran and Johnson Ihyeh Agbinya, "Early and Lightweight Distributed Detection of Node Replication Attack in Sensor Networks," in Proc. IEEE Wireless Communications and Networking Conference 2010 (WCNC 2010), Sydney, Australia, April 2010, pp. 1-6.

Thanh Dai Tran and Johnson Ihyeh Agbinya, "Revisiting key-swapping Collusion Attack in Sensor Networks," in Proc. 4th International Conference on Sensor

Technologies and Applications (SENSORCOMM 2010), Venice, Italy, July 2010, pp. 381-388.

Thanh Dai Tran and Johnson Ihyeh Agbinya, "Feasibility of Key-swapping Collusion Attack on Distributed Sensor Networks," in Proc. 5th International Conference on Broadband and Biomedical Communications (IB2Com'10), Malaga, Spain, December 2010.

Chapter 2

Models and Assumptions

This chapter specifies the models and underlying assumptions about sensor nodes, sensor networks, security, and the potential capabilities of attackers which are used throughout this thesis.

2.1 Network Assumption and Model

Throughout this thesis, we consider large-scale DSNs which comprise thousands of small homogeneous sensor nodes controlled and monitored by a single authority and uniformly distributed at random over a wide and hostile area of interest. There is no topological information available before deployment. There is also no explicit assumption made about an online powerful base station, yet there does exist an off-line security server which is responsible for pre-configuring sensor nodes.

Sensor nodes are battery-operated with limitations of memory storage, computational capability and communication bandwidth. One particular example is a Crossbow's MICAz mote [49] which has 8-bit 8-MHz processor, 4KB primary memory (SRAM), 4KB EEPROM and 128KB program flash memory. They are usually expected to be unshielded by tamper-resistant hardware to reduce the cost of their massive deployment. After deployment, nodes are stationary and cooperate among themselves in an entirely distributed manner allowing operators to retrieve aggregated data from any of the nodes in the network.

The network deployment can be carried out through aerial scattering, physical installation, or unmanned vehicles. In this thesis, we also adopt the network deployment

model presented in [27]. Specifically, there is only one deployment for the entire network lifetime in which the majority of sensor nodes are randomly scattered onto the field. Sensor nodes are allowed to be added a few times in small quantities after the deployment only to either grow the network or replace failing and unreliable nodes. Therefore, nodes from the first deployment and later deployments/additions are grouped respectively into generations (or groups) indexed from 1 to t.

A DSN can be described using graph theory language. Accordingly, the entire network can be modelled as a graph $G(\mathbb{V},\mathbb{E})$ of a vertex set $\mathbb{V}$ and an edge set $\mathbb{E}$. A vertex v ($v \in \mathbb{V}$) in the graph represents a node in the network and an edge e ($e \in \mathbb{E}$) represents a communication link, usually a wireless link. Providing that two nodes are able to communicate directly, there exists an edge connecting them in the graph. In this case, it is said that the two nodes are adjacent or they are neighbouring nodes. In normal condition, the graph G is connected, i.e. for every $u,v \in \mathbb{V}$ in G there exists a u,v-path, a sequence of vertices beginning at u and ending at v such that from each of its vertices there is an edge to the next vertex in the sequence.

In this dissertation, it is assumed that the communication among sensor nodes is through a single local broadcast channel. This means that any nodes in the transmission range of a node will definitely hear the data transmitted by that node. In contrast, any node located outside the node's transmission range cannot communicate with the node. As illustrated in Figure 2.1(a), nodes S_1 and S_2 can communicate with each other while S_1 and S_3 cannot. In wireless communication, transmitting and receiving issues such as packet loss, packet corruption, transmission delay may occur due to collision, unreliable physical channels, and contention. These issues can be alleviated significantly through reliable communication mechanisms such as media access controls, acknowledgment and retransmission mechanisms.

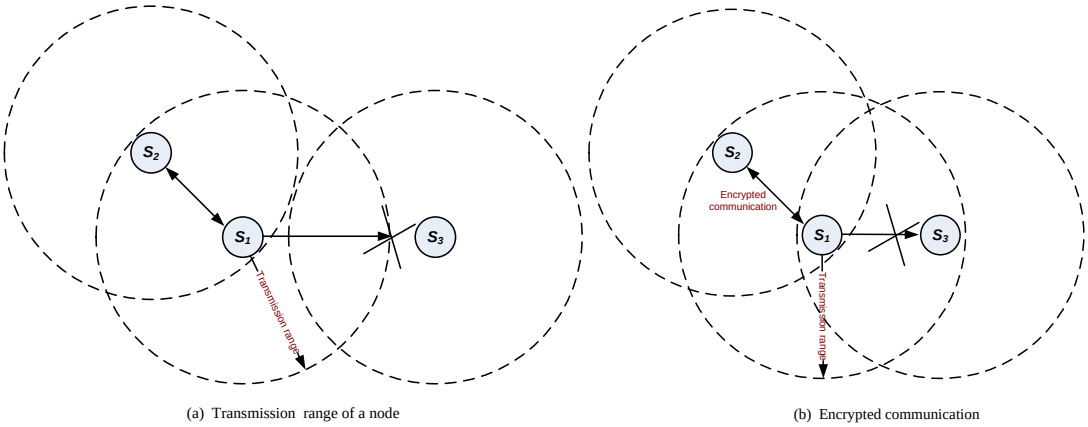


Figure 2.1: Illustration of local broadcast communication in DSNs

It is the case, as shown in Figure 2.1(b), that even through a node is in the communication range of a sender, the node cannot communicate with the sender. The reason is that the sender's messages are encrypted and destined for other node (receiver) which shares a secure key with the sender.

Typically, there are two or three types of nodes in the network: base station, aggregator (optional), and sensor node. A base station is a node which issues queries to collect data from sensor field, processes the data and provides them for interested users. In general, the base station can be any node in the network. In this dissertation, no specific assumption is made about features and functions of the base station. It does not play any important role in the proposed schemes. A sensor node senses elements of its surrounding environment and sends sensed data to aggregators (if they exist) for data aggregation or to the base station in a multi-hop communication manner. To join the network, new sensor nodes require neither administrative intervention nor interaction with the base station. Instead, they initiate simple neighbour discovery protocols [27, 28] by broadcasting their pre-loaded information such as unique node IDs and/or unique key IDs.

2.2 Security Model

DSN security can be grouped into two broad categories: (i) operational security, and (ii) information security [50]. The objectives of the operation-related security are to fulfil the following properties.

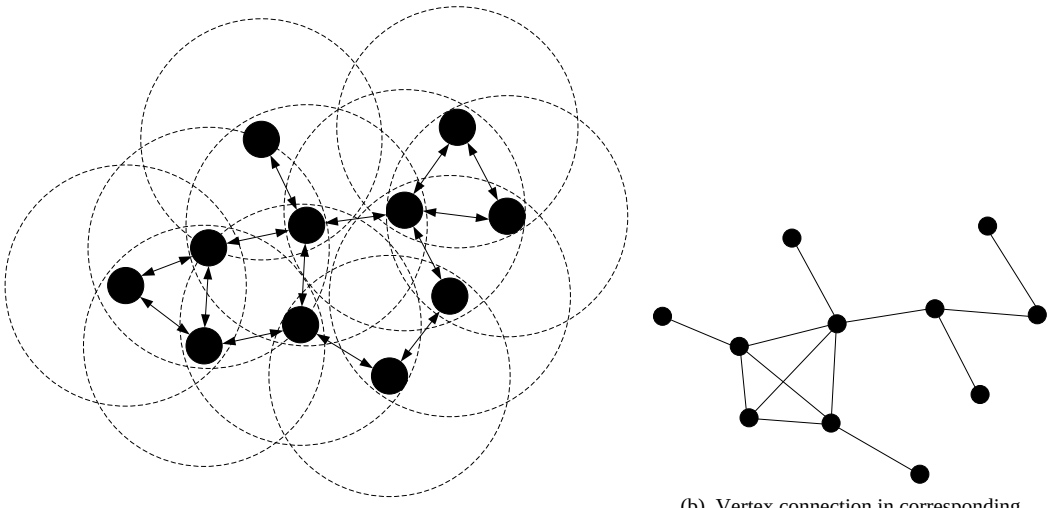
- Availability: Ability to ensure that services offered by the network as a whole must be available even when some of its components are under attack.
- Survivability: Ability to provide a minimum level of service in the presence of power loss, failures or attacks.
- Degradation of security services: Ability to change security level as resource availability changes.

The objectives of the information-related security are to meet the following security requirements.

- Authentication: Authenticating other nodes, cluster heads, and base stations before granting a limited resource, or revealing information. It is also related to authentication of data source and data in transit.
- Integrity: Ensuring that message or the entity under consideration is not altered.
- Confidentiality: Providing privacy of the wireless communication channels to prevent eavesdropping.
- Freshness: In many cases, sensor networks are used to monitor real-time events, it is thus important to ensure that the data provided by the network is fresh at all times. In other words, the attacker cannot replay old messages in the future.
- Non-repudiation: Preventing malicious nodes from hiding their activities.

The basis for information security in DSNs assumes that security services such as encryption and authentication mechanisms are established prior to the exchange of date. These services can be enabled by having sensor nodes follow key establishment schemes [27-30, 37, 39, 51-53] to establish secret pairwise/group keys with their neighbours. After secret keys are set up among neighbouring sensor nodes, the entire network forms a key-sharing graph $G_{ks}\left(\mathbb{V}_{ks},\mathbb{E}_{ks}\right)$ [29]. In essence, $G_{ks}\left(\mathbb{V}_{ks},\mathbb{E}_{ks}\right)$ is a spanning graph of $G\left(\mathbb{V},\mathbb{E}\right)$ in which $\mathbb{E}_{ks}$ is constructed in the following manner: An

edge is drawn between any two vertexes $u, v \in \mathbb{V}_{ks}$ if and only if (i) the two nodes corresponding to u, v share at least one secret key, and (ii) the two nodes are in the transmission range of each other. The drawn edge corresponds to a secure communication link in the network. Figure 2.2(a) shows the wireless connection of eleven nodes in a DSN while Figure 2.2(b) illustrates the key-sharing graph of the DSN after key establishment.



(a) Node connection in a DSN

(b) Vertex connection in corresponding key-sharing graph

Figure 2.2: Illustration of connection in a DSN and corresponding key-sharing graph

2.3 Adversary Assumptions

Generally speaking, it is impossible to achieve perfect security because a powerful strong-incentive attacker can always devise methods for defeating security measures. Furthermore, establishing and operating a comprehensive security mechanism to defend against and respond to every possible attack prove to be prohibitively costly. Therefore, defining what we want to secure against and what the adversary model is plays important role in designing and developing a robust security measure.

Attacks on sensor nodes in particular and DSNs in general have been explored extensively in the research literature. The Dolev-Yao well-established model of the attacker in [54] assumes that the attacker knows all the details of the algorithms we use.

However, the adversary model considered in this thesis is different from the Dolev-Yao model in several aspects. First, the Dolev-Yao model allows the attacker to read and write messages at arbitrary locations in the network while in this discussion the attacker is restricted to read and write messages using only the nodes under its control. Second, the Dolev-Yao model does not allow the attacker to compromise and replicate legitimate nodes in an adaptive manner, whereas these capabilities are available to our attacker.

In practice, our adversary model accounts for a myriad of potential attacks in DSNs especially when they are deployed in hostile environments. Because of the broadcast nature of wireless communications, the attacker can intercept messages, modify packets, replay old messages, inject bogus data, block or analyse network traffic. Since sensor nodes are left unattended, the attacker is able to furtively capture a number of legitimate sensor nodes randomly over time. However, this number is limited because if most or all of the nodes in the network are under the control of the attacker, apparently all security mechanisms will collapse. Having obtained these nodes, the attacker can exploit the lack of tamper-resistant hardware to compromise them. Once compromised, all secret information stored in the captured nodes can be extracted by the attacker. The attacker can then duplicate multiple controlled nodes which are copies of the compromised nodes using extracted secret information. The attacker also can reprogram and load multiple extracted secrets into a single controlled node. These controlled nodes can be then injected back into the network at strategic locations, communicating and collaborating with each other to launch other malicious attacks.

In terms of node compromise, this thesis considers two types of attack models as follows.

- Class I (Sporadic attack): They are able to capture sensor nodes randomly and independently. Their decision of capturing a node does not depend on what they have obtained from previous compromised nodes. However, as assumed in many recently published works [52, 55-59], they are not able to compromise newly deployed nodes during a short secure interval Y_s since their deployment. Y_s (less than 60s as estimated in [53]) is the time needed for newly deployed/added sensor nodes to complete the key establishment process. In

other words, the time interval I_k for a newly added node to complete its key establishment task is smaller than the time interval I_c, a lower bound for the attacker to compromise the node. This node compromise model is reasonable due to the following reasons: (i) The success of the first generation deployment plays the most important role in completing the mission of an entire sensor network. Thus, this deployment is most likely to be secure against the attacker's physical access until pairwise/group keys are established. (ii) Node capture and compromise take the attacker's time since the attacker must first gain a physical access to a sensor node, connect to the node, and then use some programming tools in order to extract its secret information. (iii) Since the number of nodes of later deployments is small in quantity, it takes a longer time for the attacker to encounter newly deployed nodes.

- Class II (Continuous-time attack): The attacker may have the capability to compromise nodes of a newly deployed generation to obtain secret information over Y_v since the deployment of the generation. Similar to the sporadic attack, the attacker of this type cannot have access to the secured nodes of the first deployment until pairwise/group keys are established.

In this thesis, it is further assumed that the attacker has no capability to create new legitimate IDs for nodes. The reason is that a legitimate node's ID is usually tied to some security mechanisms. For instance, the ID is signed by the node's private key using some asymmetric key cryptosystem. In the network using a key pre-distribution scheme [27, 28], a node's ID can be associated with the set of secret keys. In such a system, the attacker benefits almost nothing by claiming to have an ID without actually holding the appropriate keys.

2.4 Notation

In Table 2.1, we list some important symbols used through out the dissertation.

Table 2.1: List of symbols used in the thesis

Symbol	Description
t	Number of sensor node deployment and additions over the network lifetime; number of generations/groups
Y_s	Short secure interval needed for key establish process accomplishment during which no node compromise occurs
I_k	Time interval for a newly added node to complete its key establishment task
I_c	Lower bound of time interval for the attacker to compromise a node
Y_v	Short interval needed for key establishment process accomplishment during which node compromise can occur
S_i	Sensor node ID
n	Number of sensor nodes in the network; maximum network size
n'	Average number of node in a neighbourhood; neighbourhood size
$n(i)$	Average number of nodes in a neighbourhood after ith generation deployment
m	Size of key ring in each sensor node
p	Probability of a shared key existing between two sensor nodes
c_i	Number of compromised/controlled nodes of ith generation
$S_{i,j}$	Name/ID of the ith node of the jth generation
$GenID_i$	ID of ith sensor node generation
K_i	Key encryption key (KEK); ith element of a OWHC
$r_{i,j}$, $f_{i,j}$	Random number (salt) generated by $S_{i,j}$
$K_{i,j-1}$	Key recovery key (KRK) in $S_{i,j}$ such that $K_{i,j-1} = H(K_i \| r_{i,j})$
$K_{ij,k}$	kth plaintext key in $S_{i,j}$'s key ring
$E_{S_{i,j}}$	Encrypted key ring in $S_{i,j}$: $\{E_{K_j}(K_{ij,1} \| K_{ij,2} ... \| K_{ij,m}\}$
$e_{S_{i,j}}$	Initial encrypted key ring in $S_{i,j}$: $\{E_{K_j}(K_{ij,1}),..., E_{K_j}(K_{ij,m})\}$
$e_{S_{i,j}k}$	Remaining encrypted key ring in $S_{i,j}$ after key establishment with kth generation deployment

T_i	Deployment time of $(i+1)st$ generation
δ	Number of time slots between two deployment points of time
λ_1	Geometric distribution parameter of the population of generations
λ_2	Poisson process rate of node capture/compromise
p_c	Probability of at least one sensor node being compromised over one interval Y_v
p_{ci}	Probability of at least one sensor node of ith generating being captured/compromised over Y_v
c_i	Expected number of ith generation nodes being captured/compromised within the interval Y_v
$c_{1\to i}$	Expected number of captured/compromised nodes within intervals Y_v from generation 1 to generation i
$\Phi_{i,j}$	Set of secret information generated by a given key establishment scheme for $S_{i,j}$
$\|\|$	Concatenation symbol
$E_{K_i}(X)$	Encrypting X using key K_i
$D_{K_i}(X)$	Decrypting X using key K_i
E	Cost of one symmetric encryption operation
D	Cost of one symmetric decryption operation
H	Cryptographic one-way hash function, e.g. MD5 or SHA-1; Cost of one hashing operation
$H^y(X)$	Hash value of X after y one-way hashing operations or an element of the OWHC.
H_k	Keyed hash function, e.g. HMAC-MD5 or HMAC-SHA-1
L	Network lifetime
A_c	Event that K_c is compromised over the network lifetime
X_a	Event that K_a is compromised over the interval I_k
Y_a	Event that an ath generation node is compromised
Z	Event that node is in interval of establishing pairwise keys I_k
$P(X)$	Probability that event X occurs

ΔT	Time interval after that counter values increase by 1 by default
T_i^S	Time point i at security server's side
T_i^n	Time point i at sensor node's side
ΔT_{i+1}	Label of time interval between T_i^S and T_{i+1}^S
δT	Small time window for secure link establishment
C_s	Server counter
C_i	Counter of sensor node S_i
$\varepsilon_{S,\tau}$	Value of server counter at τ-th interval
$\varepsilon_{i,\tau}$	S_i's counter value at τ-th interval
s_i, N_i, $s(i)$	Population of ith node generation; Number of ith deployment nodes
$N(i)$	Total number of nodes from first generation to ith generation
$HelloMsg_{i,\tau}$	S_i's HELLO message generated for use over τth interval
$t_{S,k}$	Moment when kth node deployment occurs
$\varepsilon_{j,k}^*$	Constant counter value in $HelloMsg_{j,k}$
Δ_d	Delay in receiving the first legitimate HELLO message since node deployment
δ_{diff}	Upper limit of the clock difference between the security server and any sensor node
$\delta_{d,i}$	The clock difference between the security server and S_i
r	Number of line segments per node in the extended adaptive scheme
l_λ	Node S_λ's authenticated location claim
α	Average number of nodes on each line segments

Chapter 3

Literature Review

The goal of this chapter is to provide a comprehensive survey of issues directly related to the main topics of this thesis which are CLEA and its countermeasures. The chapter starts with the key establishment issue in sensor networks by first presenting a taxonomy of key establishment schemes. It then classifies the key establishment schemes according to their level of vulnerability to CLEA. The classification indicates that no key establishment schemes are totally immune from CLEA. Thereafter, it details NRA and KSCA as the concrete instances of CLEA followed by existing indirect and direct countermeasures which may be utilised to thwart CLEA. The chapter concludes that CLEA is a serious threat yet no countermeasures found in the literature are robust and efficient enough against the attack. As a result, new countermeasures will be indispensable for secure operations in DSNs in the presence of CLEA.

3.1 Key Establishment in Sensor Networks

3.1.1 Taxonomy of Key Establishment Schemes

Traditionally speaking, key establishment is a process or protocol whereby a shared secret becomes available to two or more parties, for subsequent cryptographic use [60]. The proposed key establishment mechanisms for sensor networks differ in various aspects because DSNs operate in significantly different environments with different levels of security threat, supporting different applications with different communication patterns and network topologies, and using different sensor nodes with different resource capabilities. These key establishment schemes can be classified broadly according to employed cryptographic techniques including symmetric cryptography and

asymmetric cryptography. The schemes can also be categorised based on methods whereby shared keys are generated and delivered into key transport and key agreement protocols. In addition, they can be classified into static or dynamic approaches based on whether rekeying (update) of administrative keys is supported after network deployment [61]. Another classification criterion that can be used is the role of individual network entities in the key management process. In a homogeneous scheme, all nodes perform the same functionality; on the other hand, nodes in a heterogeneous scheme are assumed different roles. Homogeneous schemes are generally designed for flat network architecture; meanwhile heterogeneous schemes are intended for both flat and clustered network topologies. Note that in the taxonomy shown in Figure 3.1, only prominent representatives of each type of the key establishment schemes are discussed and taken into account.

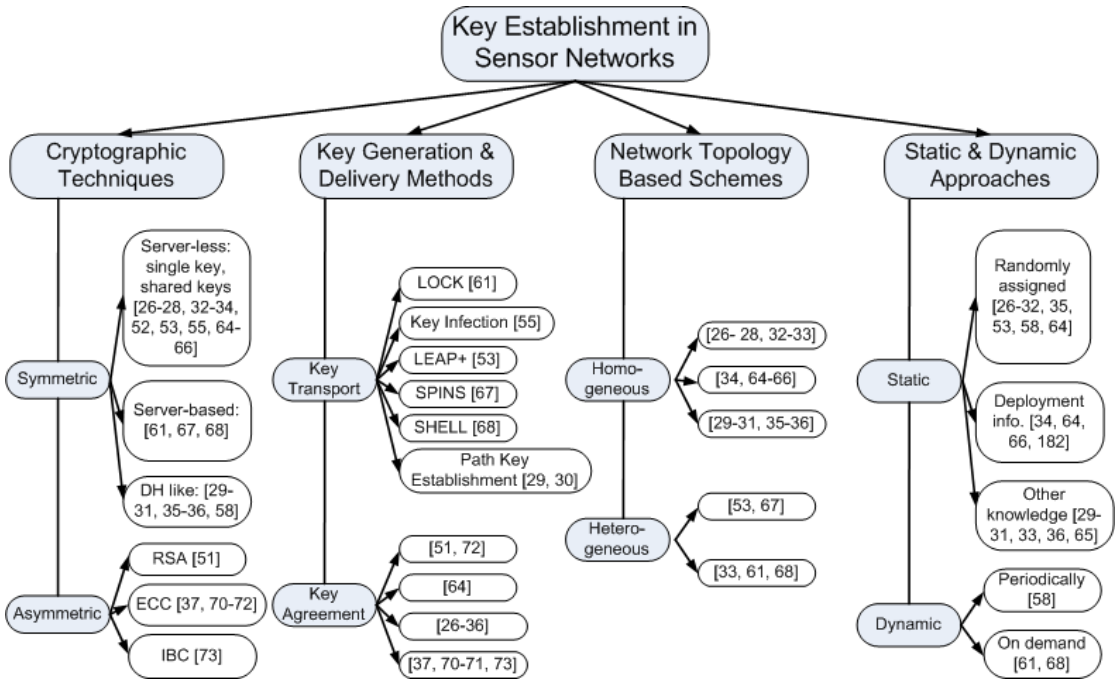


Figure 3.1: Classification of key establishment schemes in sensor networks

3.1.1.1 Symmetric Techniques vs. Asymmetric Techniques

Like its traditional counterpart, the key establishment problem in sensor networks can be approached through symmetric-key techniques or public-key techniques.

- Symmetric techniques: In this approach, the key establishment is performed via the aid of symmetric-key primitives such as symmetric-key ciphers, arbitrary length hash functions [62, 63], signatures, and pseudo random sequences. In addition, symmetric-technique based key establishment schemes can also be identified according to key (keying) distribution models such as server-less schemes including single network-wide key scheme, pairwise shared keys scheme [26-28, 32-34, 53, 55, 64-66]; server-based schemes [61, 67, 68]; and Diffie-Hellman like schemes [29-31, 35, 36, 58].
- Asymmetric techniques: In contrast to the proliferation of symmetric-key based methods, applications of asymmetric (public-key) techniques in resolving the key establishment problem for sensor networks have remained in an early stage due to the limited capabilities of sensor nodes in terms of memory storage, computation, communication and power supply. Most of the research efforts focus on demonstrating feasibility of public-key algorithms on sensor platforms via requirements of special hardware support, algorithm and instruction-level optimisation for particular architectures [37, 51, 69-71]. Public-key based key establishment schemes can be subdivided into several types based on the utilised cryptography such as RSA [51], elliptic curve cryptography (ECC) [37, 70-72], and identity-based cryptosystem (IBC) [73].

One straightforward observation drawn from this classification is that symmetric-key approaches for establishing shared keys in sensor networks are still preferred choices since they involve only simple cryptographic operations, have low cost, and can be implemented independent of the architecture.

3.1.1.2 Key Transport Protocols vs. Key Agreement Protocols

- Key transport protocols: Classically speaking, a key transport protocol or mechanism is a key establishment technique where one party creates or otherwise obtains a secret value, and securely transfers it to the other(s) [60]. In this sense, there are a number of schemes and protocols for sensor networks worth being named such as LOCK [61], key infection [55], LEAP+ [53], SPINS [67], SHELL [68] , and path key establishment [29, 30].

- Key agreement protocols: As for key agreement schemes, during the initialisation stage, secret information is generated by a trusted server and distributed to users, such that any pair/group of users may subsequently compute a shared key unknown to the others (aside from the server). By this definition, the schemes in [26-37, 51, 64, 70-73] can be taken into account.

3.1.1.3 Static Schemes vs. Dynamic Schemes

- Static schemes: In these schemes, administrative and link keys are assumed to be preloaded into the nodes once for their whole life, they will not be updated. Administrative and link keys are generated prior to deployment, assigned to nodes randomly [26-32, 35, 53, 58, 64], based on some deployment information [34, 64, 66], or other knowledge such as attack probabilities [29-31, 33, 36, 65] and then distributed to nodes. When these keys are disclosed due to node compromise, they are revoked and deleted from related sensor nodes via key or node revocation schemes [27, 28, 74].
- Dynamic schemes: Dynamic key management schemes may change administrative keys periodically [58], on demand or on the detection of node capture [61, 68]. The major advantage of dynamic keying is the enhancement of network survivability, since any captured key(s) is replaced in a timely manner in a process known as rekeying. Another advantage of dynamic keying is the provision of better support for network scalability; upon adding new nodes, unlike static keying in which a fixed pool of keys is used, the probability of network capture does not necessarily increase [61]. However, the downside is also worth considering. In order to operate properly, these schemes require either powerful hardware components which are sometimes expensive to satisfy or costly software functionalities such as intrusion detection systems.

3.1.1.4 Network Topology Dependent Schemes

- Homogeneous schemes: As mention earlier, in homogeneous schemes, the role of each network node in establishing shared keys is identical except for the base stations. More specifically, each node is pre-loaded with some secret

information before sensor deployment. This secret information is then used by adjacent communicating nodes on the fly after deployment to establish shared keys in absence of superior nodes [26-36, 64-66] . Such schemes are widely used in the applications that make use of the flat network topology wherein hop-by-hop communication is the dominant communication pattern in DSNs.

- Heterogeneous schemes: Heterogeneous schemes are designed to support multi-tier architectures wherein different network nodes take on different network missions. For example, normal sensor nodes are used to sense interested events and deliver their readings to a cluster head (or data fusion node, aggregation point). The readings are then processed and aggregated there before being forwarded to the base station. Since there are several types of communications between different network nodes of different roles, a generic single keying mechanism is not suitable for meeting security requirements. Typically, heterogeneous schemes support the establishment of several types of keys: a pairwise key shared between two neighbouring sensor nodes, a pairwise key shared by a sensor node and a cluster head, an individual key shared between a sensor node and the base station, a cluster key shared by all nodes in the cluster, and a global key shared by all the nodes in the network [53, 67]. Furthermore, cluster heads are further assumed to be more powerful than normal sensor nodes and have extra functionalities to ease the burden of the key management process [33, 68].

3.1.2 Vulnerability of Key Establishment Schemes to CLEA

The vulnerability level of key establishment schemes for sensor networks to CLEA varies from scheme to scheme according to their approaches and to whether the anti-CLEA feature has been built-in or not. Table 3.1 below lists the aforementioned key establishment schemes and their corresponding levels of CLEA vulnerability. These schemes are marked with a cross in the table if they possess the corresponding vulnerability feature.

3.2 Node Replication Attack

The concept of NRA was first mentioned in the research work presented by Chan et al. [28]. The pioneering research on NRA was conducted by Parno et al. [9] followed by a series of

Table 3.1: Classification of key establishment schemes in terms of their vulnerability level to CLEA

Schemes/Class of schemes	Totally vulnerable	Highly vulnerable	Partially vulnerable	Least vulnerable
Server-less, random assignment [26-33, 35, 58]	X			
Server-less, random assignment [64]		X		
Server-less, non-random assignment [36, 55]	X			
Server-less, non-random assignment [52, 53, 75, 76]			X	
Deployment and/or location knowledge [34, 66]			X	
Deployment and/or location knowledge [65]				X
Asymmetric [37, 51, 70-73]	X			

research efforts [77-79] to thwart the attack. In a different approach, Fu et al. [80] focused on analysis, characterisation, and discussion of the relationship among the replicas, sensor networks, and resiliency of various pairwise key establishment (PKE) schemes against NRA. Their study aims at providing practical insights into the design of more secure and efficient key establishment schemes rather than an attack countermeasure.

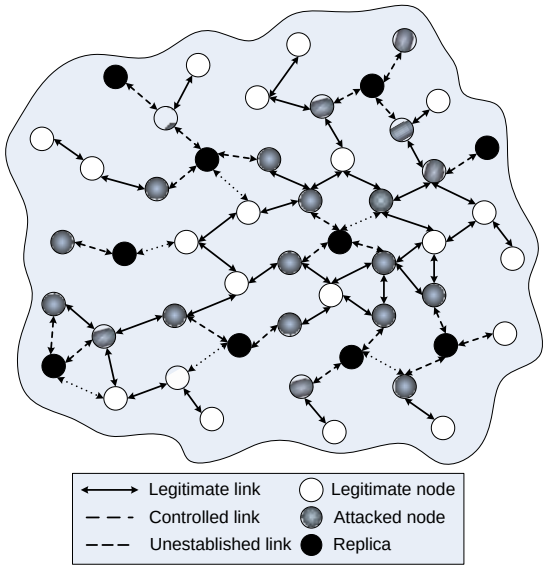


Figure 3.2: Illustration of node replication attack

It is assumed that to launch NRA, the attacker has to execute four steps: (i) *compromising nodes*, (ii) *generating replicas*, (iii) *inserting replicas*, (iv) *establishing controlled links*. Firstly, the attacker manages to stealthily infiltrate or break into the network to capture a limited number of legitimate nodes. Having captured these nodes, the attacker can exploit the unshielded nature of the nodes to extract their secret information [44]. Secondly, having obtained the secrets, the attacker then duplicates the compromised nodes by loading the obtained secrets onto multiple generic nodes. Thirdly, the attacker inserts the replicas back into the original network. The attacker can realise this in several ways such as physical installation of each node, or random scattering. Finally, the replicas attempt to establish pairwise/group keys with legitimate nodes. The detailed process of establishing keys varies according to the implementation of key establishment schemes, which is discussed in Section 3.1. The links secured by the established keys are called controlled links because they are under the control of the

replicas. In addition, these replicas are allowed to communicate and collaborate with each other under the attacker's control. Figure 3.2 illustrates NRA with 10 replicas in a network of 50 nodes.

Once successfully mounted, NRA can cause substantially disastrous impacts on DSNs. Indeed, attackers can carry out a wide variety of malicious attacks on various mechanisms and protocols using their control of replicas and controlled links. Several potential attacks to name but a few are described briefly as below.

- Routing attacks: As pointed out in [81], making illicit use of controlled links, replicas can corrupt routing tables of their neighbours by spoofing, altering, or replaying routing information. Consequently, the attacker is able to create routing loops, attract or repel network traffic, lengthen or shorten source routes, generate false error messages, partition the network, increase end-to-end latency, or counteract redundant/multi-path routing.
- Data aggregation attack: Many sensor applications [1, 82-84] emphasise the importance of data aggregation or in-network processing in eliminating data redundancy, minimising the number of transmissions, and thus saving network resources such as energy, bandwidth to prolong network lifetime. However, by using the replication attack, the attacker having replicas reporting incorrect sensor readings might be able to significantly and badly skew the computed aggregates network-wide.
- Distributed voting attack: Due to the lack of infrastructure support and being distributed by nature in most sensor applications, many sensor network protocols and mechanisms make use of a distributed voting scheme using sensor nodes as voters to resolve tasks that require collaborative decision making. Consequently, this renders the network vulnerable to an attack. In this attack, the attacker via replicas can cast biased votes in an overwhelming number to gain the right of determining the outcome of any decision making process. For instance, in reputation based schemes [28], the attacker can revoke legitimate nodes by claiming that the nodes are misbehaving. Additionally, the attacker can also use votes to protect the replicas from being detected.
- Resource allocation attack: In some scheduling algorithms such as medium-access control (MAC) protocols, network resources are allocated soundly only

based on legitimate operations of sensor nodes. In such cases, the replicas can be ordered not to follow the algorithms to obtain an unfair share of any resources and, thus, putting legitimate nodes at resource starvation.

3.3 Key Swapping Collusion Attack

While NRA can break through the defence of both public-key based key management schemes and symmetric-key based key pre-distribution schemes (KPSs), KSCA can be made possible to the latter only. This attack first studied by T. Moore in [41] emerges from the fact that while the compromised cryptographic secrets can be illegitimately utilised network-wide via the collusion among the controlled nodes, this utilisation is unlikely to be detected due to the absence of collaboration among sensor nodes whose communicating capability is only local.

Similar to NRA, the enabling assumption of KSCA is that DSNs are deployed in hostile environments where nodes are subject to node compromise but cannot afford expensive tamper-proof hardware. Consequently, the attacker can compromise a limited number of nodes and then manipulate their memories and program codes. The manipulated nodes referred to as attacker-controlled nodes are then injected back into the networks under the attacker's control. Thereafter, attacker-controlled nodes start colluding with each other by swapping their cryptographic secrets and IDs. By reusing these swapped secrets and IDs, a single node can present multiple legitimate identities and thus pretend to be different nodes to different neighbours. This means, using attacker-controlled nodes, the attacker can establish sufficient forged communication channels to outnumber legitimate ones. As pinpointed in [41], only a small fraction of network compromise, approximately 5%, can result in control over half of the valid communication channels. Therefore, a colluding minority can launch similar attacks as NRA does on, for example, routing, data aggregation, distributed voting, resource allocation, to name but a few.

3.4 Indirect Countermeasures

The indirect countermeasures refer to counteractive techniques presented in the literature to address different research problems. Limitations are exposed when these techniques are applied to counteract CLEA. The very first technique can be confining the usability of pre-distributed secret information to local areas using deployment knowledge and location information as presented in [34, 66, 85, 86] or location awareness as discussed in [65]. Because compromised secret information is only usable within a geographically specific location, when it is used by controlled nodes in different locations, they can not be used to establish illegitimate links. Thus, CLEA is made impossible. Unfortunately, the existence of deployment knowledge, location information, or location awareness is not always a justifiable assumption. For applications to which this assumption can not be applied, this technique becomes unusable.

The second technique can be key infection [55] which excludes the key pre-distribution phase by means of weakening the attack model and key transmission in plain-text in the key establishment phase. At first glance, this technique seems to be able to defend against CLEA but in fact it does not due to two reasons. First, plain-text key transmission enables the powerful on-site attacker to learn keys set up between two nodes with ease. Second, the key infection schemes provide no access control mechanisms for preventing controlled nodes from joining the network. Any controlled node can create links with its neighbours by simply generating a key and broadcasting it.

The third technique [52] can be for nodes to discard unused keys after the initialisation phase, but this means once initialisation is complete, the network can no longer accommodate new nodes. LEAP+ [53] exhibits the ability to tackle both the problems of network scalability and CLEA as follows. It is assumed that there exist a lower bound on the time interval $T_{\min}$ that is needed for the attacker to compromise a node, and the time T_{est} for a newly deployed node to discover its immediate neighbours such that $T_{\min} > T_{est}$. Each new node deployed during time interval T_i is pre-loaded with a network-wide initial key K_{IN}^{i}. The node uses this key to derive pairwise keys with its

neighbours within $T_{\min}$. After $T_{\min}$, K_{IN}^{i} is removed to prevent the attacker from compromising all the established pairwise keys and establishing illegitimate links within T_i. Furthermore, the node is also loaded with master keys derived from the node's ID and initial keys of future intervals. These master keys are used to establish pairwise keys with new nodes deployed in the future intervals. However, if the attacker is powerful enough to compromise a node within T_{est} of each interval, it can launch CLEA with ease.

The fourth technique can be applied to a network in which sensor nodes are deployed with a uniform density. In this case, nodes can detect whether they might be under CLEA by monitoring how many established links they have. If a node finds that the number of the established links is greater than a threshold value, it can consider itself to be attacked by CLEA. Nonetheless, it can be very difficult to determine if any of its neighbours are lying.

As discussed earlier, CLEA results from the node capture attack. Hence, CLEA can be foiled provided that the node capture attack is detected and prevented [46]. A defence against the node capture attack can be devised based on the observation that the continuous absence of a sensor node from the deployment area, which is usually assumed to be a prerequisite for the attack, can be detected by its neighbours using, for example, heartbeat messages or topology change notifications. The neighbours then consider the absent node a captured node and take network-wide action to revoke authorisation tokens of the captured node or initiate recovery when this node is inserted back into the network. Nevertheless, heavily relying upon the node absence for the detection of the node capture attack may introduce other issues. For example, a multitude of sensor network applications make use of energy-efficient MAC protocols such as S-MAC [87]. According to these protocols, the node goes to sleep for some time, and then wakes up and listens to see if any other node wants to talk to it. However, a legitimate node may be revoked from the network by its neighbours if the node's sleeping time is long enough. As another example, the attacker can mislead the node's neighbours about the absence of the node using jamming techniques [88, 89]. In other words, the node is not actually under the node capture attack but its neighbours believe that it has been captured since the regular communication between the node and its

neighbours is disrupted by jamming attacks. As a consequence, the node would be expelled from the network.

The node capture activity can also be noticed by the targeted node itself using an accelerometer, for example. The node might react to the suspected physical attack by automatically triggering a self-destroying process or an erasure process to all confidential materials stored on it. However, the accelerometer can be activated by some other natural phenomena other than the node capture. The examples of these phenomena include a wind, a collision between the sensor node and animals, and terrain vibration induced by heavy animals' movements, and military vehicles, to name but a few. This may lead to the needless loss of the node. Furthermore, the attacker can massively kill sensor nodes equipped with memory erasing or self-destroying technologies by imitating that he/she is physically attacking the nodes.

Another defensive technique against CLEA involves precluding the attacker from misusing the programming interfaces such as USB, serial, or JTAG (Joint Test Action Group) to read/write data from/to the captured node. To this end, the interfaces have to be removed, disabled, or protected. Unfortunately, as discussed in [46], the attacker can still obtain access to the programming ports by directly connecting to the appropriate pins on the micro-controller which can be looked up in the datasheet.

Obviously, CLEA is forestalled if the secret information within sensor nodes is protected by a privacy technique. Inspired by this observation, Alarifi and Du [90] developed a scheme which diversifies data and code segments by creating different and obfuscated data and code segments for each node. The aim of this scheme is to ensure that the attacker's method used to retrieve secret information from one node cannot be reused by another node. The first step of the scheme scrambles the data structure for storing secret information using a set of hash functions. The hash functions used for one node must be different from those used for the other nodes to make the reverse engineering attack more difficult. The second step deals with code obfuscation to hide the hash functions. In the final step, the control flow of code in one node is randomised to make it different from that in another node. After this step, different versions of the same code are created for different nodes. However, just making it difficult to tamper with the program code is not sufficient to protect against a determined attacker. As

discussed by the authors, sooner or later, by reverse engineering the obfuscated code, the attacker can eventually retrieve the secrets from a captured node. As a result, the scheme fails in combating CLEA.

The idea of employing hardware tamper resistance technologies has been repeated in a plethora of publications pertaining to sensor networks, as an alternative cure for the node compromise attack, and thus CLEA. The goal of these technologies is to provide any security-sensitive program with protection from unauthorised disclosure of secrets or inner working details [91]. However, this hardware-based protection will likely fail to provide acceptable security and efficiency due to two reasons. First, strong tamper resistance hardware is prohibitively expensive to implement in massively deployed, disposable and resource-constrained sensor devices. Second, constantly relentless and rapid improvements in technology have extended the reach and capability of the attacker. As a consequence, the tamper-resistant hardware itself is not always absolutely safe. Recent advances in physical attack show that even memory chips with built-in tamper-resistance mechanisms are subject to various memory read-out attacks such as reverse-engineering on chips, microprobing, glitch and power analysis, cipher instruction search attacks, chip rewriting attacks, and memory remanence attacks [92-98].

3.5 Direct Countermeasures

The term direct countermeasures refer to counteractive techniques developed in the literature to defend DSNs against NRA head-on. Although they are developed especially for NRA, they possess features against CLEA as well. Nonetheless, none of them can detect NRA in a real-time manner. In other words, there is a delay between the time replicas join the network and the time the detection process is executed. In the following, the mention of NRA (replica) can be regarded as the mention of CLEA (controlled node).

3.5.1 Witness-Based Detection Schemes

In their pioneering work, Parno et al. [9] presented a wide range of detection protocols against NRA including *centralised detection*, *local detection*, *node-to-network broadcasting*, *deterministic multicast*, *randomised multicast*, and *line-selected multicast*. In the centralised detection scheme, each node is required to send a list of its neighbours and their claimed locations to the base station. The base station then scrutinise every list to identify replicas. Thereafter, it can revoke the detected replicas by flooding the network with authenticated revocation messages. However, several drawbacks render this approach impractical. First, the use of the base station for security purpose apparently introduces a single point of failure. The scheme becomes useless if the attacker can compromise the base station or communication channels around it. Second, the nodes nearest to the base station will suffer from a heavy routing load and will become appealing targets for the attacker. The scheme also introduces delays in replica revocation since it takes the base station a long time to identify conflicts in the lists before flooding revocations throughout the networks. Finally, many DSNs are not supported by powerful base stations, thus a distributed approach is unavoidable.

The centralised detection approach can be replaced by the local detection scheme which depends on a node's neighbours to perform the replication detection. The neighbours use a voting mechanism to reach a consensus on the legitimacy of the node. Unluckily, although this scheme achieves detection in a distributed manner, it fails to detect distributed node replication in disjoint neighbourhoods within the network where replicas are at least two hops away from each other.

The node-to-network broadcasting scheme was the very first attempt to detect NRA in a distributed fashion. Accordingly, each node is required to flood the network with its location information in an authenticated manner. When receiving the flooded information, a node compares this information with the location information of its neighbours in its memory. If it detects a conflicting claim, it revokes the detected replicas. While this scheme guarantees 100% detection of duplicate location claims providing the broadcast messages reach every node, it puts the total communication cost

of $O(n^2)$ messages on the network. This cost is not a cause for concern for small networks. But as the size of the network grows, it becomes too costly.

The deterministic multicast scheme is developed to reduce the communication overhead of the previous scheme in such a way that a node's location claim is not sent to all the other nodes in the network, but just to a limited set of nodes, called witnesses. The witnesses are chosen as a function of the node's ID. If a replica exists somewhere in the network, then the witnesses will receive two conflicting location claims for the same node. Assume that the size of the set of witnesses is g and the degree of each node d, then as long as each of the neighbours randomly selects $\frac{g \ln g}{d}$ witnesses from the set, the coupon collector's problem [99] assures that each of the witnesses will receive at least one location claim. Assuming an average network path length of $O(\sqrt{n})$ nodes, the resulting communication cost is $O(\frac{g \ln g \sqrt{n}}{d})$ messages. Unluckily, this decline in the communication cost is achieved at cost to security. Since the function used to select the witnesses is deterministic, the attacker can predict them as well. Therefore, if the attacker is able to capture or jam all of the messages bound for the witnesses, then it can create as many replicas as it wants.

The security weakness in the deterministic multicast scheme can be overcome by randomising the witnesses as proposed in the randomised multicast (RM) scheme. This prevents the attacker from anticipating their identities. Specifically, the scheme requires each node to broadcast its signature-based authenticated location claim. Each of the node's neighbours, with probability p, selects g random locations within the network and uses a routing protocol (e.g., geographic routing [100]) to forward the claim to the nodes closest to the selected locations. The security analysis shows that the probability of choosing the same node as a witness more than once is negligible. In the network of n nodes where each location is testified by $\sqrt{n}$ witnesses, the birthday paradox [99] anticipates that a pair of conflicting location claims will be encountered under NRA with high probability. For instance, if $n = 10,000$, $g = 100$, $d = 20$, and $p = 0.05$, the probability of detection will be greater than 63% for a single replication and 95% for double replication. However, this scheme remains inefficient in terms of storage and

communication requirements. For example, with the above network setting, each node is required, on average, to store 3,700 bytes, which is just less than 91% of the MICA2/MICAz's total RAM. Similarly, as estimated in [9], the total communication cost is $O(n^2)$, equivalent to those of the node-to-network broadcasting scheme.

Due to the above limitations, a different scheme named line-selected multicast (LSM) was investigated to improve the performance of the randomised multicast scheme. In this scheme, each node is required to send out its location claim to r initial witnesses. The function of witnessing is then extended to each of the nodes along the routes from the broadcasting node to the initial witnesses as well. The node en route also stores a copy of the location claim while forwarding the claim to an initial witness. By storing location claims at intermediate nodes, we have effectively drawn line-segments through the network as illustrated in Figure 3.3. Two line segments intersecting correspond to the fact that a witness at the intersection receives two conflicting location claims.

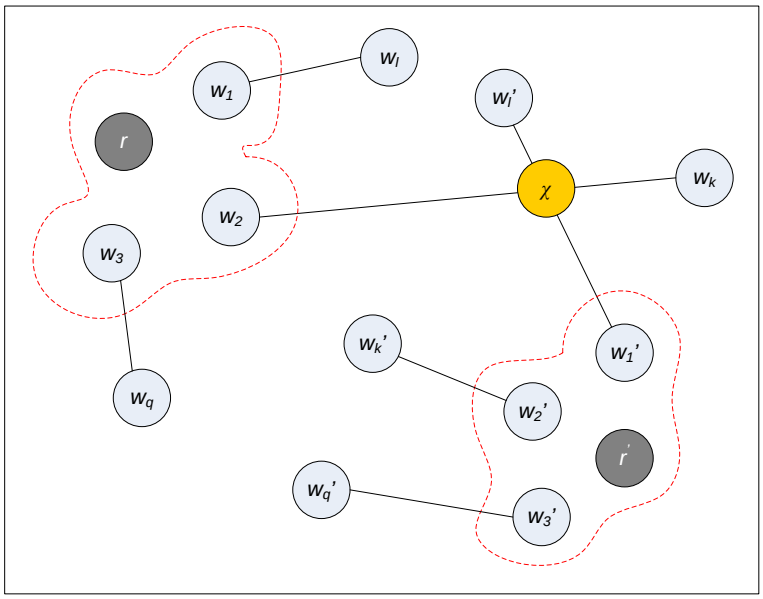


Figure 3.3: Line-selected multicast: The attacker is assumed to have created a replica of r, r'. The storage of the replicas' location claims at nodes en route to the witnesses (w_is and w_i's) results in an intersection at χ

Thereafter, it can flood the network with the undeniable evidence to revoke the detected replicas. This scheme offers a very high detection rate. As pinpointed in [9], the

detection rate is 56% with one line segment per node and soars to 95% with five line segments per node. Furthermore, LSM scheme also exhibits reasonable performance feature. Assume that the length of each line segment is $O(\sqrt{n})$, then the scheme only requires $O(n\sqrt{n})$ messages for the whole network and each node stores $O(\sqrt{n})$ location claims.

Table 3.2 summarises the storage and communication costs for each of the distributed protocols discussed previously. The communication costs are for the whole network while the storage costs are per node.

In their work [77], Conti et al. pointed out the security weakness of LSM by introducing a stronger attacker, named smart attacker. This attacker aims to anticipate and corrupt prospective witnesses before each round of the detection scheme. If the attacker has successfully corrupted these nodes, the scheme will fail to detect NRA. The anticipation can be carried out based on the nodes' ID information and nodes' location information.

Table 3.2: Summary of scheme costs

	Communication	**Memory**
Node-to-network Broadcast	$O(n^2)$	$O(d)$
Deterministic Multicast	$O\left(\frac{g \ln g\sqrt{n}}{d}\right)$	$O(g)$
Randomized Multicast	$O(n^2)$	$O(\sqrt{n})$
Line-Selected Multicast	$O(n\sqrt{n})$	$O(\sqrt{n})$

To defend against this attacker, the authors proposed a Randomised, Efficient, and Distributed (RED) protocol. RED tries to provide *ID oblivious* and *area oblivious* features by continually changing a node's witness set after each RED iteration to avoid the anticipation. Specifically, RED executes at fixed time intervals. Each round of the protocol consists of two steps. First, a random value, *rand*, is distributed among all the

nodes. Second, each node digitally signs and broadcasts its claim including node ID and geographic location. Each of the node's neighbours, with probability p , sends the claim to a set of network locations. These locations are selected using the PseudoRand function. The inputs of this function consist of the node's ID, the current rand value and the output is the selected network locations. The function also guarantees that the neighbours, who are forwarding the claim, will select the same set of the network locations. The witnesses are the closest nodes to the selected locations. However, RED does not offer any better resiliency and performance than the original (LSM). The *ID oblivious* and *area oblivious* features are not ensured since anyone who has the knowledge of *rand* through either node compromise or eavesdropping a broadcast message containing the value of *rand* at the beginning of each round will most likely discover the prospective witness set. RED also incurs significant communication, computation, and storage overheads which make it unappealing. Moreover, similar to LSM, it encounters difficulties in accommodating new nodes to join the network.

Ho et al. [101] proposed a distributed detection protocol of NRA using group deployment knowledge. In this protocol, nodes are expected to be in their home zone and if it is the case they are marked by their neighbours as trusted. Nodes placed outside their home zones have to prove their legitimacy by requesting their neighbours to forward their location claims to their home zones for conflicting location claim detection. Because the assumption of the deployment knowledge is not often reasonable and general, the heavy dependence on it makes the protocol undesirable.

The latest effort in the witness-based series to defend against CLEA was Localised Multicast (LM) developed by Zhu et al. [102]. In this approach, the network is assumed to be a geographic grid consisting of cells. For each detection round, each sensor node broadcasts its location claim. Each neighbour of the node first verifies the plausibility and validity of the claim. It then independently decides whether to forward the claim to a single destination cell (as for Single Deterministic Cell scheme) or multiple deterministic cells (as for Parallel Multiple Probabilistic Cells scheme) with a certain probability. The destination cell(s) is(are) selected using a geographic hash function [103] of the claim broadcaster's ID. Once the location claim arrives at the destination cell, the first node receiving the claim verifies the claim and then checks if the claim is in fact destined for this cell. If the claim passes both of the verifications, it is flooded

throughout the cell. Each node or witness in the cell independently decides with a certain probability whether to store the claim. Whenever a witness observes two different location claims with the same identity, it triggers the replica revocation process using the two conflicting claims as the evidence.

Nonetheless, the LM approach exposes several undesired limitations. First, similar to [101], it requires the knowledge of deployment and network topology. Second, multiple copies of the forwarded location claim arrive at a destination cell from different routes. It is not straightforward how to determine which copy is the first one to the cell. This leads to the flooding of one location claim multiple times in the cell and thus the waste of network resources. Lastly, since the geographic hash function and the IDs of claim senders are known, the attacker can also determine the targeted cells and thus the witnesses for subversion.

3.5.2 SET: Set Operation Based Detection Scheme

The witness-based schemes described above must rely on public key cryptography which may be undesirable for low-end DSNs. To overcome this limitation, Choi et al. [78] proposed a new scheme, named SET, to detect NRA. The key idea is based on computing set operations (intersection and union) of exclusive subsets in a DSN to detect replicas. Correspondingly, the DSN can be perceived as a set of non-overlapping subregions. Nodes in each subregion form an exclusive subset. Because node IDs are unique throughout the network, the intersection of any two subsets should be empty. If the attacker deploys replicas in the network, the intersection of subsets including these replicas will not be empty, and thus NRA can be detected.

SET consists of five components: *exclusive subset construction*, *authenticated subset covering*, *verifiable random member selection*, *distributed set computation on subset trees*, and *interleaved authentication on subset trees*. The first component uses an exclusive subset maximal independent set (ESMIS) algorithm to form exclusive subsets in a distributed fashion. An exclusive subset consists of one subset leader (SLDR) and a number of subset members within the SLDR's transmission range. The process of forming the exclusive subset in ESMIS happens as follows. The base station generates a

random *seed* and broadcasts it to the network. Upon receiving the seed, every node puts itself in *Init* state. Each node then computes locally hash values for itself and its neighbours using a hash function $H_1: (seed \mid x) \rightarrow y \in [1, d]$, where d is the average number of neighbours in the network and x is the node's ID or the ID of one of the node's neighbours. The node corresponding to the maximum hash value in its neighbourhood becomes a SLDR and switches its state to *Ruler*. If there is more than one node with the same maximum hash values, the node having the biggest ID will be selected as the SLDR. This SLDR then informs its neighbours of its promotion. Upon receiving the SLDR's message, the other neighbours in the Init state change their state to *Ruled*. This means that they become the members of the exclusive subset dominated by the SLDR. Thereafter, the nodes in the ruled state announce their SLDR to their neighbours.

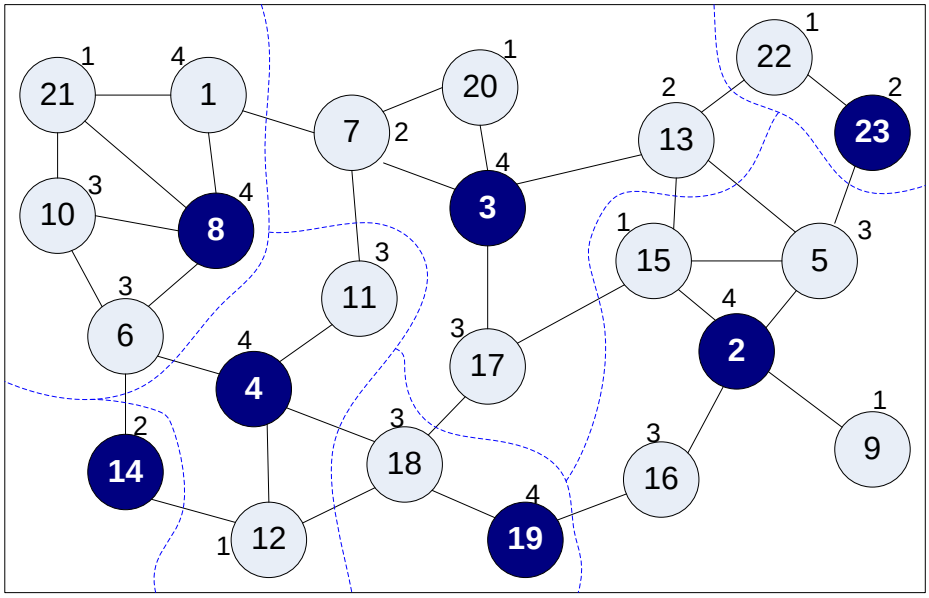


Figure 3.4: The construction of seven subsets in accordance with the ESMIS algorithm

Figure 3.4 illustrates an example of a small network with 23 nodes and $d = 4$. The value outside the circle of each node is the H_1 computation result. The construction of seven subsets in accordance with the ESMIS algorithm is demonstrated. Nodes 2, 3, 4, 8, 14, 19, and 23 are selected as SLDR nodes of the neighbourhoods bounded by dashed curves.

The second component of SET, authenticated subset covering, is used to cope with the presence of corrupted nodes in hostile environments. These corrupted nodes may attempt to hide their existence by either not following the ESMIS algorithm or announcing their ruled state with bogus SLDR identifiers. To address this problem, the component requires each ruled node to send the identifiers of all its neighbouring SLDR to its SLDR. This SLDR then generates membership authentication of the ruled node for each neighbouring SLDR. The membership authentication is a message authentication code (MAC) derived from the ruled node ID and the SLDR ID using a MAC function and the pairwise key between the SLDR and the neighbouring SLDR. This membership MAC is then sent to the neighbouring SLDR and used by this node to vouch for the SLDR's covering of the ruled node. Because the ruled node does not have the pairwise key between the SLDR and the neighbouring SLDR, it cannot generate the membership MAC. Hence, the ruled node can not convince its neighbouring SLDRs of a bogus SLDR ID.

According to the basic idea, every node ID is reported to the base station through SLDRs. This can introduce significant communication overhead to nodes near the base station. The component of verifiable random member selection is used to reduce the communication cost. The key idea is to allow the base station to randomly select which nodes should be reported, instead of all the nodes. The base station releases addition information which is buried in a string of bits "0" and "1" when broadcasting the seed. A SLDR uses this string to select the reported members and then send them to the base station for the attack detection.

Due to the random selection of SLDRs, the base station might not know which nodes are SLDRs. As a consequence, the attacker can drop reports sent by SLDRs without detection. To tackle this issue, a multiple tree based approach is employed in the network. A tree consists of nodes which are the SLDRs of the exclusive subsets. The tree construction starts from selecting a SLDR root randomly. Thereafter, the root discovers the neighbouring SLDRs and admits them to be its children. The process continues until the last SLDR joins the tree. After the tree construction, the leaf SLDR sends its subset report to its parent. The parent collects its children's subsets and computes the intersection of these subsets with its own subset to detect NRA. If NRA is

not detected, the parent generates a union of its children's subsets and its own subset, and sends this new report to its parent. Each root forwards its final report to the base station. If a node on the tree detects that the computed intersection is not empty, it knows that replicas exist in its subtree. It then notifies the base station of NRA for further action.

The tree-based approach works well in the absence of corrupted nodes. However, if a parent is corrupted, it may delete replicated identifiers from the computed union to conceal NRA. Therefore, the set operations (intersection and union of subsets) on the tree should be verified. This verification can be enabled by employing the idea in [104] to develop an interleaved authentication scheme on the tree. This scheme requires that the SLDRs store the path information from the root to themselves during the tree construction. When a SLDR sends a report to its parent, it computes a keyed MAC, e.g. HMAC [105], not only for its parent but also for its grandparent (interleaved MAC). The grandparent can detect any changes made by corrupted parents on the results of set operations by computing and checking the interleaved MACs.

The drawback of SET is that it has to resort to three costly methods: authenticated subset covering, distributed set computation on subset trees, and interleaved authentication on a subset tree which substantially increase the performance overheads and complicates the detection procedure. Furthermore, SET requires the involvement of the base station which introduces the single point of failure problem. Thus SET becomes an undesirable solution.

3.5.3 Bloom Filter Based Detection Scheme

Approaching NRA in a different manner, specifically for random key pre-distribution schemes [27-30], Brooks et al. [79] introduced the concepts of a Bloom filter [106] and a counting Bloom filter [107] to monitor the use of keys as authentication tokens and thus detect statistical deviations that indicate NRA.

The Bloom filter is a space-efficient probabilistic data structure that is used to test if an element is a member of a set via membership queries. Elements can be added to the set,

but not removed. Initially, an empty Bloom filter, which is a bit array of m bits, all set to 0, is created. There must also be k different defined hash functions, each of which maps or hashes a set element to one of the m array positions with a uniform random distribution. The set element is added to the filter by feeding it to each of the k hash functions to get k array positions. As shown in Figure 3.5, bits at all these positions are then set to 1. The membership query about the set element is performed by feeding it to each of the k hash functions to get k array positions. If any of the bits at these positions are 0, the element is not in the set. Otherwise all the bits would have been set to 1 when it was added. If all are 1, then either the element is in the set, or the bits have been set to 1 during the addition of other elements.

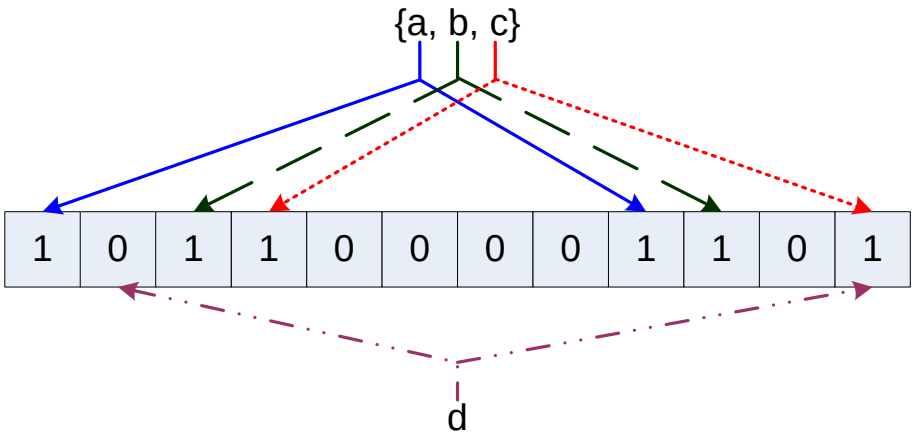


Figure 3.5: An illustration of a Bloom filter, representing the set $\{a, b, c\}$. The arrows show the positions in the bit array to which each set element is mapped. The element d is not in the set $\{a, b, c\}$, because it hashes to one bit-array position containing 0 ($m = 12$, $k = 3$)

Unfortunately, the removal of an element from the Bloom filter is impossible. The element is represented by k bits in the filter and can be removed by setting any one of these k bits to zero. However, this setting may lead to the removal of other elements that are also represented by that bit. A counting filter provides a solution to support a *delete* operation on a Bloom filter. In the counting filter, the array positions (buckets) are extended from being a single bit to being an n-bit counter. The add operation is extended to the increment of the value of the buckets and the look-up operation checks

if each of the required buckets is non-zero. The delete operation essentially consists of decrementing the value of each of the respective buckets.

The main idea of the counting filter-based protocol is as follows. Since each node is pre-loaded with m randomly selected keys from the pool, the distribution of the number of nodes possessing a given key can be predictable. The distribution of the number of times a key is used for the PKE is directly proportional to the number of nodes owning that key. When NRA is launched, the key usage distribution is skewed. The reason is that replicated keys appear on a greater number of nodes than normal, and thus are used more frequently than keys that have not been replicated. By gathering key usage statistics, the keys which have been replicated can be specified and revoked.

According to the protocol, the gathering of key usage statistics is conducted as follows. Each node constructs a counting filter from the keys it uses to connect to its neighbouring nodes. It appends a random number (nonce) to the created filter and encrypts the result using the base station's public key. This encrypted report is forwarded to the base station. The base station decrypts the received filters and discards duplicate reports by checking nonces. The base station performs membership queries on the filter and counts the number of times each key is used in the network. Keys used above a threshold value are considered replicated. The base station creates a Bloom filter from the replicated keys, signs the filter with its private key and broadcasts this filter to the network using a gossip protocol [108]. Each node verifies the received filter and checks the keys in its key ring for membership of replicated keys in the filter. The node then removes the replicated keys from its key ring and terminates all connections using the replicated keys.

The drawback of this protocol is the high false negative and positive rates. Moreover, it does not handle the situation when replicas report their key usage deceitfully.

3.5.4 Randomly Directed Exploration Based Scheme

Li and Gong [109] proposed an enhanced version of node-to-node broadcasting scheme [9] to tackle the node clone attack in a dense sensor network. During one detection round, each node generates c claiming messages which contain its list of neighbours.

The node then sends each of these messages to a randomly selected neighbour. The message is forwarded in such a manner that the route it travels is roughly as a direct line (randomly directed exploration). The forwarding process stops when either the time-to-live field (*ttl*) in the message comes to zero or the message reaches the network border. Each node en route compares its own neighbour list with the neighbour list in the received message to detect replicas. If replicas are detected, that node invokes the replica revocation process network-wide to expel the detected replicas from the network.

Several limitations render this scheme undesirable. First, the randomly directed exploration technique helps the attacker locate the message forwarders en route with ease. Once the attacker knows the locations of these forwarders, the attacker can subvert them to stop the detection process. Second, the attacker can alter the value of *ttl* to reduce the detection rate (decrease of *ttl*) or increase communication cost (decrease of *ttl*). Finally, the protocol does not require intermediate nodes to buffer claiming messages. Therefore the detection rate is not high since claiming messages may not traverse nodes containing relevant replicas in their neighbour lists.

3.5.5 Sequential Analysis Based Detection Scheme

Based on the observation that existing detection schemes only work in fixed sensor networks, Ho et al. [110] proposed a detection scheme for mobile sensor networks where replicas are also mobile. The enabling assumption of this scheme is that a benign mobile node should never move faster than the system-configured maximum speed. In contrast, because replicas need to be at two (or more) different locations at once, they will be likely to move faster than benign nodes which results in their measured speeds to be over the configured maximum speed. The scheme makes use of the sequential probability ratio test (SPRT) proposed in [111] as follows. Each time a mobile node moves to a new location, each of its neighbours asks for an authenticated location claim from the node which contains its location and time information. The neighbours then probabilistically forward the valid claim with a certain probability to the base station. The base station computes the node's speed from every two consecutive claims and performs the SPRT by considering speed as an observed sample. Each time the mobile node's speed exceeds the configured maximum speed, it will speed up the random walk

to hit or cross the upper limit. This results in the base station's acceptance of the alternate hypothesis that the mobile node has been replicated. When the mobile node's speed remains below the configured maximum speed, it will speed up the random walk to hit or cross the lower limit. This results in the base station's acceptance of the null hypothesis that the mobile node has not been replicated. Once the base station decides that the mobile node has been replicated, it revokes the replica nodes from the network.

As discussed in Chapter 2, this thesis only focuses on CLEA and its countermeasures in stationary distributed sensor networks. This scheme is beyond the scope of this thesis and is only included here for the sake of completeness. In addition, it exhibits two limitations. First, the attacker can drop the claims to minimise the detection rate. Second, the scheme introduces single point of failure by employing the base station in the detection process.

3.5.6 Random-Walk Based Detection Approach

Emphasising the vulnerability of witnesses to smart attacks in the previous witness-based approaches, Zeng et al. [112] proposed two new non-deterministic and fully distributed protocols: RAndom WaLk (RAWL) and Table-assisted RAndom WaLk (TRAWL). They are the modified version of RM [9]. Similar to RM, each node S_i's location claim is first forwarded to several randomly selected nodes. These nodes then becomes the first witnesses of S_i. From this point RAWL and TRAWL are different from RM. As illustrated in Figure 3.6, each of the initial witness starts a w-step (w: system parameter) random walk in the network by sending the location claim together with a counter of walking steps (s_c) initialised to 1 to a random neighbour. This neighbour, which is also a witness, increases s_c by 1 and continues to forward the message to another random neighbour (witness) until s_c reaches w. When conflicting location claims collide at a witness, the witness will flood the network with the conflicting claims to revoke replicas.

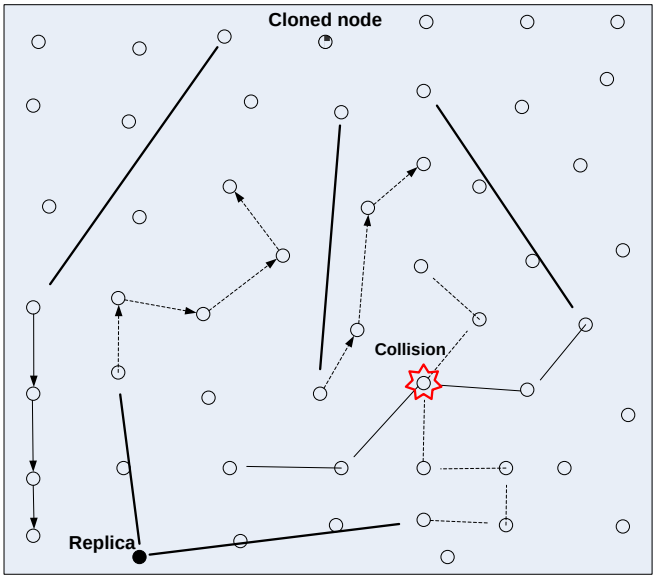


Figure 3.6: Illustration of random-walk based detection approach

TRAWL improves RAWL by reducing memory cost of RAWL while its main idea remains identical to RAWL. This reduction is obtained by requiring only a certain percentage of witnesses on the random walk, instead of all of them, to store the location claim. In addition, to maintain the same detection capability as RAWL, each witness also builds a trace table for recording the pass of the location claim. Each entry recording the pass consists of two fields: NodeID and ClaimDigest. NodeID is the ID of the claim sender. ClaimDigest is a truncated MAC of the entire location claim: $claimDigest = \{MAC_{rand}(Claim)\}_{\mathrm{mod}(256)}$ where $rand$ is a random value generated by each witness.

A witness detects NRA when it comes across two different digests of the same node ID. If two conflicting location claims corresponding to the digests are available, the witness floods the network with them to revoke replicas. Otherwise it will flood a HELPREV request with the received location claim. Any node receiving the HELPREV message will check locally whether there is a collision between the received claim and stored one. If such a collision is found, it will flood the stored claim into the network as evidence for the revocation purpose.

The authors claimed that RAWL and TRAWL boost the security of witnesses against smart attackers while their costs remain acceptable. However, their arguments for this boost is not convincing because RAWL and TRAWL introduce other security threats. First, they are susceptible to a DoS attack in which the attacker claims herself as a chosen node (witness) to send random walk messages. Since these messages involve legitimate nodes in the undesirable detection process, the attack indeed wastes significant network resources. Second, since the value of s_c can be modified en route, the attacker can either increase s_c to w to decrease the number of witness nodes (or detection ability) or decrease (reset) s_c to waste witnesses' resource. Third, TRAWL facilitates another DoS attack on network resources since HELPREV message forwarders have no evidence that the received message is from legitimate nodes or compromised nodes. Thus, the compromised nodes can create and flood the network with a large number of faked HELPREV messages containing legitimate location claims with the purpose of wasting network resources. Lastly, similar to RM and LSM, RAWL and TRAWL still fails to detect NRA in the presence of the masked-replication attacks [9].

3.6 Chapter Remark

In this chapter, we have provided the taxonomy of key establishment schemes and shown that they are all more or less vulnerable to CLEA. We described two instances of CLEA existing in current literature: NRA and KSCA in order to provide a deeper understanding of CLEA. Finally, we reviewed existing countermeasures against CLEA and specified their limitations. Based on these discovered limitations, we conclude that none of the existing countermeasures are a compelling solution to CLEA and thus new approaches are essential to secure operations of DSNs.

Chapter 4

Background

This chapter discusses primary building blocks as preparation for the development and discussion of proposed countermeasures given in Chapters 6 and 7. These building blocks are cryptographic primitives, message authentication schemes, denial-of-service (DoS) prevention for broadcast authentication, anti-jamming techniques, secure localisation algorithms, and secure time synchronisation protocols. In general, the discussion of each building block starts with the introduction of the block followed by its importance to sensor networks and finally up-to-date outcomes. The last section of the chapter is used to enumerate important symbols mentioned through out the dissertation.

4.1 Cryptographic Primitives

A myriad of security-critical applications such as battlefield surveillance and homeland security monitoring have placed very high demands upon DSNs for security measures such as key management schemes, attack detection and prevention mechanisms, secure routing and localisation schemes, and secure data aggregation schemes to protect them against security menaces like network infiltration, data manipulation, and denial of service. To make such measures possible, security primitives such as encryption and authentication need to be enabled in the first place. This can be obtained using either public-key cryptography (PKC) or symmetric-key cryptography (SKC). In DSNs, PKC such as RSA [127] and ECC [128, 129] is much less preferable than SKC due to several reasons. Firstly, although PKC, especially ECC, has recently been demonstrated to be feasible for deployment in DSNs, it is still much more expensive than its symmetric-key counterparts. For instance, as shown in [130] the energy costs of 160-bit point

multiplication (main operation in any ECC-based algorithm), digital signature generation and verification are 55 *m*J, 52 *m*J, and 63 *m*J respectively. Meanwhile, the encryption of a 128-bit data block using a symmetric cipher like advanced encryption standard (AES) requires only 38 μJ on the same MICAz platform [131]. Secondly, public key management plays a very important role in deploying PKC. However, thus far the problem of how to manage public keys involving public key authentication and revocation in DSNs has not been addressed properly yet. The difficulty lies in the fact that conventional approaches to this problem introduce both the battery-exhaustion attack (induced by expensive public key authentication operations) on sensor nodes and immense communication overhead (induced by public key revocation operations) in the network. Thirdly, in the context of public-key security protocols, usually a symmetric session key is derived and used for encryption and decryption to minimise the computational overhead. However, this minimisation comes at the cost of implementing and loading symmetric-key algorithms into sensor nodes. This dual implementation results in bulky source code and high memory consumption. Therefore, this section focuses on giving a brief overview of SKC primitives including symmetric key ciphers and cryptographic hash functions.

4.1.1 Symmetric Key Ciphers

In general, there are two kinds of symmetric key ciphers: stream ciphers and block ciphers [60, 184]. The most widely used stream ciphers are RC4 [185] and SEAL [186]. To encrypt plaintext, a stream cipher first generates a pseudorandom cipher bit stream (keystream). The encryption is carried out by combining the keystream with the plaintext using the bitwise XOR operation in such a manner that plaintext digits are encrypted one at a time, and the transformation of successive digits varies during the encryption. The stream cipher can be synchronous in which the generation of the keystream is independent of the plaintext and ciphertext or self-synchronising where the keystream generation depends on the data and its encryption. Stream ciphers generally represent a different approach to the encryption and decryption of data from block ciphers. However, this difference does not always exist because any block cipher can be used as a stream cipher when used in certain modes of operation, e.g. data encryption standard (DES) in cipher feedback (CFB) or output feedback (OFB) modes. The

advantages of stream ciphers include faster processing speed than block ciphers in hardware and lower complexity of hardware circuitry. However, stream ciphers are vulnerable to some serious attacks such as correlation attacks [187], linear consistency test [188], linear syndrome algorithm [189], and linear cryptanalysis [190] if used improperly.

According to [60], block ciphers are primarily used to provide confidentiality. However, they can be used for the construction of pseudorandom number generators, stream ciphers, MACs, and hash functions. They may further serve as a key component in message authentication techniques, data integrity mechanisms, entity authentication protocols, and (symmetric-key) digital signature schemes. Skipjack [132], RC5 [133], DES [134], and AES [135] are prime examples of the block ciphers. As illustrated in Figure 4.1, block ciphers transforms a fixed-length block of *plaintext* (unencrypted text) data into a block of *ciphertext* (encrypted text) data of the same length. This transformation takes place with the aid of a user-provided key. The plaintext is recovered from the ciphertext by applying the reverse transformation to the ciphertext block using the same secret key. The fixed length is called the block size which is often 64 or 128 bits. When a block cipher is used to encrypt a message of arbitrary length, techniques known as modes of operation for the block cipher are utilised. The standard modes are electronic code book (ECB), cipher block chaining (CBC), CFB, and OFB. To be useful, a mode must be at least as secure and efficient as the underlying cipher.

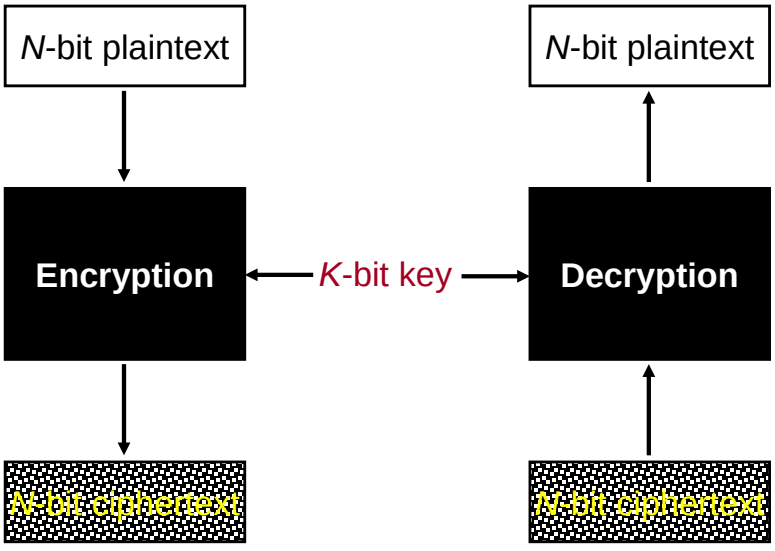


Figure 4.1: The concept of a symmetric-key block cipher

In addition to the terms mentioned in the description of block ciphers above, there are several common terms mentioned frequently when block ciphers are discussed as follows.

- S-box: A component of a block cipher which is used to substitute a small block of bits (the input of the S-box) by another block of bits (the output of the S-box). An S-box may or may not be invertible. In an invertible S-box, the number of input bits should be the same as the number of output bits.
- P-box: A component of a block cipher which takes the outputs of all the S-boxes of one round, permutes the bits, and feeds them into the S-boxes of the next round.
- Diffusion and confusion: The idea of diffusion and confusion in a block cipher is to hide the relationship between the ciphertext and the plaintext. This is done by guaranteeing that the change of one bit of the plaintext or the key will result in the complete change of the ciphertext in a pseudorandom manner.
- Round: Iteration of an iterated block cipher is termed a round.
- Substitution cipher: A method of encryption by which units of plaintext are replaced with those of ciphertext according to a regular system. The units may be bits or letters. The receiver deciphers the text by performing an inverse substitution.
- Transposition cipher: A method of encryption by which the positions held by units of plaintext are shifted according to a regular system, so that the ciphertext constitutes a permutation of the plaintext.
- Iterated block cipher: A block cipher where the encryption/decryption process consists of several rounds.
- Product block cipher: A block cipher which combines two or more transformations such as substitution, permutation, and modular arithmetic with the aim of making the resulting cipher more secure than the individual components.
- Subkey or round key: A key which is used in one round of an iterated block cipher and usually derived from the user-provided secret key by a special function.

- Key schedule: The set of subkeys in which one subkey is used for one round of an iterated block cipher.
- Round function: A function which is used by each round of an iterated production block cipher. It consists of multiple layers of transformations that perform substitution and permutation.
- Feistel cipher: A special class of iterated block ciphers where the ciphertext is calculated from the plaintext by the repeated application of the same transformation or round function.
- Non-Feistel cipher: A cipher which uses only invertible components. A component in the encryption cipher has the corresponding component in the decryption cipher.
- Substitution-permutation network (SPN): A series of linked mathematical operations used in block cipher algorithms such as DES and AES (Rijndael). Such a network takes a block of the plaintext and the key as inputs, and applies several alternating "rounds" or "layers" of S-boxes and P-boxes to produce the ciphertext block.

Table 4.1: Common elements in block ciphers [136]

Algorithm	No. of rounds	Block size (bits)	Feistel	Substitution-permutation Network	Arithmetic S-box	Pseudo-random S-box	Fixed permu-tations	Addition (mod 2w)	Fixed shift/ rotation	Variable shift/ rotation	Modular multipli-cation	Multipli-cation with a constant
DES/ Triple DES	6/48	64	Yes			Yes	Yes		Yes			
IDEA	8	64	Yes					Yes			Yes	
RC5	12/16	32/128	Yes					Yes		Yes		
AES (Rijndael)	10/12/14	128		Yes	Yes		Yes		Yes			Yes
RC6	20	128	Yes					Yes		Yes	Yes	
MARS	2 x 16	128	Yes			Yes		Yes	Yes	Yes	Yes	
Serpent	32	128		Yes		Yes	Yes		Yes			
Twofish	16	128	Yes			Yes		Yes	Yes		Yes	Yes
XTEA	More than 31	32	Yes					Yes	Yes			

As an example, Table 5.1 summarises the common features of the well-known block ciphers. The table indicates whether the block ciphers support certain features. For

example, DES, RC5, and MARS are variations of the Feistel cipher and support round structures while substitution and permutation networks are used in international data encryption (IDEA) and AES (Rijndael). The reader is referred to [seberry, internet security Man lee] for fascinating reading of the history, detailed explaination, and applications of the block ciphers.

4.1.2 Cryptographic Hash Functions

A cryptographic hash function (hereafter referred to as hash function) is a deterministic procedure that takes a message or bit-strings of arbitrary finite length and produces an output or bit-strings of fixed length. The output is usually referred to as digest, hash code, hash result, hash value, or simply hash. Hash functions are designed in such a manner that any accidental or intentional modification to the input will result in the change of the output.

Hash functions can be classified into two categories: unkeyed hash functions with a single input (a message) and keyed hash functions with two distinct inputs, a message and a secret key. The unkeyed hash functions can find their implementation in digital signature schemes or other applications which requires data integrity assurances. One specific class of the unkeyed hash functions is one-way hash functions (OWHFs) which are used in proposed schemes presented later in Chapter 6. According to [60], an OWHF H is a unkeyed hash function which satisfies the following conditions:

- Compression: H maps an input x of arbitrary length to an output $H(x)$ of the fixed length of n bits.
- Ease of computation: $H(x)$ is easy to compute given H and an input x.
- Preimage resistance: Given a y in the image of H, it is computationally infeasible to find a message x such that $H(x) = y$.
- Second preimage resistance: Given x and $H(x)$, it is computationally infeasible to find a message $x' \neq x$ such that $H(x') = H(x)$.

The well-known OWHFs include Message-Digest algorithm 4 (MD4), Message-Digest algorithm 5 (MD5) [62], Secure Hash Algorithm (SHA) family such as SHA-0, SHA-1 [63] and SHA-2.

Typical examples of the keyed hash functions are message authentication codes (MACs). This subclass of the keyed hash functions accepts as input a secret key and an arbitrary-length message to be authenticated, and outputs a MAC. According to [60], a MAC algorithm is a family of functions h_k parameterized by a secret key k, with the following properties:

- Ease of computation: For a known function h_k, given a value k and an input x, h_k is easy to compute. This result is called the *MAC-value* or MAC.
- Compression: h_k maps an input x of arbitrary finite bit-length to an output $h_k(x)$ of fixed bit-length n. Furthermore, given a description of the function family h, for every fixed allowable value of k (unknown to an adversary), the following property holds:
- Computation-resistance: Given zero or more text-MAC pairs $\left(x_i, h_k\left(x_i\right)\right)$, it is computationally infeasible to compute any text-MAC pair $\left(x, h_k\left(x\right)\right)$ for any new input $x \neq x_i$ (including possibly for $h_k\left(x\right) = h_k\left(x_i\right)$ for some i).

The MAC value assures both the source of a message and its integrity by allowing verifiers (who also possess the secret key) to detect any changes to the message content. The MAC algorithms are different from the OWHFs in the sense that given a message as input anyone may compute the hash values of the OWHFs but cannot compute MACs unless he/she knows the secret keys. MAC algorithms can be constructed from cryptographic hash functions such as HMAC-MD5 and HMAC-SHA1 [137] or from block cipher algorithms like One-key MAC (OMAC), Cipher Block Chaining Message Authentication Code (CBC-MAC), and Parallelizable MAC (PMAC).

4.2 Message Authentication in DSNs

This thesis considers two communicating models that need message authentication services: BS-to-node (base station to/from nodes) and node-to-node. Providing authentication for the latter is much more challenging than that for the former since the network topology is usually not known a priori and thus pre-distribution methods

cannot be applied. This section therefore discusses recent advances in message authentication for DSNs which cover both symmetric-key approaches and public-key approaches. The discussed approaches are selective and chosen based on their compatibility with the proposed schemes in Chapter 7.

4.2.1 Symmetric-Key Approaches

The basic symmetric-key approach to message authentication is to share a secret key among both senders and receivers. Unfortunately, this approach is not secure in sensor networks where receivers are not trusted due to node compromise. Since all nodes hold the key, any compromised node can be used by an attacker to forge or spoof messages from senders.

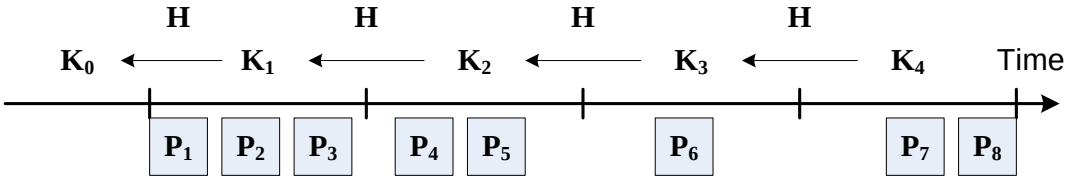


Figure 4.2: Illustration of μTESLA

In fact, message authentication based on symmetric-key approaches requires an asymmetric mechanism such that the receivers can only verify the authenticity of received messages, but not generate valid authenticated data. μTESLA [67] is the first effort to provide broadcast authentication for sensor networks. μTESLA uses the loose time synchronisation to obtain the asymmetry and a one-way hash chain of self-authenticating keys to compute MACs for outgoing messages. As illustrated in Figure 4.2, each key of the chain is associated with one time interval and used to compute MACs for packets sent over that interval. This key is disclosed after a certain number of time intervals. Upon receiving a packet, a receiver buffers this packet and verifies the packet when it receives the disclosed key corresponding to the packet later. Unluckily, μTESLA suffers from two major drawbacks which make it uncompelling. Firstly, μTESLA only allows authenticated broadcast/multicast from the base station but not from sensor nodes. Secondly, the protocol cannot provide immediate packet verification.

The receivers have to wait for a certain amount of time before being able to verify received packets.

Heer et al. [138] proposed ALPHA, an adaptive and lightweight protocol for hop-by-hop authentication. The design goal of ALPHA is to replace traditional shared secret based end-to-end integrity protection mechanisms, which exclude intermediate nodes from verification process, by enabling not only end-to-end but also hop-by-hop integrity protection of message in transit. This goal is achieved by making use of two one-way hash chains: signature chain and acknowledgment chain. Before communicating with each other, the signer and the verifier have to go through a handshake process to bootstrap anchors (key chain commitments) of the hash chains. Thereafter, when the signer has a message m to send, it first computes the MAC of m keyed with an unreleased key of the signature chain $h_{i-1}^{S_s}$. Then the signer, as illustrated in Figure 4.3, sends out the packet $S_1 :< h_i^{S_s}, M(h_{i-1}^{S_s} \| m) >$ to the verifier where $h_i^{S_s} = H(h_{i-1}^{S_s})$, used to identify the signer. Each intermediate node also buffers $M(h_{i-1}^{S_s} \| m)$ while waiting for the issue of m. Upon receiving S_1, the verifier sends a packet $A_1 :< h_i^{V_a}, h_i^{S_s} >$ authenticated with an element $h_i^{V_a}$ of the acknowledgment chain back to the signer. On receipt of the valid A_1, the signer discloses the key of the MAC $h_{i-1}^{S_s}$ and m in a packet $S_2 :< h_{i-1}^{S_s}, m >$. Once $h_{i-1}^{S_s}$ and m are disclosed, it is straightforward for intermediate nodes and the verifier to verify the authenticity of m.

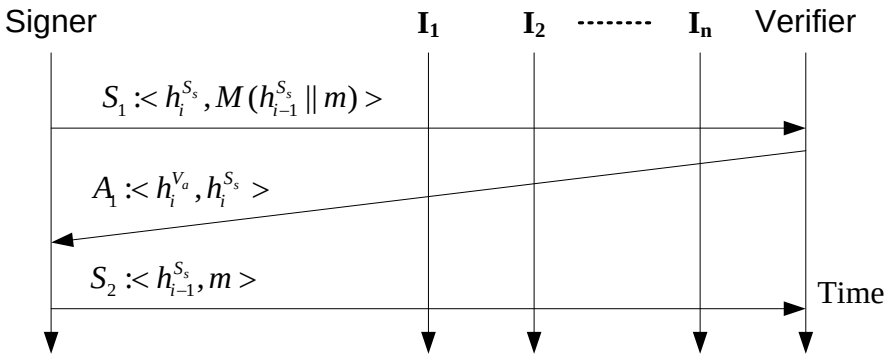


Figure 4.3: Illustration of the basic ALPHA

Nonetheless, the ALPHA bootstrapping problem becomes challenging if sensor nodes need to authenticate their messages. A very naïve method for this problem is to require that each node stores all of the anchors of the other nodes in the network. However, this results in a high storage overhead for memory-constrained sensor nodes. One plausible solution is to combine the identity-based signature approach which will be discussed further in section 4.2.2 with ALPHA. The idea is that the sender signs its anchor using an identity-based signature scheme before sending the anchor to the verifier. Upon the arrival of the signed anchor, the verifier generates the sender's public key from the sender's ID and use the key to verify the authenticity of the received anchor. The actual communication between the sender and the receiver only starts when the validity of the anchor is assured.

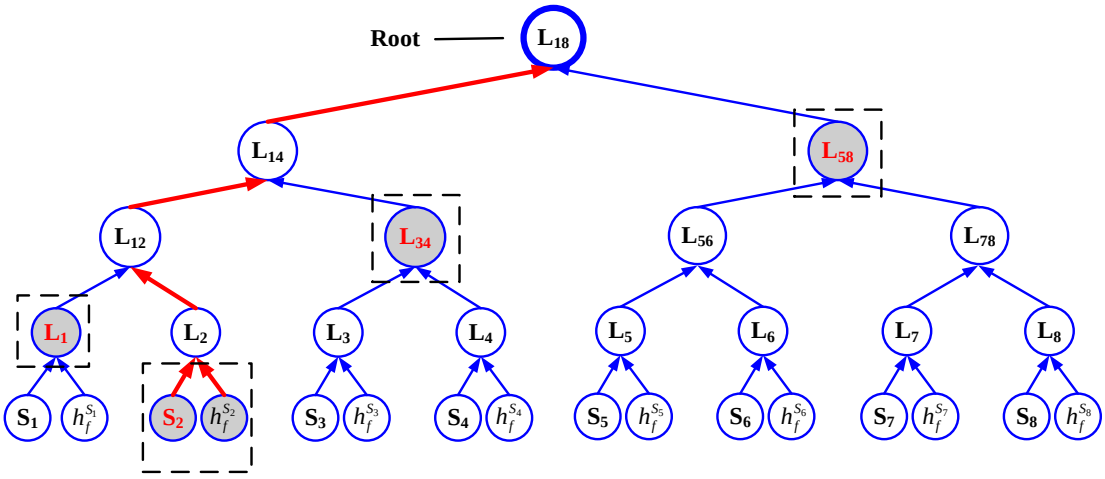


Figure 4.4: An illustrative anchor distribution tree for eight sensor nodes

Another solution can be the adaptation of the work in [139]. The basic idea is to use a Merkle hash tree [140] to allow nodes to authenticate other nodes' anchors. Before describing how to build the tree, let us denote H as a hash function, $h_f^{S_i}$ as the anchor of node S_i, and n as the number of nodes in the network. To build the tree, the security server first computes $L_i = H\left(S_i \,||\, h_f^{S_i}\right)$, $i = \overline{1,n}$, and constructs the Merkle tree (anchor distribution tree) using $L_1,...,L_n$ as leaf nodes. Specifically, $L_1,...,L_n$ are arranged as leaf nodes of a full binary tree and each non-leaf node is computed by applying H to the concatenation of its two children nodes. Figure 4.4 illustrates the anchor distribution

tree for eight sensor nodes, where $L_1 = H\left(S_1 \| h_f^{S_1}\right)$, $L_{12} = H\left(L_1 \| L_2\right)$, $L_{14} = H\left(L_{12} \| L_{34}\right)$, $L_{18} = H\left(L_{14} \| L_{58}\right)$.

To enable the anchor bootstrapping, each sender S_2, for instance, is pre-loaded with a certificate $Cert_2 = \left\{S_2, h_f^{S_2}, L_1, L_{34}, L_{58}\right\}$ while each verifier S_6, for example, is pre-loaded with the root of the tree L_{18}. When the sender S_2 needs to bootstrap its anchor $h_f^{S_2}$, it sends out $Cert_2$ to the verifier S_6. S_6 can promptly verify the authenticity of the anchor by checking if $H\left(H\left(H\left(L_1 \| H\left(S_2 \| h_f^{S_2}\right)\right) \| L_{34}\right) \| L_{58}\right)$ equals the pre-distributed root L_{18}. As a result, the verifier can get the authenticated anchor from the sender. To further reduce the communication cost of transmitting certificates, the authors in [139] introduced several trimming techniques. However, this reduction comes at the cost of some increase in memory usage.

4.2.2 Public-Key Approaches

In addition to the symmetric-key approaches, message authentication can be achieved from public-key cryptosystems using digital signatures. Numerous digital signature schemes for both source and data authentication have been proposed so as RSA algorithm [127], Rabin algorithm [141], Schnorr algorithm [142], digital signature algorithm (DSA) [143], elliptic curve DSA [144], ElGamal signature scheme [145], identity-based signature schemes [146]. Among them, elliptic curve cryptography (ECC) based and identity-based signature schemes are more desirable for use in DSNs due to two reasons. First, they make use of the advantages of ECC. Recent studies [37, 69, 72, 147] show that ECC is especially attractive and viable to resource-constrained sensor nodes due to its fast computation, smaller key size, and compact signature. For instance, to provide equivalent security to 1024-bit RSA, an ECC scheme only needs 160 bits on various parameters, such as 160-bit finite field operations and 160-bit key size [69]. Further, as evaluated in [72], the implementation of ECC package TinyECC including ECDSA, ECDH, and ECIES consumes about 10,374 bytes in ROM and 786 bytes in RAM on MICAz motes. In terms of computation, ECDSA in TinyECC requires around 2001.62 ms for signature generation and 2436.46 ms for signature verification

on MICAZ motes. Second, they inherit appealing features from identity-based cryptography. In identity-based cryptosystems, a user's public key is directly derivable from her publicly known identity information. The user's private key is computed by a trusted party, called Private Key Generator (PKG). Therefore, the user's identifier serves as her public key while the user's private key is in fact her public key certificate. Due to this dynamics, IBC requires no public key certificate and thus eliminates the need for certificate transmission and verification.

One salient candidate of ECC-based IBS schemes is BNN-IBS proposed by Bellar et al. [148]. However, BNN-IBS is not efficient when applied in DSNs due to its large signature size. To optimise the signature size, a variant of BNN-IBS called vBNN-IBS was proposed by Cao et al. in [149]. In vBNN-IBS, the following symbols are used:

- $\mathbb{E}/\mathbb{F}_q$: Elliptic curve $\mathbb{E}$ over a prime finite field $\mathbb{F}_q$ defined by an equation $y^2 = x^2 + ax + b$ with $a, b \in \mathbb{F}_q$ such that $\Delta = 4a^3 + 27b^2 \neq 0$.
- $\mathbb{E}(\mathbb{F}_q)$: Group of points formed by the points on $\mathbb{E}/\mathbb{F}_q$ and extra point o called the point at infinity. $\mathbb{E}(\mathbb{F}_q) = \{(x, y) : x, y \in \mathbb{F}_q; (x, y) \in \mathbb{E}/\mathbb{F}_q\} \cup \{o\}$.
- m: Order of $\mathbb{E}(\mathbb{F}_q)$.
- p: Prime number such that $p^2 \nmid m$.
- P: Point of order p.

In the setup phase of BNN-IBS, given the security parameter k, PKG takes the following steps:

i) Select a system secret key x at random from $\mathbb{Z}_p$ and set the system public key $P_0 = xP$.

ii) Choose two cryptographic hash functions $H_1 : \{0,1\} \times \mathbb{G}_1^* \rightarrow \mathbb{Z}_p$ and $H_2 : \{0,1\}^* \rightarrow \mathbb{Z}_p$.

iii) Publish system parameters $\langle \mathbb{E}/\mathbb{F}_q, P, p, P_0, H_1, H_2 \rangle$ and keep x secret.

In the node key extraction phase, given a node S_i's unique identifier $S_1 \in \{0,1\}^*$, PKG generates S_i's private key Pri_{S_i} based on Schnorr signature [142] as follows:

i) Choose at random $r \in \mathbb{Z}_p$ and compute $R = rP$.

ii) Use system secret key x to compute $s = r + cx$, where $c = H_1(S_1 \| R)$.

S_i's private key is the pair $Pri_{S_i} = (R, s)$, and is pre-loaded into S_i before deployment.

In the signature generation phase, S_i signs a message msg with its private key Pri_{S_i} as follows:

i) Choose at random $y \in \mathbb{Z}_p$ and compute $Y = yP$.

ii) Compute $h = H_2(S_1, msg, R, Y)$ and $z = y + hs$.

The tuple $\langle R, h, z \rangle$ is S_i's signature on msg.

In the signature generation phase, given $\langle R, h, z \rangle$, S_i and msg, a verifier first computes $c = H_1(S_1 \| R)$. Then it checks if the condition $h = H_2(S_1, msg, R, zP - h(R + cP_0))$ holds. msg is accepted as authentic if the condition holds otherwise it is rejected.

4.3 DoS Prevention for Broadcast Authentication

In the context of broadcast authentication, DoS attacks are perceived as a process of flooding faked and format-compatible packets in order to exhaust bandwidth, computation and memory resource network-wide. Typically, sensor nodes participating in the broadcast process follow forwarding-first approach, i.e. forward faked packets to their neighbours before they verify the packets. Consequently, the faked packets will be spread across the entire network. Even though these packets are discarded after verification, they cause the following serious problems [150]:

- Packet verification wastes scarce energy and computational resources.
- Temporary storage of the packets before they are verified wastes buffer space.
- Transmission of the packets wastes limited wireless bandwidth.

A straightforward countermeasure against this kind of attacks is to verify each packet before forwarding it (authentication-first method). In this manner, the faked packets will be discarded at the first-hop neighbours of malicious nodes and thus nodes beyond these neighbours will not be battery exhausted. Although this countermeasure can provide instant rejection of the faked packets, it does increase the response time of sensor nodes to broadcast packets.

Wang et al. proposed a dynamic window scheme in [151] which combines both forwarding-first and authentication-first methods to achieve a good trade-off between the broadcast delay for authentic messages and resource savings for faked messages. In the dynamic window scheme, there are two parameters under consideration: authentication window size (ω) and distance (d_a). ω, maintained by each sensor node, defines the maximum number of hops an incoming message can be forwarded without being verified. d_a is a field in each broadcast message and used to record the number of hops the message has passed since its last authentication. When the node receives the message, it compares ω with d_a. If $d_a < \omega$, the node is in the forwarding-mode. It then increases d_a and forwards the message before verifying it. In contrast, when $d_a \geq \omega$, the node is in the authentication-first mode. It then authenticates the message first. If the authentication fails, the node discards the message. Otherwise, it resets d_a to 0 and forwards the message to its next hop neighbours. The increase and decrease of ω follow the additive increase multiplicative decrease law: $\omega = \omega + 1$ (increase) and $\omega = \left\lfloor \frac{\omega}{2} \right\rfloor$ (decrease). Figure 4.5 demonstrates how the dynamic window scheme works.

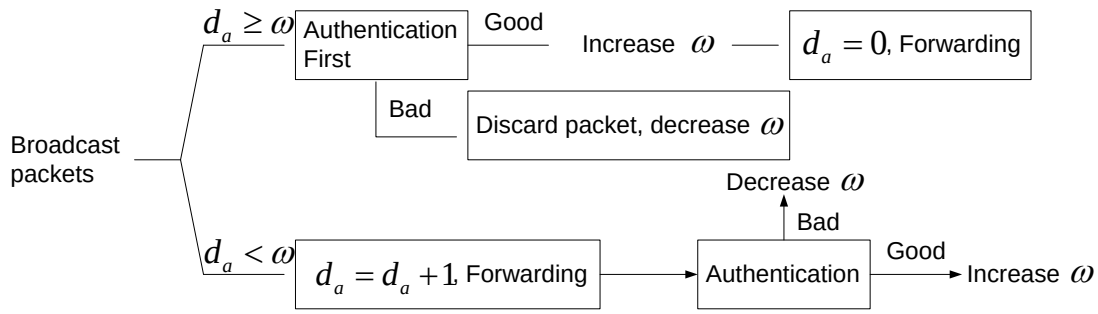


Figure 4.5: Illustration of dynamic window scheme

Another approach to DoS attacks on broadcast authentication was presented in [152]. The key idea is to equip sensor nodes with pre-authentication filters used to remove bogus messages in a hop-by-hop manner before nodes actually verify digital signatures. There are two types of filters discussed: group-based filter and key chain-based filter. In the group-based method, every sender divides its neighbours into multiple groups. A group key is shared among the sender and each node of the group. This group key acts as a pre-authentication filter to filter out forged messages. The key chain-based filter technique consists of two filtering layers. The first layer makes use of a one-way key chain as discussed in LHAP [153] to filter out fake signatures. The second layer deals with the DoS attacks on the verification of chained key by controlling the number of hash operations used for verifying a chained key. Accordingly, when a receiver realises that the verification of the key requires many hashing operations, it decides to wait for additional evidence before continuing the verification. The additional evidence such as commitment value can be derived from the shared key with the sender.

4.4 Anti-Jamming Techniques

Due to the shared use of wireless medium, wireless radio communications are highly susceptible to jamming attacks. To launch the attack, a jammer simply transmits jamming signal in the wireless channel while the legitimate transmission is occurring. As a consequence, the attacker can achieve a DoS by blocking, modifying, destroying, or overwriting the original signal.

Spread spectrum techniques such as Direct Sequence Spread Spectrum (DSSS), Frequency Hopping Spread Spectrum (FHSS), and Chirp Spread Spectrum (CSS) have been considered as effective countermeasures against jamming attacks. However, these techniques have to depend on shared secrets to generate identical hopping patterns, spreading codes, or timing of pulses for communication between senders and receivers. This dependency is not desirable because in the setting of broadcast communication a pair of sensor nodes may not have shared secrets before communication. Furthermore, if jamming attackers know the secrets, the spread spectrum techniques fail in their counteractive goal.

A number of recent attempts have been made to eliminate the dependency of existing anti-jamming techniques on shared keys. For example, Baird et al. [154] introduced a coding approach to encode transmitted data into “marks” such as short pulses at different times that can be decoded without any prior knowledge of keys. Another example is Uncoordinated Direct Sequence Spread Spectrum (UDSSS) approach developed by Pöpper et al. [155]. This approach makes use of a public set of spreading sequences used by the sender and the receivers. The sender transmits a message by selecting a spreading sequence from the code set and spreading the message with this sequence. The receivers record the signal on the channel and despread the message by applying sequences from the set code using trial-and-error method.

Liu et al. proposed a randomised differential DSSS scheme (RD-DSSS) [156] in the latest work against jamming attacks. In RD-DSSS, the sender and receivers share a set of spreading codes. When the sender needs to send a message, she encodes each bit of the message according to the correlation of two unpredictable spreading codes. In more detail, bit “1” is encoded by two identical and thus highly correlated spreading codes while bit “0” is encoded by two different and thus low correlated spreading codes. Different bits in the message can be encoded using randomly chosen and different pairs of codes from the code set. In the de-spreading process, high correlation and low correlation are translated into “0” and “1” respectively at the receiver’s side.

4.5 Secure Localisation Algorithms

Secure localisation algorithms deal with how to assign locations to sensor nodes consistently with respect to some local or global coordinate system in the presence of malicious attacks. Due to their importance to sensor network applications, a great number of secure localisation schemes have been proposed for sensor networks. One approach to secure localisation is to detect the cheating nodes and eliminate them from consideration over the localisation process [157-159]. For instance, Park and Shin [159] proposed a method for attack-tolerant localisation via iterative verification of locations. In their work, they made use of anomaly detection approaches to secure localisation process. Accordingly, each sensor node maintains and adaptively updates a baseline profile of normal localisation behaviours based on past announcements of all its

neighbours. Afterwards, when it receives new announcements, it compares them with this normal profile. If the node detects a remarkable deviation from maintained profile, it decides on the presence of a possible malicious behaviour and takes action to locate the malicious nodes.

The second approach does not focus on detecting and eliminating malicious nodes (or beacons) but developing mechanisms that exhibit sufficient robustness to tolerate the cheating impact of malicious nodes [160-163]. An example of this approach is the voting-based location estimation method presented in [161]. In this method, the deployment area is divided into a grid of cells such that the target node resides in one of the cells. Each beacon node votes on each cell depending on the distance between the target node and itself. The location of the target node is estimated as being within the cell that had the maximum number of beacon votes.

4.6 Secure Time Synchronisation Protocols

Time synchronisation plays an essential role in the maintenance of time precision among sensor nodes which is required by many sensor applications and protocols, such as measurement of round-trip time for localisation, formation of time division multiple access (TDMA) radio scheduling, coordination of sensor nodes' sleep-wakeup schedules, prevention of replay attacks [164]. Hence, it is vital to secure time synchronisation protocols in hostile environments. Ganeriwal et al. [165, 166] introduced a suite of secure time synchronisation protocols based on cryptographic techniques. The protocols are adapted to pairwise single-hop, multihop, and group synchronisation. The protocols can also detect the existence of a pulse delay attack by computing the end-to-end delay of the message and comparing the delay with a predetermined threshold.

Song et al. [167] proposed two approaches based on techniques of outlier detection to time synchronisation which are resilient to delay attacks mounted by compromised nodes. These approaches rely on the assumption that the time offsets received from compromised nodes will be much different from others. Therefore, messages coming

from compromised nodes can be identified as outliers using statistical methods and filtered out. The synchronisation process continues normally with the remaining nodes.
Sun et al. [168] proposed a TinySeRSync system with the aim of providing secure time synchronisation among sensor nodes. TinySeRSync assumes there exists a source node with which all the sensor nodes are synchronised in two asynchronous phase. In phase I, pairs of neighbouring nodes periodically exchange messages, protected by authentication, with each other to achieve single-hop pairwise time synchronisation. The goal of phase II is to obtain global time synchronisation. This can be accomplished via broadcast synchronisation messages periodically sent from the source node. These broadcast messages are authenticated by μTESLA [67].

Hu et al. [169] took a distributed and mutual synchronisation approach to developing an Attack-tolerant Time Synchronization protocol (ATSP). In fact, ATSP is a cooperative intrusion detection system where sensor nodes can detect abnormal synchronisation information by adaptively building and tracking the normal profile of synchronisation behaviours of their neighbouring nodes. To avoid being detected, attackers have to weaken their attack power to small perturbations which will be smoothed out by distributed synchronization where each sensor node adjusts its own clock based on multiple neighbours' clock values. The clock drift in-between adjacent synchronisation points is compensated by utilising "calibrated clock". That is, each sensor node uses its neighbours' normal profiles to estimate their clock drift without further communication.

4.7 Chapter Remark

In this chapter, a wide range of research issues in sensor networks as well as respective up-to-date solutions have been discussed. These include cryptographic primitives, message authentication, DoS prevention for broadcast authentication, anti-jamming techniques, secure localisation, and secure time synchronisation. The aim of the chapter is to provide various options of related work which can be chosen and utilised later in Chapters 6 and 7 as building blocks to develop proposed countermeasures against controlled link establishment attacks.

Chapter 5

Feasibility of Controlled Link Establishment Attack

This chapter investigates to affirm that CLEA indeed poses a real security threat to DSNs. This affirmation is supported by the feasibility of compromising sensor nodes to extract secret information, the availability of methods for repetitive use of compromised nodes and secret information in CLEA. Moreover, a prototype of CLEA is implemented in the last section of the chapter to further demonstrate the likelihood of CLEA. To the best of our knowledge, this prototype implementation is the first effort made to establish the feasibility of CLEA.

5.1 Feasibility of Node Compromise

To launch the CLEA, the attacker must first compromise a limited number of sensor nodes in a stealthy manner. As well-accepted in the referenced literature, the node compromise in DSNs is more than likely to occur due to the low-cost and throwaway nature of sensor nodes, the unattended operation of the network, and the hostility of deploying environments. In essence, the node compromise involves physical access to the sensor field to capture sensor nodes and hardware tampering to gain read/write access to the micro-controller and the memory of sensor nodes. The physical capture of sensor nodes can be performed with ease because the network is usually left unattended for long periods of time. Hardware tampering can be carried out on-site or in a well-equipped distant laboratory after captured nodes have been physically removed from the network. Therefore, the time required to tamper with sensor hardware varies greatly from several seconds [44, 46] to several years [46]. The tampering time also varies according to whether anti-tamper measures are applied and how they are implemented on sensor hardware.

Most of the current research on node compromise in DSNs assumes that it is quite straightforward to compromise a node. This is due to several reasons. First, most of the current sensor hardware has been designed and implemented without taking the node compromise into account. As a consequence, the current sensor hardware such as MICA2 [20], Micaz [49], Telos [113], ESB [114] exhibit features which can be exploited by the attacker for the tampering purpose. For example, both the Atmel and Texas Instruments micro-controllers used on the sensor hardware above support the IEEE 1149.1 JTAG standardised interface [115]. This interface allows programmers to read from and write to memory and execute on-chip debugging which includes single stepping through code as well. A attacker over this interface can gain the full capacity for compromising sensor nodes using a JTAG ICE pod, a programming board with a JTAG connector, and a free serial port (from a laptop) to connect the JTAG ICE pod [116]. Second, most of sensor nodes are comprised of "off the shelf" components and programmed using open-source software (the source code is widely available). This enables the attacker to promptly pinpoint the location for storing security sensitive information in sensor nodes. Third, tamper resistant hardware is yet to be available on sensor nodes because of the significant increase in cost, reduction of the leeway for user/programmer error and the elimination of the reprogramability [44].

Even when sensor nodes are protected by anti-tamper technologies such as used in smartcards and other security processors they are not absolutely immune from a wide variety of the physical tampering methods. According to [117], these methods can be classified into two major categories: *invasive attacks* and *non-invasive attacks*. Examples of the invasive attacks are micro-probing techniques which require chip testing equipment such as probing stations and focused ion beam workstations in order to observe, manipulate, and interfere with the integrated circuit. They usually take hours or weeks in a specialised laboratory and in the process of destroying the packaging. In addition, they require very little initial knowledge and usually work with a similar set of techniques on a wide range of products. Examples of the non-invasive attacks include software attacks, eavesdropping, and fault generation. In contrast to the invasive attacks, launching non-invasive attacks requires specialised expertise in both the processor and software. They usually involve the exploitation of unintentional electromagnetic emissions, protocol design flaws, and other vulnerabilities that manifest themselves externally.

To further emphasise the susceptibility of tamper resistant technologies, Skorobogatov and Anderson [118] introduced a new class of attacks termed “semi-invasive”. These attacks are similar to invasive attacks in the sense that they require de-packaging the chip to get access to the chip surface but they keep the passivating layer of the chip intact. Although, the semi-invasive attacks are considered to be more costly than the non-invasive attacks, they are much less expensive than the invasive attacks which require a relatively high capital investment for lab equipment. As reported by The New York Times newspaper [119], secret information contained in widely used tamper resistant devices can be extracted using a semi-invasive technique that employs only a $30 camera flashgun and a microscope.

5.2 Methods for Repetitive Use of Compromised Secret Information

To amplify the impact of CLEA on the network, the attacker has to increase the number of controlled links between controlled nodes and legitimate nodes under the constraint imposed by the limited amount of secret information obtained from node compromise. This constraint can be overcome by using the compromised nodes and secret information repetitively without being detected at different network locations. There are several methods for this use. They can be classified into cloning method and swapping method. The swapping method can be divided further into short-distance swapping method, long-distance swapping method, and mixed-distance swapping method. In addition, these methods can be mixed together to obtain even better performance.

5.2.1 Secret Information Cloning

In this method, the attacker loads the obtained secret information onto multiple off-the-shelf sensor nodes to turn them into the clones of the compromised sensor nodes. However, these clones, also referred to as controlled nodes in this thesis, differ from legitimate nodes in the aspect that in addition to the normal code, they also have malicious code added by the attacker. These clones are then inserted by the attacker into the network to establish controlled links.

The secret information cloning method is easy to implement since it only needs a programming tool to copy and write the obtained secret information into the controlled nodes' memory. Nevertheless, it may incur high hardware cost because the number of controlled links created by this method is proportional to the number of controlled nodes added into the network. Consequently, if the attacker wants to gain more controlled links by this method, it has to spend more money on sensor hardware.

5.2.2 Short-Distance Swapping

This method is the most cost-effective way provided that the following conditions are satisfied:

- The controlled nodes receive no attacker's communication support after having been put back into the network. In other words, the attacker neither installs any extra communication hardware onto the controlled nodes nor deploys additional communication infrastructure into the sensor field.
- Each of the controlled nodes is reachable by the other controlled nodes via colluding tunnels established based on existing routing mechanisms. For example, the controlled nodes can make use of existing routing infrastructure to discover each other initially. Thereafter, they exchange necessary information such as a list of the closest controlled nodes to establish colluding tunnels. These tunnels are then used to swap or update the controlled nodes' secret information.
- Every controlled node is reachable by at least one of its peers over a short distance. Ideally, that distance should be at most four hops long since within such a distance the controlled nodes can swap their keys in timely manner without consuming too many resources and easily escape from being detected by wormhole attack detection mechanisms such as geographical and temporal leashes [120].

For the sake of presentation, let us assume a short-distance swapping scenario of eight controlled nodes numbered from 1 to 8 where, for each node, it is always possible to find another node such that the distance between them is not more than four hops away as illustrated in Figure 5.1. Take the secret information swapping among three

controlled nodes 1, 2, and 3 for example; although node 3 has four nodes in its neighbourhood, it can only establish a pairwise key with node 11. Furthermore, assume that, according to a given key establishment scheme, the secret information of nodes 1 and 2 can be used to establish pairwise keys with two nodes 9 and 10 respectively. When no swapping occurs, the controlled nodes obtain no added gain. Nevertheless, through the swapping node 3 can achieve the IDs of nodes 1 and 2 and their secret information. Using the acquired information, node 3 can establish controlled links with nodes 9 and 10. By this manner, the attacker can expand its attack and eventually gain the control of most of the network.

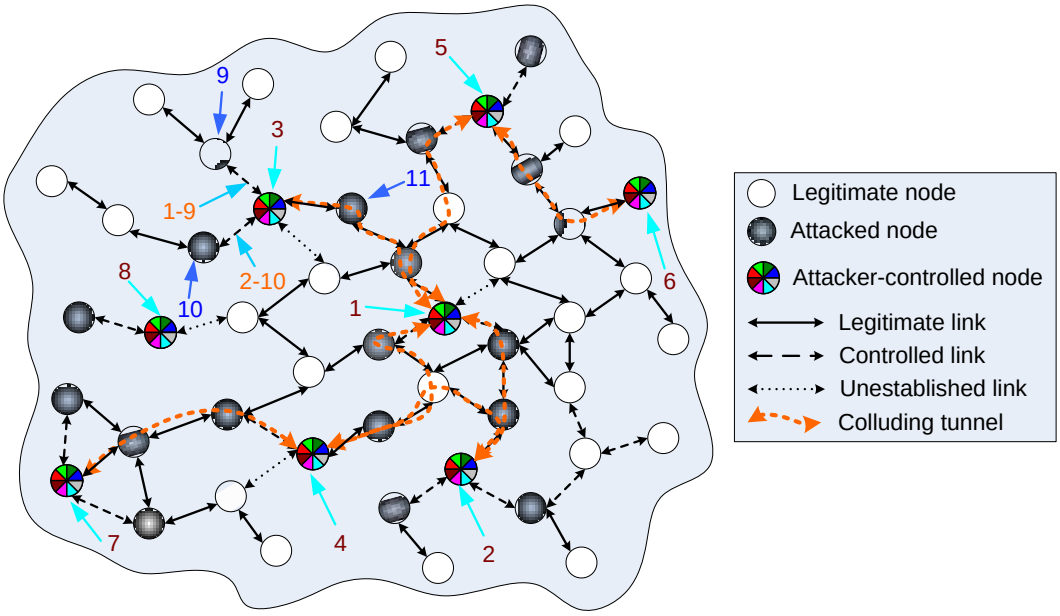
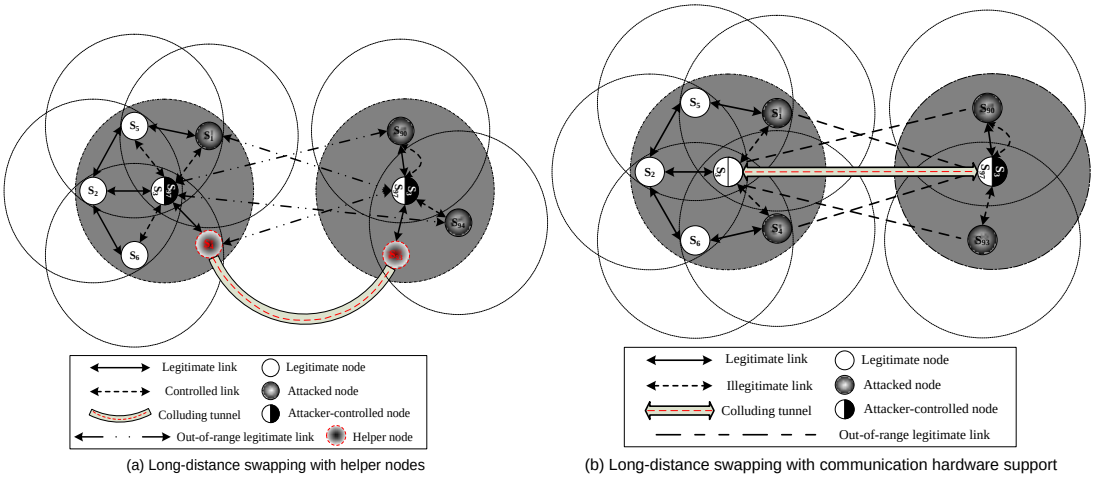


Figure 5.1: An illustration of the short-distance secret information swapping method

5.2.3 Long-Distance Swapping

In this method, the distance between any pair of two controlled nodes is far enough such that it is not reasonable for them to communicate with each other using existing routing mechanisms. For efficient communications among the controlled nodes, low-latency links can be utilised such as a wired connection, an optic connection, or long-range, out-of-band wireless directional transmission. These links are made possible if the attacker can gain access to the sensor field to deploy extra long-range, low-latency

communicating devices (referred to as helper nodes) as neighbours of the controlled nodes as illustrated in Figure 5.2(a). Alternatively, the attacker can install additional pieces of communications support hardware onto sensor main boards as shown in Figure 5.2(b).



(a) Long-distance swapping with helper nodes

(b) Long-distance swapping with communication hardware support

Figure 5.2: The illustrations of long-distance collusion attack

Using the long-distance swapping method, CLEA is somewhat similar to the wormhole attack [120, 121] in the sense that some information is obtained at one point of the network, then tunnelled to another point of the network via a long-range, low-latency link, and injected back into the network. However the two attacks are different from each other due to the following points:

- Node compromise is compulsory in long-distance swapping CLEA while it is not mandatory in the wormhole attack.
- Information exchanged in CLEA is secret information while in the wormhole attack the exchanged data is mainly routing information, neighbourhood information, and location information.
- Target of CLEA are key establishment schemes while the wormhole attack mainly aims at subverting routing protocols, neighbour discovery mechanisms, and local broadcast protocols such as localisation protocols.

5.2.4 Mixed-Distance Swapping

This sort of method might happen when the controlled nodes are distributed unevenly. In other words, there are some regions where the controlled nodes can deploy short-distance swapping efficiently, whereas in other regions, long-distance swapping might be a more suitable choice. Note that it is possible that two kinds of methods can co-exist in one controlled node. That is, on one side, the node has short-distance swapping tunnels with other controlled nodes in its close vicinity. On the other side, it also has long-distance swapping tunnels with the other remote ones.

5.3 CLEA Prototype Implementation

The goal of the CLEA prototype implementation is to further demonstrate the feasibility of mounting CLEA. The following will detail how this prototype is implemented.

5.3.1 Assumptions

In this implementation, the following assumptions are made:

- Key establishment scheme: Similar to [41], this implementation also selects the random-pairwise keys scheme by Chan et al. [28] as the key establishment scheme which needs to be protected from CLEA. The two phase scheme is summarised as follows. In the initialization phase, unique IDs are generated for a sensor network of up to $n = m/p$ nodes. Each node is assigned different m pairwise keys linking the node with m other randomly selected distinct nodes. The keys are pre-stored in the node's key ring along with the corresponding IDs of the selected nodes. In the post-deployment key setup phase, each node first broadcasts its ID to its one-hop neighbours. The neighbouring nodes are able to determine a pairwise key shared with the broadcasting node if they find its ID in their key rings. According to this bootstrapping method, apparently, the probability of a pairwise key existing between two nodes is p. This probability is calculated such that the network connectivity is achieved with high probability using random graph theory [122].

- Network topology: As shown in Figure 5.3, this implementation assumes that the network consists of four sensor nodes 1, 2, 3, and 4. The reason behind this assumption is that we have to keep within a tight budget for purchasing sensor node hardware. Nodes 1 and 4 are innocent while nodes 2 and 3 are the controlled nodes. Node 1 has only one neighbour which is node 2. Node 2 has two neighbours 1 and 3. Node 3 has two neighbours 2 and 4. Node 4 has only one neighbouring node which is node 3. It is assumed that nodes 1 and 4 share common keys with nodes 3 and 2 respectively. Meanwhile there is no shared key between 1 and 2 or between 3 and 4.
- Methods for use of compromised secret information: The method used in this implementation is the short-distance swapping as discussed in Section 5.2.2.

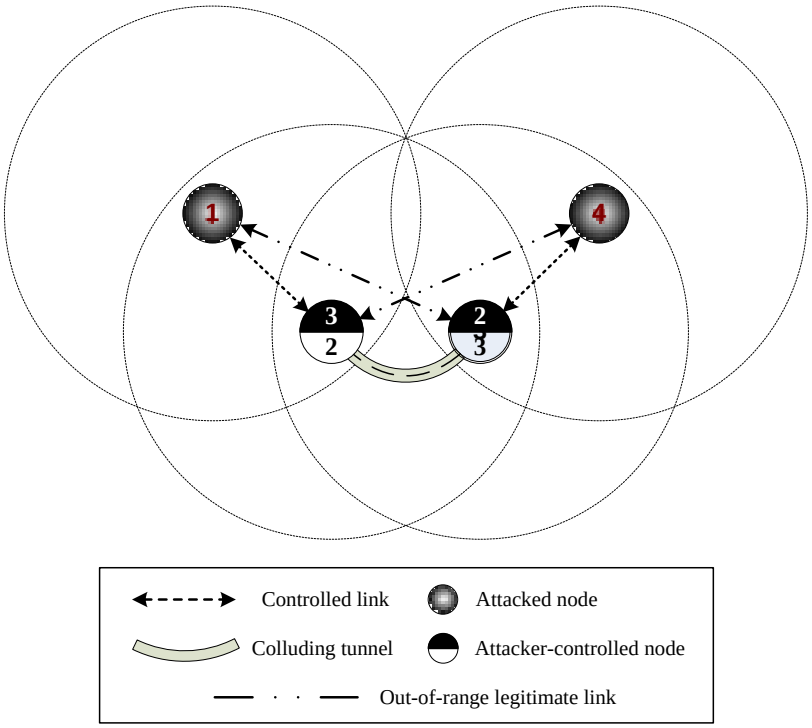


Figure 5.3: The network scenario of the CLEA prototype implementation

5.3.2 Software Tools and Hardware Devices

- Software tools: The prototype is implemented in Cygwin [123], a Linux-like environment for Windows. It consists of two parts: (i) A DLL (cygwin1.dll) which acts as a Linux API emulation layer providing substantial Linux API

functionality. (ii) A collection of tools which provide Linux look and feel. In addition to Cygwin, we also need TinyOS 1.1 [21], Sun JDK [124], Sun's javax.comm package [125]. TinyOS is a simple, event-driven, and application-specific operating system designed for resource-scarce sensor networks. TinyOS supports an event-driven concurrency model based on split-phase interfaces, asynchronous *events*, and deferred computation called *tasks*. The TinyOS system, libraries, and applications are written in nesC [126], a language for programming structured component-based applications primarily intended for embedded systems. nesC has a C-like syntax, but supports the TinyOS concurrency model, as well as mechanisms for structuring, naming, and linking together software components into robust network embedded systems. The principal goal is to allow application designers to build components that can be easily assembled into complete and concurrent systems, and yet perform extensive checking at compile time.

- Hardware devices: The hardware devices used in this implementation consist of a desktop computer, four MICAz motes from Crossbow Technology, a Crossbow's serial port based programming device MIB510CA, and a USB to Serial Adapter connecting MIB510CA with the computer.

5.3.3 Implemented Applications

Using the software tools and hardware devices mentioned in the previous section, we implemented two TinyOS applications: RandPKS and CtrlledNode. RandPKS is used by neighbouring nodes to establish pairwise keys in accordance with the random-pairwise keys scheme. CtrlledNode allow controlled nodes to not only follow the random-pairwise keys scheme but also be able to swap their secret information. As assumed in section 5.3.1, nodes 1 and 4 are innocent whereas nodes 2 and 3 are controlled by the attacker. Therefore, nodes 1 and 4 are programmed with RandPKS while nodes 2 and 3 are programmed with CtrlledNode.

The graphical representation of the component relationships within RandPKS is shown in Figure 5.4. As seen in the figure, RandPKS consists of the following components:

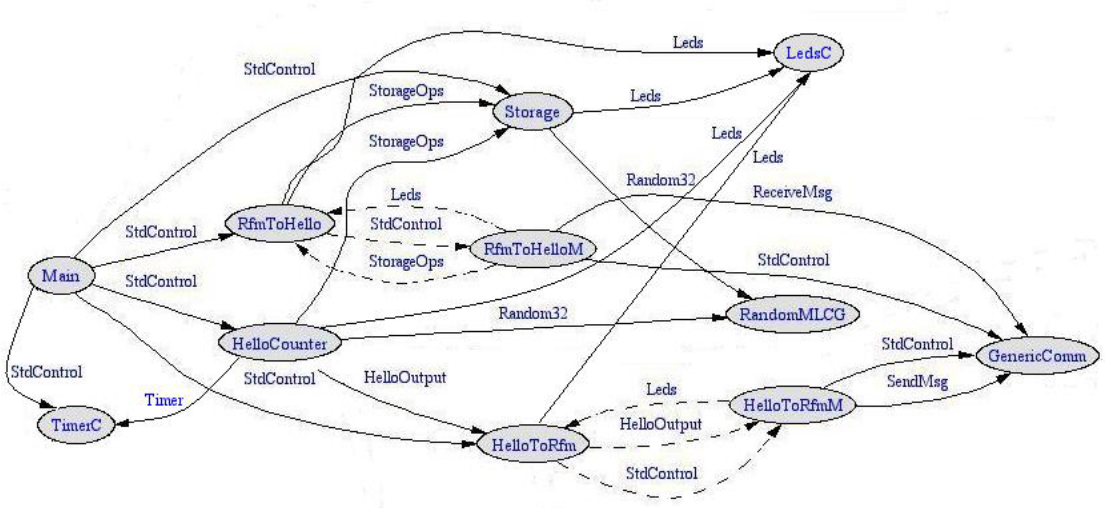


Figure 5.4: The component relationship within RandPKS

- Main: This component is provided by TinyOS developers and executed first in a TinyOS application when a node boots up. It is used to initiate other components of the application including RfmToHello, HelloToRfm, HelloCounter, Storage, and TimerC through StdControl interfaces embedded in these components. In general, StdControl is a common interface used to initialize and start TinyOS components. It defines three commands, *init()*, *start()*, and *stop()*. *init()* is called when a component is first initialized and then *start()* when the component is started, that is, actually executed for the first time. *stop()* is called when the component is stopped, for example, in order to power off the device that it is controlling.
- HelloCounter: This component is executed after the Main component. It instructs a node to wait for a short period of time chosen randomly before broadcasting a hello message, which contains the broadcaster's ID, to its neighbours. The random value for the period is provided by the RandomMLCG component. The transmission of the hello message occurs when the time interval provided by the TimerC component expires. The node keeps sending the hello message until it receives hello messages from the neighbours. The status of the hello message receipt is kept track by a Boolean variable stored in the Storage component.

- HelloToRfm: This component is used by the HelloCounter component to send the hello message downwards to the physical layer. It uses the LedsC component to provide the indication of message sending activity.
- RfmToHello: This component is used to receive the hello message from the physical layer. After the receipt of the message finishes, the component will look for a common key shared with the message sender in the receiver's key ring. The Leds controlled by the LedsC component are used to indicate the message receiving and shared key finding activities.
- Storage: This component is used to pre-store a key ring of a sensor node. The key ring is implemented using two arrays. The first array is used to store the IDs of sensor nodes sharing keys with the node. The second array is used to store keys corresponding to the IDs in the first array. Storage also provides functions to get and set the status of the hello message receipt, to store neighbours' IDs, and to search for common keys.

CtrlledNode consists of the following components as depicted in Figure 5.5:

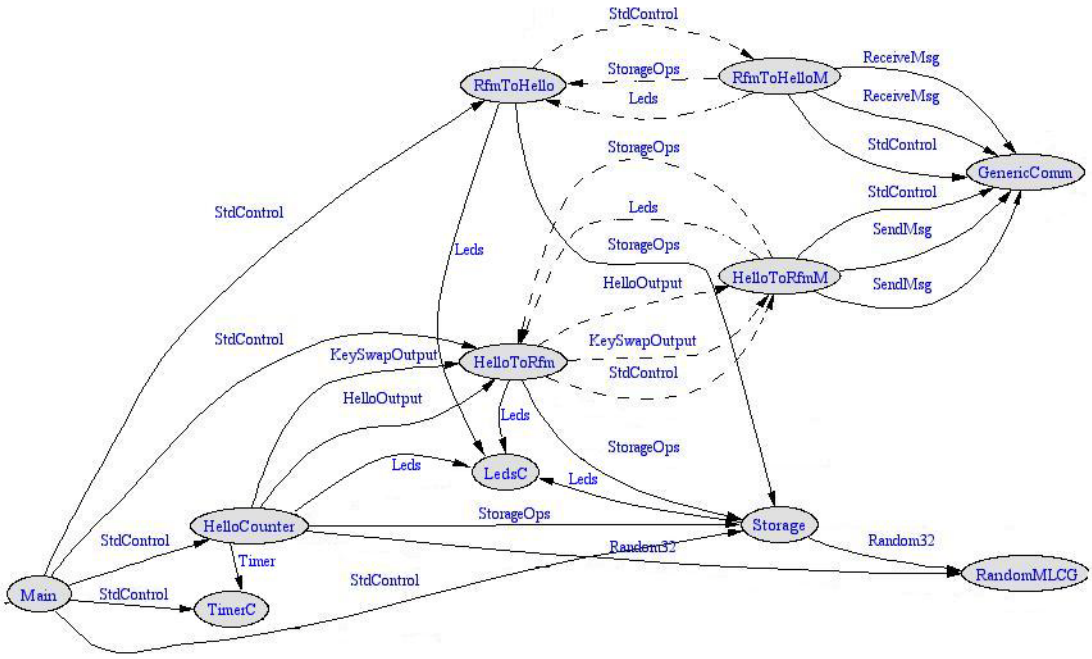


Figure 5.5: The component relationship within CtrlledNode

- Main: Similar to Main in RandPKS, this component is used to initialise the other components of CtrlledNode including HelloCounter, HelloToRfm, RfmToHello, and TimerC.
- HelloCounter: This component is designed to carry out two tasks: swapping the secret information between node 2 and node 3 and sending hello messages. After being deployed, nodes 2 and 3 first swap their secret information. Only when the secret information swapping process is complete, do they start broadcasting hello messages to their neighbourhood.
- HelloToRfm: This component is different from HelloToRfm in RandPKS in the aspect that it does not only send the hello message but also the key-swapping message, which contains the sender's node ID and key ring, down to the physical layer for transmission.
- RfmToHello: The only difference between this component and RfmToHello in RandPKS is that this component is also used to pick up the key-swapping message from the physical layer and save the secret information in the received message to the receiver's memory storage.
- Storage: In addition to the features provided by Storage in RandPKS as described above, Storage in CtrlledNode provides space to store and retrieve the swapped secret information furnished by RfmToHello.
- TimerC: This component is used by HelloCounter to determine when to send the key-swapping message.
- LedsC: This component is used by HelloCounter, HelloToRfm, RfmToHello, and Storage to indicate the activities of sending, receiving, storing, and retrieving information.
- RandomMLCG: This component is used by HelloCounter and Storage to obtain random values for their activities.
- GenericComm: This component provides HelloToRfm and RfmToHello with facilities to send messages to and receive messages from various physical layers.

5.3.4 Operation of the Demonstration

The demonstration of short-distance swapping CLEA consists of two steps. The first step is to show that without the secret information swapping, node 2(3) cannot establish

a pairwise key with node 1(4). The second step is to show that node 2(3) is successful in establishing a pairwise key with node 1(4) after the secret information swapping completes between node 2 and node 3. In this demonstration, LEDs on the sensor nodes are used to keep track of the key establishment and attack activities. The green, yellow, and red LEDs are associated with the hello message receipt activity, the key swapping activity, and the pairwise key establishment activity respectively. If a LED is lit up, the corresponding activity is successful, otherwise it fails. The hardware devices used in this demonstration are four MICAz motes from Crossbow Technology as shown in Figure 5.6.

Figure 5.6: The hardware devices used in the demonstration

For the first demonstration step, after node 2(3) and node 1(4) are turned on, they first exchange the hello message with each other and then try to establish a pairwise key following the random-pairwise keys scheme. As shown in Figure 5.7, although node 2(3) and node 1(4) succeeded in exchanging the hello message but failed in their attempt to agree on a common key between themselves. The second demonstration step is illustrated in Figure 5.8. After nodes 2 and 3 are switched on, they first execute the code in CtrlledNode to swap their IDs and key rings with each other using a unique type of message: SwapKeyMsg. The completion of the key swapping process is indicated by the yellow LEDs on the nodes.

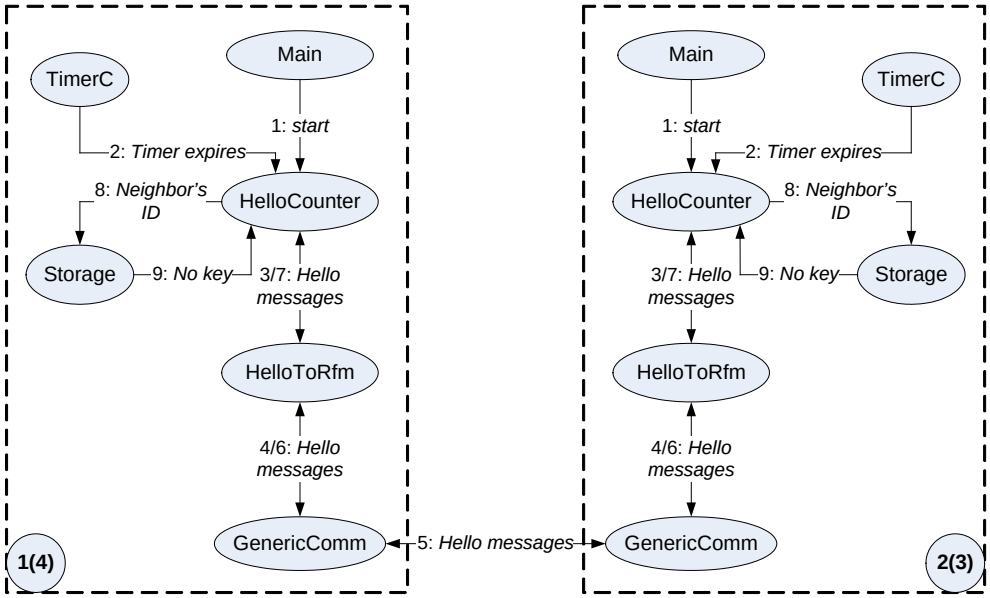


Figure 5.7: The completion of the hello message exchange between nodes 1(4) and 2(3) with the failure of key establishment

Then, they place the received IDs and key rings in Storage. This activity is followed by the normal key establishment process. Specifically, nodes 1 and 2, for example, execute the code in RandPKS by broadcasting the hello message. When they finish receiving the hello message, their green LEDs are turned on. The success of the nodes in establishing a pairwise key is signalled by having the red LEDs on the nodes.

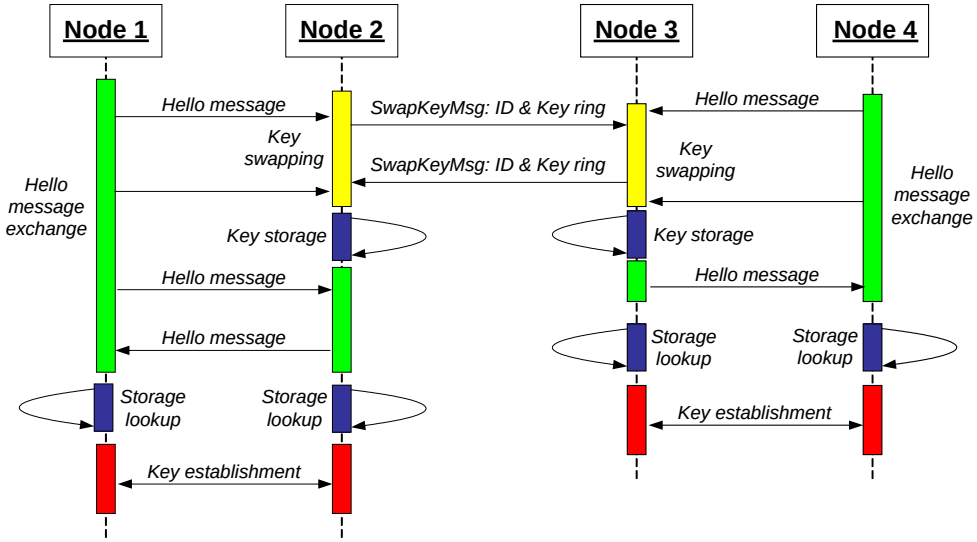


Figure 5.8: The success of short-distance swapping CLEA

5.4 Chapter Remark

This chapter is aimed at demonstrating that CLEA indeed poses a real security threat to DSNs. CLEA is more than likely to be launched due to several reasons. First, due to the unattended nature of DSNs and unshielded nature of sensor nodes, the attacker can capture and compromise sensor nodes with ease. Furthermore, research has shown that even anti-tamper techniques are not sufficient to keep secret information from being extracted. Second, the attacker can increase the number of controlled links by using the same compromised nodes and secret information repetitively without being detected. The methods available are secret information cloning, short-distance swapping, long-distance swapping, or mixed-distance swapping. Finally, a prototype of CLEA using the short-distance swapping method has been implemented successfully on MICAz.

Chapter 6

Secret Information Protection Based Countermeasures

This chapter consists of three sections presenting a series of three schemes against CLEA. Each section starts with a brief overview of the scheme itself followed by its detailed description. Thereafter, an in-depth security and performance analysis is carried out to show the plausibility of the scheme. Potential shortcomings are investigated at the end of each section and then used as a base for the development of the following schemes. The chapter ends with a system implementation of the final scheme to show its practicality for the current sensor node generation.

The proposed schemes are independent of the key establishment schemes. They can be used for defending any key establishment schemes for DSNs against CLEA. However, for the sake of presentation, let us consider the random pairwise keys scheme [28] as a concrete example throughout this thesis.

6.1 OWHCBS: One-Way Hash Chain Based Scheme

6.1.1 Overview

The main idea of our proposal is to minimise the utility of compromised nodes or secret information to attackers. It is enabled via the use of two factors: generation based deployment model and one-way hash chain (OWHC) [48] as illustrated in Figure 6.1. The generation based deployment model is derived naturally from the inherent nature of sensor network deployment. That is, in general, the sensor node deployment happens more than once throughout the course of the network lifetime. Hence, it is reasonable to classify the total sensor nodes intended for deployment into generations indexed from 1

to t based on a number of deployment times. At one deployment moment, one generation is deployed complying with application requirements. Besides, a one-way hash chain is generated from a seed K_t. In fact, this OWHC ($K_0,...,K_t$) is a collection of values such that each value K_v (except the last value K_t) is a one-way function of the next value K_{v+1}. That is, we have $K_v = H(K_{v+1})$, for $0 \le v < t$, where H is a one-way function and often selected as a cryptographic hash function such as MD5 [62] or SHA-1 [63]. To generate the chain, the security server selects at random K_t as the seed of the chain and then derives all previous values by repeatedly applying the hash function as mentioned above. Element i (K_i) of the chain is associated with sensor generation i functioning as a key encryption key (KEK). This generation-wide KEK is stored in each node together with its key ring. Upon deployment, sensor nodes follow a key establishment scheme, e.g. random pairwise scheme in [28] to establish pairwise keys. After all key setup activities have been completed, an ith generation node then performs the following operations: key ring encryption using K_i, calculation of key recovery key (KRK) K_{i-1}, replacement of K_i and key ring with K_{i-1} and encrypted key ring respectively.

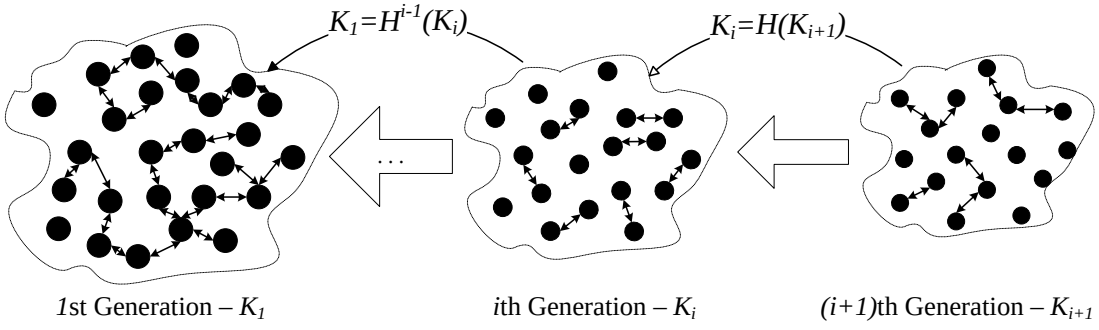


Figure 6.1: An example of the deployment model

The advantage of this approach is twofold. First, it still allows predecessor generations to integrate with successor generations upon deployment. Second, it minimizes both the attacker's opportunity to obtain secret information and the quantity of disclosed pairwise keys from compromised nodes.

6.1.2 Detailed Description of OWHCBS

OWHCBS consists of three phases namely *system initialization*, *first generation deployment* and *successive generation deployment* as follows.

6.1.2.1 System Initialization Phase

Assume that during the system lifetime of the sensor network, there are *t* times of sensor node deployment (or addition) corresponding to *t* generations. For each sensor node $S_{i,j}$, the following information is pre-distributed.

- m keys in the key ring: $K_{ij,1},...,K_{ij,m}$ together with IDs of the other nodes that also know the keys. The value of m and the keys are chosen as discussed in [28].
- A generation-wide KEK from the one-way hash chain $K_j = H^{t-j}(K_t)$.
- A generation ID $GenID_j$, satisfying the condition: $GenID_{j+1} = GenID_j + 1$.

6.1.2.2 First Generation Deployment Phase

In this phase, each senor node $S_{i,1}$ needs to perform the following steps in succession.

1. *Step 1:* $S_{i,1}$ broadcasts its ID, $GenID_1$ and receive its neighbors' IDs and generation IDs. $S_{i,1}$ uses the pre-loaded keys to establish pairwise keys with its neighbors as detailed in [28].
2. *Step 2:* After the pairwise key establishment, $S_{i,1}$ encrypts the key ring using the generation key $E_{S_{i,1}} = E_{K_1}(K_{i1,1} \| K_{i1,2} \| ... \| K_{i1,m})$. The keys in the ring should be arranged in accordance with the order of the node IDs for ease of restoring the keys.
3. *Step 3:* $S_{i,1}$ computes the key chain commitment or the KRK of K_1 : $K_0 = H(K_1)$, deletes K_1 and the pre-loaded keys. Thus, what is remaining in $S_{i,1}$'s memory storage are $GenID_1$, K_0, $E_{S_{i,1}}$, node IDs and established pairwise keys.

6.1.2.3 Successive Generation Deployment Phase

The aim of this phase is to guarantee network scalability while confining the attacker's power. Assume that the *q*th sensor node generation has just been sprinkled over the terrain, then a sensor node $S_{i,q}$ activates the key establishment process with a neighbor $S_{j,g}$ by following the steps below.

1. *Step 1:* $S_{i,q}$ broadcasts its ID, $GenID_q$ and receives $S_{j,g}$'s ID and $GenID_g$.
2. *Step 2:* If two nodes discover that they have a pre-loaded key in common, then they check whether they are in the same generation by comparing $GenID_g$ and $GenID_q$. If they are identical, step 3 will be executed otherwise step 4 will be executed.
3. *Step 3:* Two nodes follow the same procedure presented in the first generation deployment phase to accomplish the pairwise key establishment. Finally, $S_{i,q}$ has $GenID_q$, K_{q-1}, $E_{S_{i,q}}$ encrypted by K_q, node IDs and newly established pairwise keys in its memory.
4. *Step 4:* Up to this point, it is certainly claimed that $GenID_q > GenID_g$. $S_{i,q}$ computes $K_g = H^{GenID_q - GenID_g}(K_q), K_{g-1} = H(K_g)$ and sends $E_{K_{g-1}}(K_g)$ to $S_{j,g}$.
5. *Step 5:* $S_{j,g}$ uses K_{g-1} to decrypt $E_{K_{g-1}}(K_g)$ and obtain the KEK K_g. $S_{j,g}$ uses K_g to restore the original key ring by decrypting $E_{S_{i,g}}$: $K_{ig,1} \| K_{ig,2} \| ... \| K_{ig,m} = D_{K_g}(E_{S_{i,g}})$. This key ring is then used for the pairwise key establishment.

At the end of this phase, $S_{j,g}$ re-encrypts the pe-loaded keys in the key ring using $K_g : E_{S_{i,g}} = E_{K_g}(K_{ig,1} \| K_{ig,2} \| ... \| K_{ig,m})$, then deletes K_g and the pre-loaded keys. $S_{j,g}$ only retains $GenID_g$, K_{g-1}, $E_{S_{i,g}}$, node IDs and established pairwise keys.

6.1.3 Security Analysis

To evaluate OWHCBS in terms of security, a detailed analysis of a network setting of $n = 1000$, $n' = 50$, $p = 0.3$ under the class I node compromise model is carried out. Before this analysis, let us consider two terminologies: *usable* pairwise key and *CLEA-able* pairwise key. The former is used where no CLEA occurs whereas the latter is considered in CLEA context in which a single controlled node can present multiple compromised IDs and use multiple compromised pairwise keys.

The analysis is conducted based on the following metrics: (i) When nodes belonging to generations from 1 to x are compromised, what is the number of CLEA-able pairwise keys available to controlled nodes? This metric quantifies the cascading impact of CLEA. (ii) Given that b nodes from w generations (1 to w) are compromised, what is the probability that the hth generation is vulnerable? The hth generation is considered to be vulnerable if at least one node of their descendent generations is compromised. The reason is that by having captured the descendent generation node, the attacker directly or indirectly obtains K_h and backward KEKs indirectly. Thus, this analysis examines the possible maximum benefit the attacker can gain from a compromised node.

6.1.3.1 Cascading Impact of CLEA

This section highlights the efficiency of the proposed scheme by comparing the number of available CLEA-able or usable (as for no collusion) pairwise keys between three scenarios of the controlled nodes' interaction: no CLEA, CLEA without OWHCBS, CLEA under OWHCBS.

In the first scenario, the attacker gains additional $p \times n'$ usable pairwise keys for each newly compromised node on average. Therefore, the total number of usable pairwise keys can be estimated as $\left(\sum_{i=1}^{x} c_i \right) \times p \times n'$. In the second scenario, disregarding the key overlap, in addition to illegal communication with $p \times n'$ of its neighbours each controlled node can present $\left(\sum_{i=1}^{x} c_i \right) - 1$ legitimate identities to communicate with its

neighbours. Therefore, the total number of CLEA-able pairwise keys increases to $\left(\sum_{i=1}^{x} c_i\right)^2 \times p \times n'$. In the third scenario, the attacker can only gain pairwise keys from compromised nodes of the generations from 1 to $x-1$ except the xth generation ones since it does not know the KEK K_x. Moreover, the probability of a node belonging to the generations from 1 to $x-1$ is $p_b = \sum_{i=1}^{x-1} s_i \Big/ n$. Then it follows that the number of nodes of the generations from 1 to $x-1$ in a neighbourhood is $p_b \times n'$. Therefore, without collusion, each newly compromised node creates additional $p \times p_b \times n'$ usable pairwise keys on average. As a result, the total number of CLEA-able pairwise keys can be estimated using this formula $\left(\sum_{i=1}^{x-1} c_i\right)^2 \times p \times p_b \times n'$.

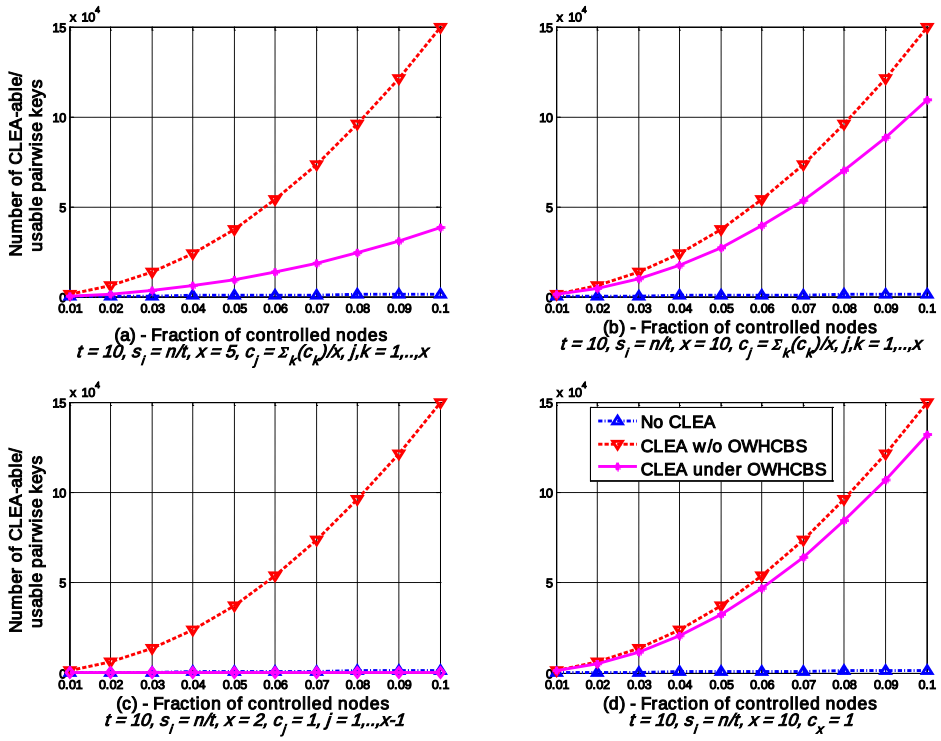


Figure 6.2: The impact of CLEA in the three scenarios

For better understanding, the relationship between the number of CLEA-able/usable pairwise keys and the fraction of compromised/controlled nodes in various example system settings is plotted in Figure 6.2. In these settings, there is a general assumption

that there have been $t = 10$ generations of equal population deployed thus far. Moreover, in one setting, the attacker is assumed to have equal possibility of capturing nodes of the generations from 1 to x (x = 5 or 10) as shown in Figures 6.2(a) and 6.2(b). In the other settings, the attacker is allowed to capture either most of the nodes as in Figure 6.2(c) or only one node as in Figure 6.2(d) of the xth generation. The aim is to evaluate the range of the attacker's gain under different settings.

Based on observation of the figures, the following remarks are drawn: (i) In a typical situation, under protection of OWHCBS, the number of CLEA-able pairwise keys available to the attacker significantly decreases and stays quite closer to that of the non-CLEA scenario. (ii) In the worst cases where there is at least one controlled node of the latest generation, OWHCBS still offers more resilience than the case without it as illustrated in Figures 6.2(b) and 6.2(d). (iii) In the most optimistic cases where most of the compromised nodes belong to the xth generation and the value of x is much smaller than that of t, our proposal even offers more resilience than the scenario of no CLEA which is what we desire to achieve. The argument can be inferred from Figure 6.2(c).

6.1.3.2 Probability of the *h*th Generation Being Vulnerable

Assume that at the time of b nodes having been compromised, there have been l deployed generations from 1 to l ($1 \le h < l$). h must be less than l since at that time none of the lth generation nodes can be compromised according to the class I node compromised model. Due to the one-way property of the KEK chain $K_0 \leftarrow K_1 \leftarrow ... \leftarrow K_l$, the hth generation is vulnerable if and only if at least one compromised node belongs to a φth generation where $h < \varphi < l$. Hence, the probability of the hth generation being vulnerable is

$$p_h = 1 - \left(\sum_{i=1}^{h} s_i \atop b \right) \Bigg/ \left(\sum_{i=1}^{l} s_i \atop b \right)$$

A particular instance is given in Figure 6.3 where $l = 10,\ s_i = s_j \left(i \neq j, i, j = \overline{1,l} \right),\ b = \overline{0, s_i},\ h = 2, 4, 6, 8,\ l - 1$. As indicated in the figure,

there is a trend that an ancestor generation is more likely vulnerable to CLEA than a successor generation.

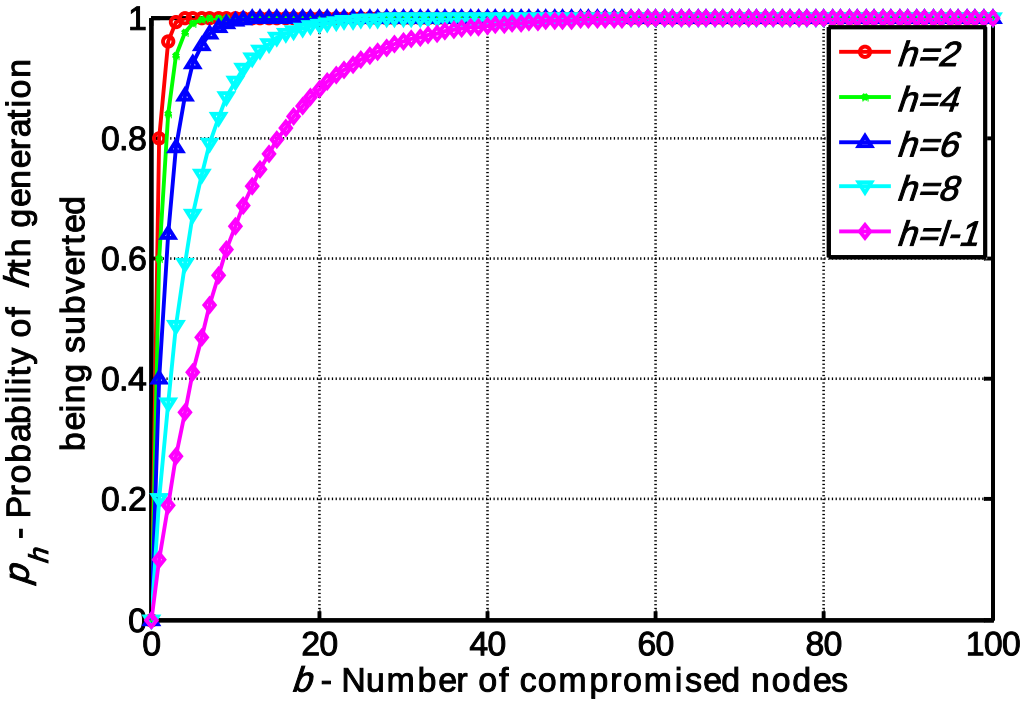


Figure 6.3: Probability of the hth generation being vulnerable

6.1.4 Performance Analysis

In general, improving security resilience of a proposal is accompanied by extra performance overheads. In this situation, these are extra overheads of memory storage, communication and computation which arise from adapting OWHCBS to the random pairwise keys scheme. This section aims to estimate them.

OWHCBS only requires small extra storage space for a KEK and a generation ID. Normally, the size of a KEK ranges from 64 to 128 bits and that of a generation ID is from 4 to 8 bits for a generation ID. Thus, the extra storage cost is negligible.

OWHCBS has very low communication overhead. If two nodes are in the same generation, only their generation IDs need to be sent additionally between themselves. These data can be piggybacked in data messages used to broadcast node IDs. Therefore, there is no additional communication cost in this situation. If two nodes are in different

generations, then only one additional message needs to be sent by only one communicating end. This overhead is quite small to be of concern.

As for computational overhead, there are two cases in consideration namely same generation key agreement and different generation key agreement. In the first case, the extra computational overhead of each deployment time emerges from one encryption and one hashing operation. In the other case, at each deployment time, the scheme costs each ancestor node a total of two decryptions and one encryption. It also costs each descendant node several hashing operations, and two encryptions. These are symmetric cryptographic primitives which have been proved to have least computational complexity and consume the least energy. As a concrete illustration, the energy consumption of cryptographic primitives in each of three types of nodes namely same generation nodes, ancestor nodes, and descendant nodes is quantified based on the following parameters:

- $m = 300$; 64-bit, 80-bit, or 128-bit pre-loaded pairwise keys; 80-bit KEK.
- Encryption/decryption algorithm: RC5; hash algorithm: SHA-1. Expected number of hashing operations is 5.
- Energy costs of encryption/decryption and hash algorithms are extrapolated from figures in [14].

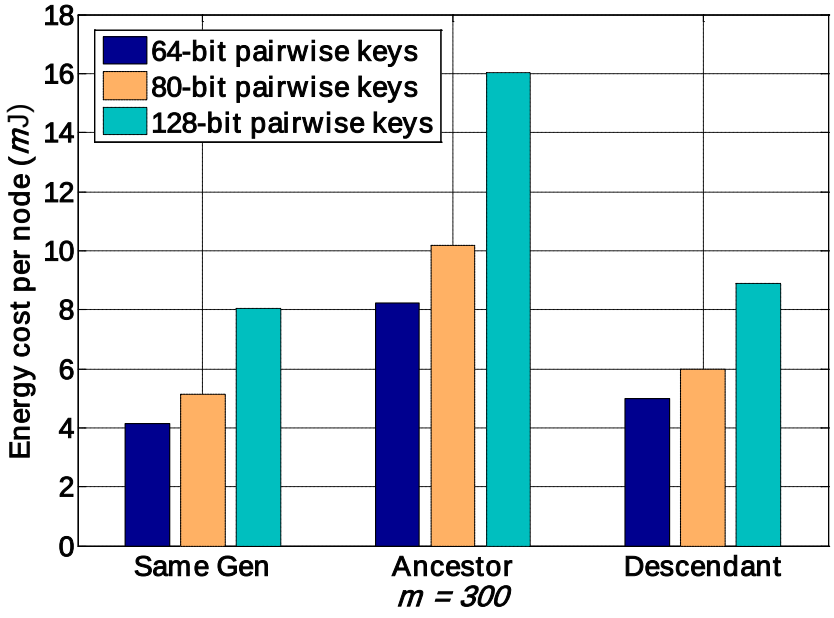


Figure 6.4: Cryptographic energy consumption

As shown in Figure 6.4, the energy consumption of cryptographic operations are very reasonable where the same generation nodes consume the least energy with an average of 6 *mJ* whereas the ancestors' energy consumption is highest averaging about 12 *mJ*. Overall, OWHCBS could be well-afforded by the energy resources of the current generation sensor nodes.

6.1.5 Limitations

As examined closely, the OWHCBS is exposed to two limitations which are summarised as follows. Firstly, as analysed in 6.1.3.2, the scheme fails to guarantee the backward confidentiality for the KEKs. The reason is that, having captured and compromised a sensor node from the *i*th generation, the attacker obtains the KRK of the *i*th KEK which is the KEK of the $(i-1)$th generation as well. The property of the OWHC allows the attacker to compute backward KEKs of the generations from 1 to $i-2$ using the $(i-1)$th generation KEK. These obtained KEKs allow the attacker to recover the plaintext secret information of compromised nodes. Using the disclosed information and KEKs, the attacker can launch CLEA among nodes from generations 1 to $i-1$ with ease.

Secondly, the threat of CLEA with the aid of the continuous-time attack has not been considered in the security analysis of the scheme. A detailed analysis of this situation is necessary for the proper evaluation of the scheme.

6.2 DOWHCBS: Diversified OWHC Based Scheme

6.2.1 Overview

Overall, the key idea of this scheme is relatively similar to that of OWHCBS. However, to equip the KEKs with the backward confidentiality, DOWHCBS employs a diversified one-way hash chain instead of the simple one-way hash chain in OWHCBS. More specifically, each element of an *j*th generation node $S_{i,j}$'s key ring is encrypted using K_j which is the *j*th element of the hash chain and is associated with the *j*th

generation of sensor nodes. The generation-wide K_j is pre-distributed in each node together with the node's K_j -encrypted key ring. Upon deployment, sensor nodes decrypt related encrypted keys using K_j and follow the description in [28] to establish pairwise keys. After the key establishment activity has been completed, $S_{i,j}$ then performs the following operations:

- Deletion of the used encrypted keys.
- KEK diversification – The object of this operation is to provide the backward confidentiality for the KEKs. Accordingly, the node generates a salt $r_{i,j}$ and calculates the key recovery key $K_{i,j-1} = H_k\left(K_j \,||\, r_{i,j}\right)$. After that, the node deletes K_j and retains $r_{i,j}$ and K_{j-1}. The KEK diversification is illustrated in Figure 6.5.

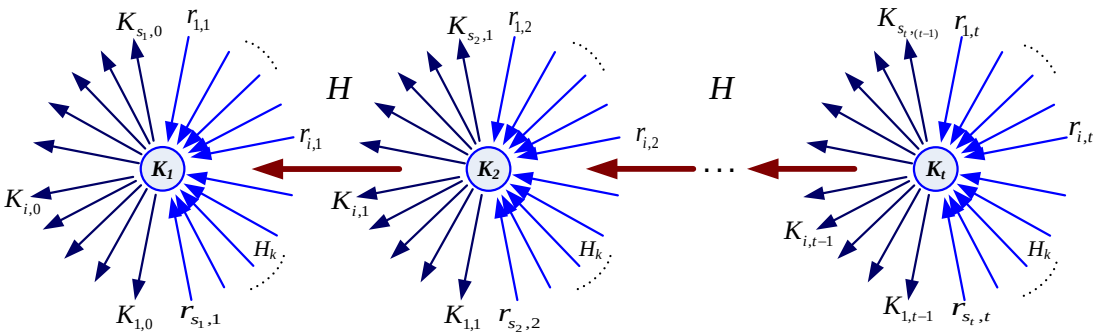


Figure 6.5: Illustration of the KEK diversification

6.2.2 Detailed Description of DOWHCBS

For the sake of completeness, the entire DOWHCBS is described even though part of it has been presented in OWHCBS.

6.2.2.1 System Initialization Phase

Assume that during the lifetime of the sensor network, there are t times of sensor node deployments (or additions). Hence, all of the sensor nodes intended to be deployed are

categorised into generations indexed from 1 to t. For each sensor node $S_{i,j}$, the following information is pre-distributed:

- Generation ID: $GenID_j$ is created such that $GenID_j = GenID_{j-1} + 1$.
- Generation-wide KEK from the one-way hash chain: $K_j = H^{t-j}(K_t)$.
- m K_j-encrypted keys in the key ring $e_{S_{i,j}}$ together with the corresponding IDs of other nodes that also know the keys. The value of m and the keys are chosen as discussed in [28].

6.2.2.2 First Generation Deployment Phase

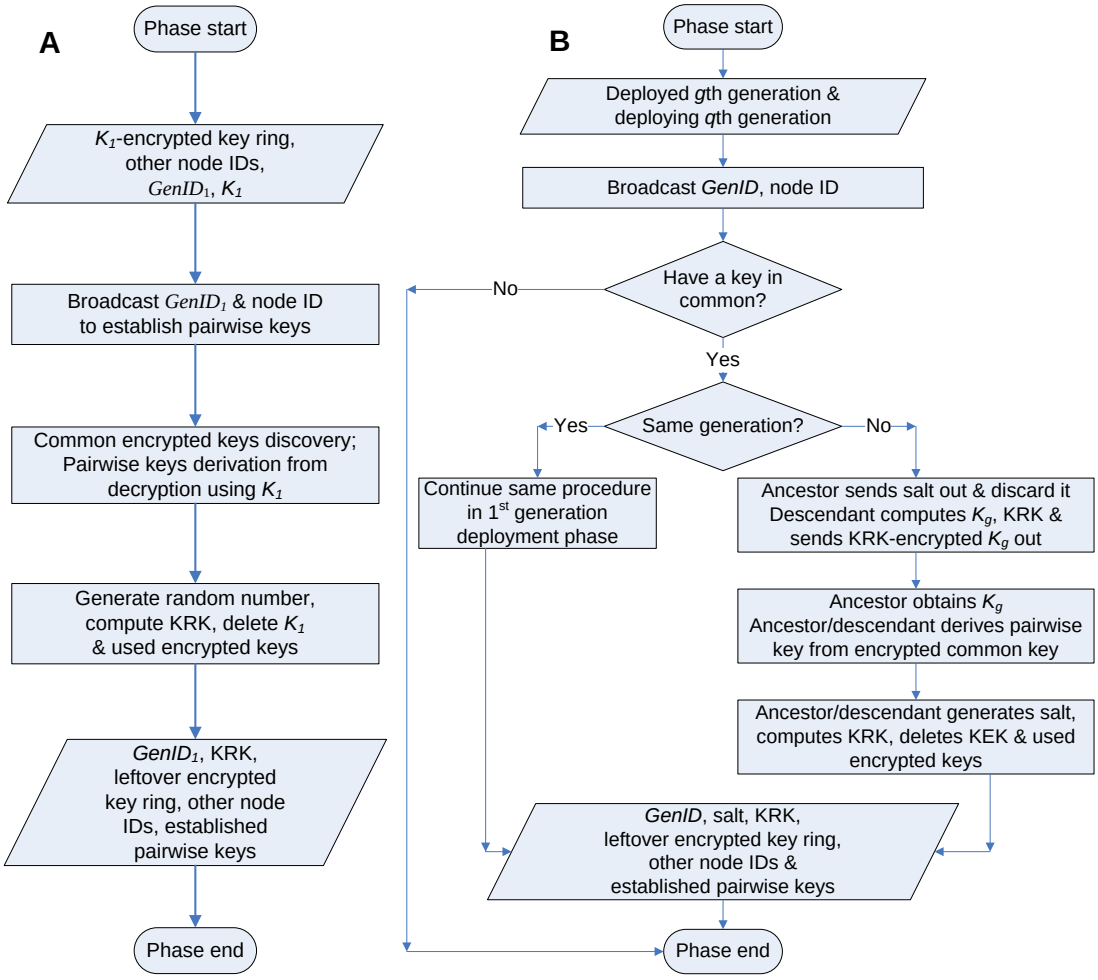


Figure 6.6: (a) First generation deployment phase flow chart
(b) Generation addition phase flow chart

In order to complete this phase, each senor node $S_{i,1}$ needs to perform the following steps illustrated in Figure 6.6a in succession.

1. *Step 1:* $S_{i,1}$ broadcasts its ID, $GenID_1$ and receives its neighbours' IDs and generation IDs.
2. *Step 2:* $S_{i,1}$ uses the received IDs to find out the common encrypted keys between itself and its neighbours as detailed in [28]. Pairwise keys are derived from the decryption of the common encrypted keys using K_1.
3. *Step 3:* $S_{i,1}$ selects a salt $r_{i,1}$, computes the KRK $K_{i,0} = H_k(K_1 \| r_{i,1})$, and then deletes K_1 and the used encrypted keys. Thus, what remain in $S_{i,1}$'s memory storage are: $GenID_1$, $r_{i,1}$, $K_{i,0}$, $e_{S_{i,1}1}$, the other node IDs and the established pairwise keys.

6.2.2.3 Successive Generation Deployment Phase

This phase is executed when sensor nodes of later generations are added to the network in such a manner that the key establishment process is immune from CLEA. The following detail how sensor nodes behave to achieve this aim.

Assume that the qth sensor node generation has just been sprinkled over the terrain, and then a sensor node $S_{i,q}$ activates the key establishment process with a neighbour $S_{j,g}$ by following the steps illustrated in Figure 6.6b.

1. *Step 1:* $S_{i,q}$ broadcasts its ID, $GenID_q$ and receives $S_{j,g}$'s ID and $GenID_g$.
2. *Step 2:* If two nodes discover that they have a pre-loaded encrypted key in common, then they check whether they are in the same generation by comparing $GenID_g$ and $GenID_q$. If they are identical, step 3 will be executed otherwise step 4 will be executed.
3. *Step 3:* Two nodes continue the same procedure presented in the first generation deployment phase to accomplish the pairwise key establishment. Finally, $S_{i,q}$ has $GenID_q$, $r_{i,q}$, $K_{i,q-1}$, $e_{S_{i,q}q}$, the other node IDs and newly established pairwise keys in its memory.

4. *Step 4:* Up to this point, it is asserted that $GenID_q > GenID_g$. First, $S_{j,g}$ sends $r_{j,g}$ to $S_{i,q}$ and then discard $r_{j,g}$ while $S_{i,q}$ computes $K_g = H^{GenID_q - GenID_g}(K_q)$. Second, $S_{i,q}$ computes KRK $K_{j,g-1} = H(K_g \| r_{j,g})$ and sends $E_{K_{j,g-1}}(K_g)$ to $S_{j,g}$.
5. *Step 5:* $S_{j,g}$ uses $K_{j,g-1}$ to decrypt $E_{K_{j,g-1}}(K_g)$ and obtain the KEK K_g. Then, $S_{j,g}$ ($S_{i,q}$) decrypts the encrypted key discovered in the step 2 using K_g (K_q) to derive the pairwise key.
6. *Step 6:* $S_{j,g}$ ($S_{i,q}$) computes another KRK of K_g (K_q). To do this, $S_{j,g}$ ($S_{i,q}$) generates a salt $r'_{j,g}$ ($r'_{i,q}$), $r'_{j,g} \neq r_{j,g}$ and calculates $K'_{j,g-1} = H(K_g \| r'_{j,g})$ ($K_{i,q-1} = H(K_q \| r'_{i,q})$). Finally, $S_{j,g}$, for example, deletes K_g and the used encrypted keys and only retains $GenID_g$, $r'_{j,g}$, $K'_{j,g-1}$, $e_{S_{j,g}q}$, the other node IDs and newly established pairwise keys.

6.2.3 Security Analysis

6.2.3.1 Analytical Analysis

In this analysis, DOWHCBS is analytically investigated taking the two attack models presented in Chapter 2 into consideration.

1) Sporadic attack: In this model, the notion of the short secure time interval Y_s is applied to not only the first generation but also to successive generations as mentioned in [10], [11], [12]. In other words, the sporadic attacker is assumed to only have the capability to compromise nodes after each key establishment process. Therefore, the attacker can not obtain any secret information existing in that process. The following lemma and theorems show the security attributes of the scheme.

Lemma 1: Under the sporadic attack, DOWHCBS guarantees that no KEK is disclosed when sensor nodes are compromised.

Proof: After time interval Y_s, sensor nodes only retain the following information in their memory: generation IDs, salts, KRKs, leftover encrypted key rings, other node IDs and

newly established pairwise keys. By capturing sensor nodes of this stage, the sporadic attacker can obtain all the above information, especially salts and KRKs. However, due to the one-way property of the hash function, the attacker can not restore KEKs from the salts and KRKs. ■

Theorem 1: DOWHCBS provides backward KEK confidentiality under the sporadic attack.
Proof: As proved in Lemma 1, no KEK can be restored under the sporadic attack. Hence, the hash function can not be applied to obtain the KEKs of the previous generations. ■

Theorem 2: DOWHCBS eradicates CLEA under the sporadic attack.

Proof: From Lemma 1 and Theorem 1, it follows that the attacker can not restore plaintext key rings from captured encrypted key rings. Therefore, the attacker gains no secret information and then CLEA can not be launched. ■

Based on the above theorems, it is certainly claimed that DOWHCBS provides key establishment schemes with complete immunity from CLEA.

2) Continuous-time attack: In the previous attack model, despite the fact that the value of Y_s after each deployment time is very small (less than 60 seconds estimated from [53]), the given assumption is not practical to a certain extent. For example, the node compromise attack has been occurring on a deployment site while a sensor generation is being deployed. In this case, it is likely that the attacker might compromise newly deployed sensor nodes within Y_v and thus obtain KEKs. Using the obtained KEKs, the attacker can get access to plaintext key rings to mount CLEA.

In order to investigate this attack model, the following assumptions about the generation deployment model and the node capture pattern are made:

- The interested attack period illustrated in Figure 6.7 which contains Y_vs is divided into t intervals of equal length T. The period ends at the time T_t. Each interval consists of δ equal time slots. The value of δ is determined based on

the time to compromise a sensor node. It can be around 1 minute according to [44]. The generation deployment occurs at the beginning of each interval.

- The population of generations follows the geometric distribution with parameter λ_1 except the last generation as specified by s_g as follows:

$$s_g \approx \begin{cases} \lceil n \times (1-\lambda_1)^{g-1} \times \lambda_1 \rceil, 1 \le g \le t-1 \\ n - \sum_{i=1}^{t-1} s_i, \qquad g = t \end{cases}, 0 \le \lambda_1 \le 1 \tag{6.1}$$

- Node capture/compromise occurs according to a Poisson process with a rate λ_2.

Given the above mentioned assumptions, the following lemmas and theorem exhibit the security property of DOWHCBS under the continuous-time attack.

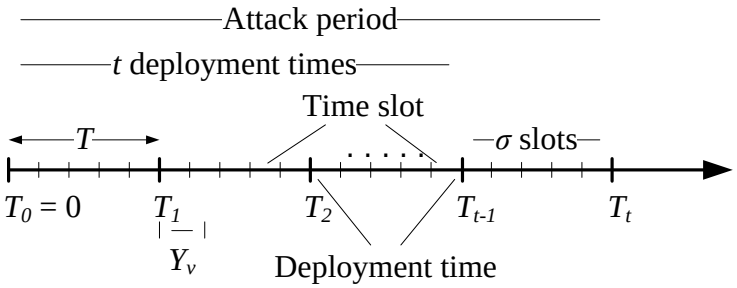


Figure 6.7: Illustration of the interested attack period

Lemma 2: The probability of at least one sensor node being compromised over one interval Y_v is:

$$p_c = 1 - e^{-\lambda_2 \times Y_v} \tag{6.2}$$

Proof: Using Poisson process theory, the probability of no sensor node being captured/compromised over one interval Y_v is:

$$p_0 = \frac{(\lambda_2 \times Y_v)^0 e^{-\lambda_2 \times Y_v}}{0!} = e^{-\lambda_2 \times Y_v}$$

Hence, $p_c = 1 - p_0 = 1 - e^{-\lambda_2 \times Y_v}$. ■

Lemma 3: The probability of at least one sensor node of the *i*th generation being captured/compromised over the interval Y_v is:

$$p_{ci} = \frac{s_i}{\sum_{k=1}^{i} s_k} \times p_c \tag{6.3}$$

Proof: On the deployment of i deployed generations, the probability of a node belonging to the *i*th generation is: $\frac{s_i}{\sum_{k=1}^{i} s_k}$. From this probability and Lemma 2, Lemma 3 follows. ■

As indicated in Figure 6.8, the trend is that the younger a generation is, the less likely its sensor nodes are to be captured/compromised over the interval Y_v after deployment. In other words, the earlier generations except the first generation are more likely to be vulnerable than the later generations. This is the design goal of the scheme since when a generation is less likely to be vulnerable, it follows that earlier generations are also less vulnerable.

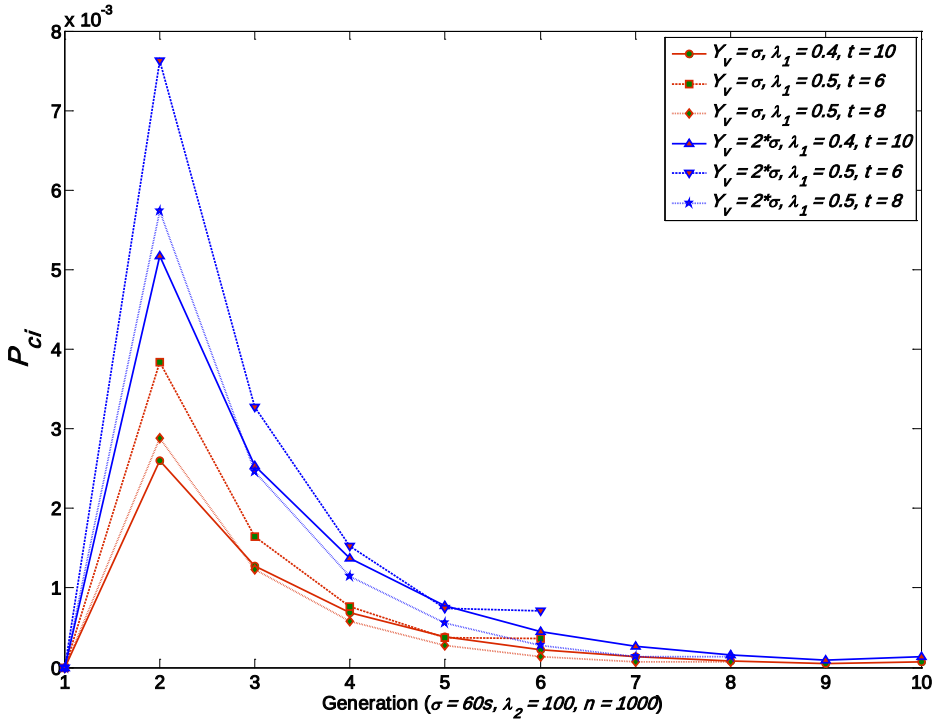


Figure 6.8: Probability of at least one sensor node of a generation being captured/compromised over the interval Y_v

Lemma 4: The expected number of *i*th ($i \geq 2$) generation nodes being captured/compromised within the interval Y_v is:

$$c_i = \frac{s_i}{\sum_{k=1}^{i} s_k} \times p_c \times \lambda_2 \times (T_{i-1} + Y_v) \tag{6.4}$$

Proof: In a Poisson process, the expected number of captured/compromised nodes over the interval (0, T_i + Y_v) is: $\lambda_2 \times (T_{i-1} + Y_v)$. From this formula and Lemma 3, Lemma 4 follows. ■

Theorem 3: The expected number of captured/compromised nodes within intervals Y_v from generation 1 to generation *i* is:

$$c_{1 \to i} = \sum_{j=2}^{i} \frac{s_j}{\sum_{k=1}^{j} s_k} \times p_c \times \lambda_2 \times (T_{j-1} + Y_v) \tag{6.5}$$

Proof: From Lemma 4, Theorem 3 follows. ■

Theorem 4: CLEA with the aid of the continuous-time attack under the protection of DOWHCBS exhibits the following properties assuming that the attacker can compromise at least one sensor node of the *i*th generation and cannot compromise any sensor nodes of the generations from *i*+1 to *t* over the interval Y_v:

(i) The attacker can launch CLEA among sensor nodes of the generations from 1 to *i*.

(ii) The attacker can only launch CLEA with sensor nodes of the generations from *i*+1 to *t* during the interval Y_v right after their generation deployment times. After this interval, the attacker is no longer able to mount the attack.

Proof: To prove (i), notice that the attacker can compromise a sensor node of the *i*th generation over the interval Y_v, consequently, it knows the KEK K_i and can compute the KEKs K_js, $j = \overline{1, i-1}$ by applying multiple hashing operations of the K_i. As a result, the attacker can compute the needed KRKs for each sensor nodes of these generations and then mount CLEA.

To prove (ii), notice that after each generation deployment, each of the KEKs K_js, $j = \overline{i+1,t}$ is only available for the short interval Y_v before being diversified to become KRKs in each sensor node. The attacker can not derive these KRKs if it only knows the KEK K_i. Based on this argument, Theorem 4 follows. ■

6.2.3.2 Further Security Discussion

The impact of CLEA can be mostly reduced if the security flaw of the first property of Theorem 4 can be overcome. Note that a controlled node of the *i*th generation needs to re-broadcast its ID and generation ID if it wants to launch CLEA after the *i*th key establishment process has finished. The purpose of this node is to deliberately mislead attacked nodes into believing that there is another generation deployment time and thus another ongoing key establishment process. Therefore, if the attacked nodes can keep track of the generation deployment, they are not only no longer misled into establishing illegitimate links but also able to detect the controlled node. One straightforward approach might be to have sensor nodes storing the generation ID of the latest deployed generation. Whenever a new generation is about to be deployed, an authenticated broadcast mechanism as mentioned in Chapter 4 can be utilised to inform the entire network about the generation ID of the upcoming generation. Once the legitimate nodes know this generation ID authentically, any attempt to mount CLEA after the key establishment process will fail and be detected.

6.2.3.3 Simulation Analysis

This simulation studies network setting of $n = 1000$ nodes with average density of $n' = 50$ sensor nodes in a neighbourhood and the probability $p = 0.3$ of a pairwise key shared between two nodes. It is assumed that the entire network is deployed in ten days. The generations are deployed on successive days. This means $t = 10$ and $T = 1$ day. The node capture rate receives different values $\lambda_2 = 100, 200, 300$ over the period of t days. KEKs are assumed to be obtainable via the capture of newly deployed sensor nodes during the interval $Y_v = 1$ or 2 minutes. The whole attack period is divided into multiple time intervals of length Y_v. During each of these intervals, at most one node is assumed

to be captured. The simulation is repeated 10000 times and the result shown in Figure 6.9 reports the average values.

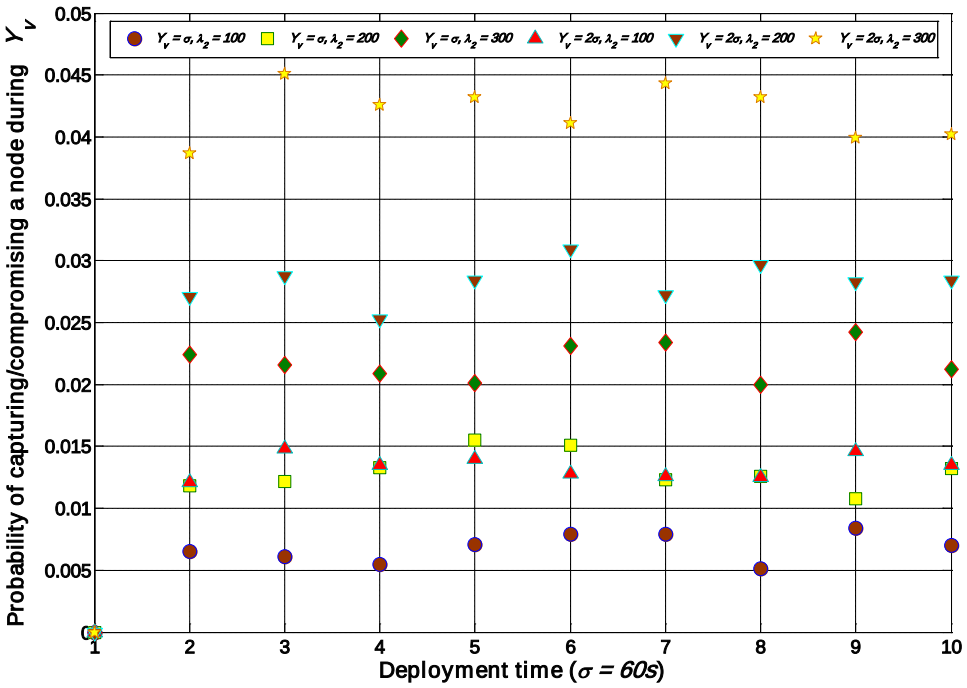


Figure 6.9: Probability of capturing a node during the interval Y_v after deployment times

According to Figure 6.9, the probability of node capture after the first deployment time is zero as assumed in Chapter 2. The other probabilities are distributed quite equally and increase when Y_v and/or λ_2 increases. However, these probabilities which are all less than 0.05 are too small to be a cause for concern. For example, when Y_v and λ_2 are equal to 120s and 100 nodes, respectively, the probability of a sensor node being captured/compromised during 120s after the 10^{th} generation deployment time is only an average of 0.0134. Note that this probability is equal to any node from first to 10^{th} generations. As a result, the probability of the 10^{th} generation KEK being compromised is even much smaller.

6.2.4 Performance Evaluation

In comparison to the random pairwise keys scheme, DOWHCBS only requires small extra storage space for a generation ID, a salt, and a KRK. Normally, the size of the salt

and KRK range from 64 bits to 128 bits and that of a generation ID is from 3 bits to 6 bits which can round up to 1 byte maximally. Thus, the extra storage cost is negligible. Table 6.1 below exemplifies this point.

Table 6.1: Comparison of storage requirements between two schemes

	***m* = 250** **64-bit keys**	***m* = 300** **64-bit keys**	***m* = 300** **128-bit keys**
Random pairwise keys scheme	2000 bytes	2400 bytes	4800 bytes
DOWHCBS-based scheme (Pre-PKE)	2009 bytes	2409 bytes	4817 bytes
DOWHCBS-based scheme (Post-PKE)	2017 bytes	2417 bytes	4833 bytes

DOWHCBS only adds very small extra communication overhead to the random pairwise keys scheme. If two nodes are in the same generation, only their generation IDs need to be sent additionally between themselves. These data can be combined with broadcast hello packets since the payload of each combined packet is only approximately 3 bytes, whereas the default packet payload in TinyOS, for example, is up to 28 bytes. Therefore, there is no additional communication cost in this situation. If two nodes are from different generations, then only one additional packet, containing either a salt or an encrypted version of a KEK (8 to 16 bytes each), needs to be sent by each communicating end. This overhead is too small to be of concern.

In terms of network-wide communication overhead, roughly speaking, each newly deployed sensor node needs to transmit one packet to sensor nodes of the same generation and two packets to its ancestors while deployed sensor nodes only have to transmit two packets. Hence, the upper bound of the network-wide communication cost is $3 \times n \times t$. While t is small, the network-wide overhead is quite low.

In order to justify the above observations, a network scenario as described in Section 6.2.3.3 is simulated. This simulation is repeated 100 times and the average values are

reported in Figure 6.10. It shows the communication overheads of each sensor node measured by the number of packets transmitted to its neighbours of the same generation, different generations and all generations with three different deployment scenarios specified by the set of parameters (λ_1, t). As can be seen, the communication overhead mostly comes from the different generation communication. This cost can be reduced by decreasing the value of t or the number of generations. For the entire network lifetime, DOWHCBS only cost each node about 62 packets.

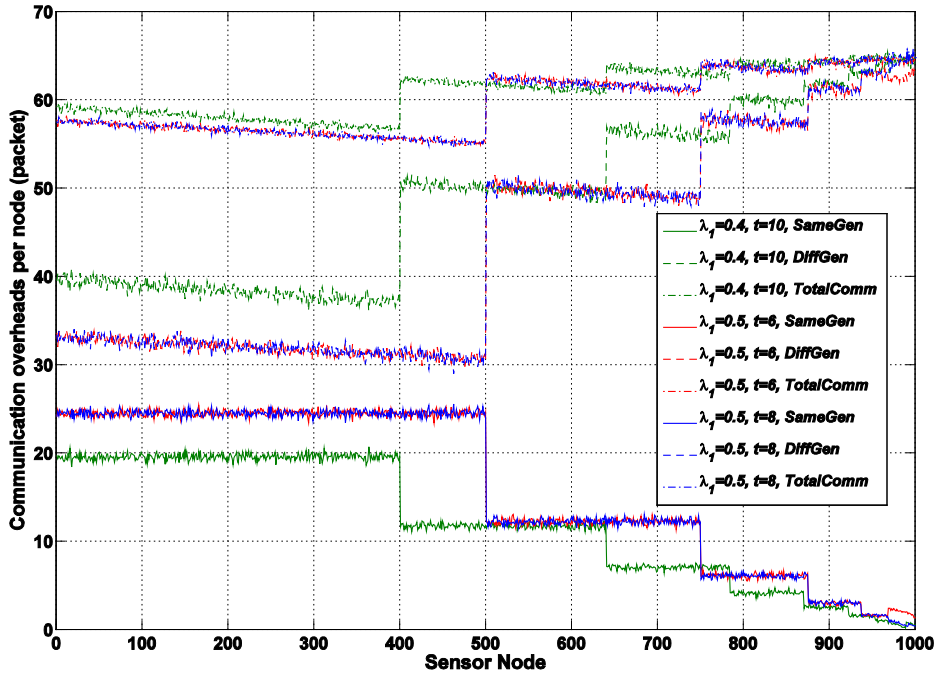


Figure 6.10: Communication overheads of each sensor node with various settings

The computational overhead is considered for two cases namely same generation key agreement and different generation key agreement. In the first case, the extra computational overhead of each deployment time emerges from one decryption and one hashing operation. In the other case, at each deployment time, DOWHCBS costs each ancestor node a total of two decryptions and one hashing operation. It also costs each descendant node several hashing operations, one decryption, and one encryption. These are symmetric cryptographic primitives which have been proved to have the least computational complexity and consume the least energy [27].

A simulation of a network setting as described in Section 6.2.3.3 is conducted to quantify the computational overhead of the scheme. The simulation was repeated 100 times to obtain the average number of encryption, decryption, hashing, and total security operations for each sensor node and the entire network with varied values of (λ_1, t). As depicted in Figure 6.11, the security primitive operations per sensor node vary from about 10 to 16 for encryption, 16 to 22 for decryption, 10 to 142 for hashing, and 40 to 178 for the total computation. These figures dominated by hashing operations increase as the number of generations grows. Therefore, the computational overhead is reduced if we can reduce the number of sensor generations.

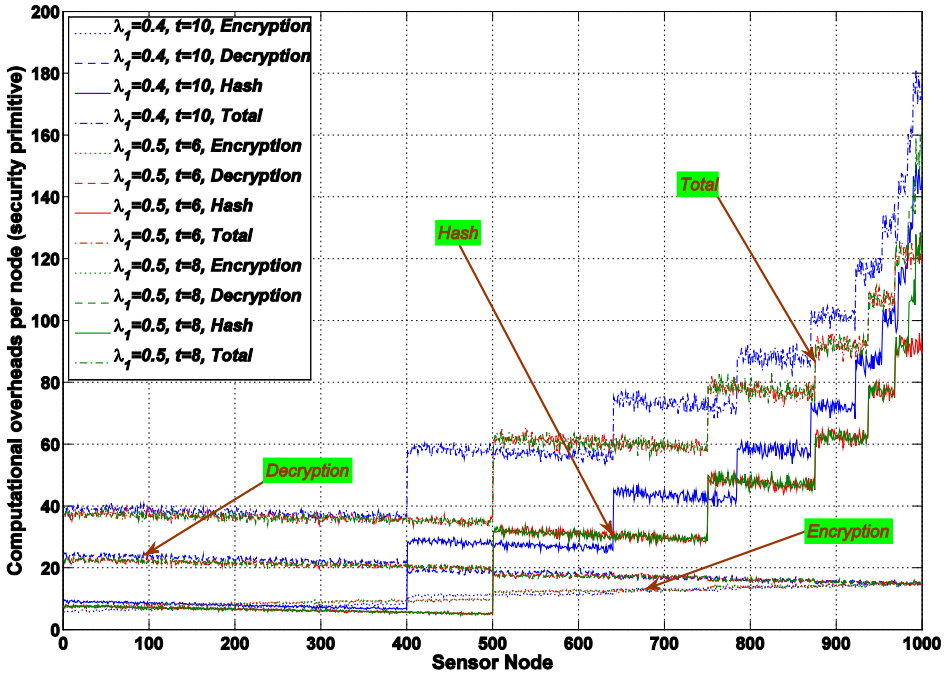


Figure 6.11: Computational overheads per sensor node with varied deployment scenarios

6.2.5 Limitations

DOWHCBS seems to be able to protect the KEKs, yet it is still susceptible to a more sophisticated attack. Suppose that nodes of the *i*th generation are joining the network and establishing keys; the attacker has compromised a node $S_{c,x}$ of generation x ($x < i$); one of the incoming nodes $S_{n,i}$ shares a key with $S_{c,x}$. In this context, provided

the attacker has compromised another node $S_{b,j}$ ($j < i$) in the neighbourhood of $S_{n,i}$, the following attack can be performed. The attacker forwards $S_{c,x}$'s ID and salt to $S_{b,j}$. $S_{b,j}$ then performs the steps in Section 6.2.2.3 to establish a pairwise shared key with $S_{n,i}$. But when it reaches step 5, $S_{b,j}$ learns K_g. This means the enhanced version also fails to support the backward confidentiality for the KEKs. The following section proposes the new countermeasure which can shield the keys used to protect the secret information from the disclosure.

6.3 HOWHCBS: Hidden OWHC Based Scheme

6.3.1 Overview

The approach used in this scheme is different from the previous two approaches in such a manner that the key used to protect the secret information of a sensor node is unique to that node. Therefore, the revelation of one key does not lead to the compromise of any other keys. To implement this idea, we use three types of keys: *secret encryption key* (SEK), *key encryption key* (KEK), and *key recovery key* (KRK) whereas only two key types were used in the previous approaches. The SEK, which have the same function as the KEK in the previous approaches, is unique to each node and used to protect secret information pre-distributed into that node by a given key establishment scheme. The KEK is an element of an OWHC generated from a seed K_t. It is unique to each node and is hidden by randomization in an exclusive-or operation with a random number before being used to protect the SEK. The other function of the KEK is to link different generations together. The hidden KEK and therefore the SEK are recovered with the aid of an incoming node and the KRK, which is the hash value of the KEK and another random number, during I_k to support the key establishment activities. After the key establishment, the plaintext SEK and hidden KEK are deleted; the ciphertext form of these keys are retained to avoid disclosure if the node were to be compromised. Because of the unavailability of these keys, the attacker is no longer able to recover plaintext secret information in order to launch CLEA.

6.3.2 Detailed Description of HOWHCBS

This section describes the technical detail on how HOWHCBS is used to protect a given key establishment scheme against CLEA. This countermeasure involves three types of operation namely *system setup*, *same generation key establishment*, and *different generation key establishment*.

6.3.2.1 System Setup

This operation is performed by the security server to initialise nodes by assigning the necessary information to them. As illustrated in Figure 6.12, the server first generates the OWHC from K_t. For each node $S_{u,v}$, the server generates two random numbers: $r_{u,v}$ and $f_{u,v}$ unique to $S_{u,v}$ in which $r_{u,v}$ has the same bitsize as K_t. Then, the server computes and loads the following information into $S_{u,v}$:

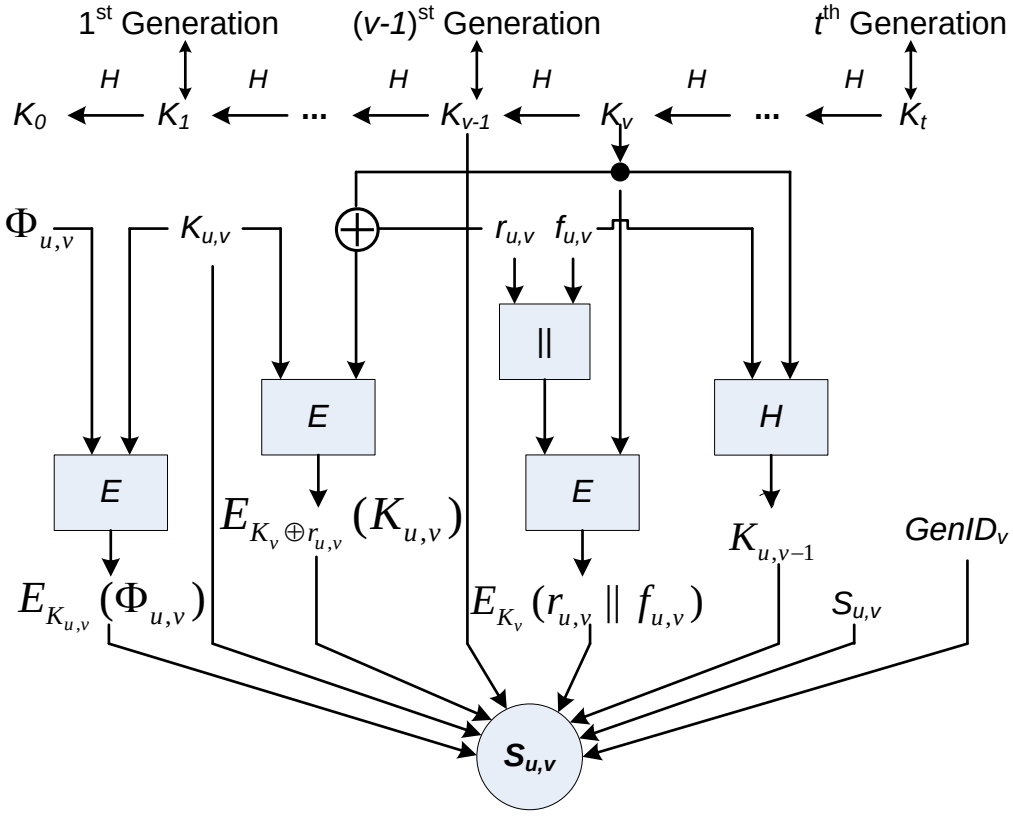


Figure 6.12: Illustration of node $S_{u,v}$'s initialisation and setup

- $S_{u,v}$ - Node ID.
- $GenID_v$ - Generation ID, satisfying $GenID_v = GenID_{v-1} + 1$.
- $K_{u,v}$ - A SEK, which is unique to $S_{u,v}$.
- K_{v-1} - A KEK, i.e., $K_{v-1} = H(K_v) = H^{t-v+1}(K_t)$.
- $E_{K_{u,v}}(\Phi_{u,v})$ - A set of encrypted secret information specified by a given key establishment scheme.
- $E_{K_v \oplus r_{u,v}}(K_{u,v})$, where $K_v = H^{t-v}(K_t)$.
- $K_{u,v-1} = H(K_v \| f_{u,v})$ - A KRK, which is unique because of the uniqueness of the random number $f_{u,v}$ to $S_{u,v}$.
- $E_{K_v}(r_{u,v} \| f_{u,v})$.

6.3.2.2 Same Generation Key Establishment

This type of operation occurs between any pair of newly deployed or added nodes $S_{x,d}$ and $S_{y,d}$, which are neighbors of each other and from the same generation. It consists of the following steps:

i) $S_{x,d}$ ($S_{y,d}$) broadcasts a HELLO message containing its ID, $GenID_d$, the key establishment-specific information.

ii) $S_{x,d}$ and $S_{y,d}$ know that they are from the same generation and use the key establishment-specific information to find out the common encrypted secret information between themselves. These secrets are then decrypted using $K_{x,d}$ ($K_{y,d}$) and used by the given key establishment scheme to establish a pairwise key.

iii) After the key establishment process ends, $S_{x,d}$ ($S_{y,d}$) removes $K_{x,d}$ ($K_{y,d}$) and K_{d-1}. Therefore, what are left in $S_{x,d}$'s memory, for example, are $S_{x,d}$, $GenID_d$, $E_{K_{x,d}}(\Phi_{x,d})$, $E_{K_d \oplus r_{x,d}}(K_{x,d})$, $K_{x,d-1}$, $E_{K_d}(r_{x,d} \| f_{x,d})$ and established pairwise keys.

6.3.2.3 Different Generation Key Establishment

This type of operation happens when a few nodes of a successor generation are just newly added into the network and want to establish pairwise keys with their neighboring nodes of ancestor generations. The following detail a scenario in which a successor node $S_{y,u}$ and an ancestor node $S_{x,d}$ ($u > d$) interact with each other as demonstrated in Figure 6.13 below:

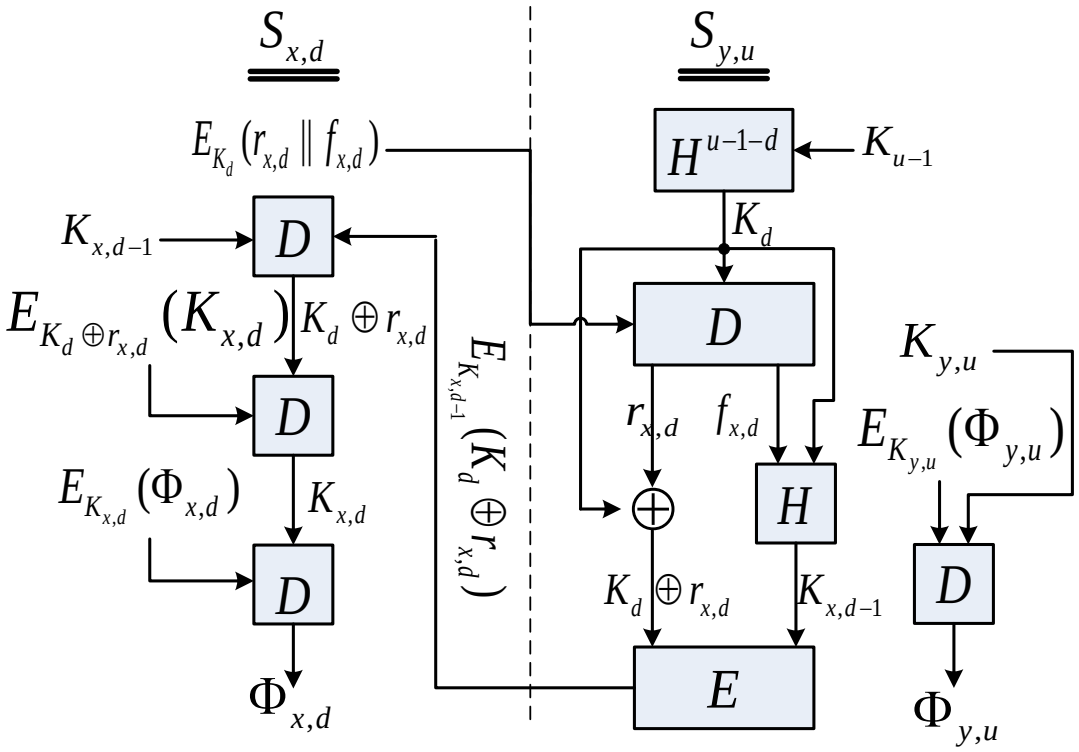


Figure 6.13: Different generation key establishment under HOWHCBS's protection

i) $S_{y,u}$ initiates the procedure by broadcasting its HELLO message and then receives a response from $S_{x,d}$.

ii) Upon receiving the HELLO message, $S_{x,d}$ and $S_{y,u}$ know that they are from different generations. Then, they check if they have any pre-distributed secret information in common. If no common secret is found, the procedure terminates. Otherwise the procedure continues as below.

iii) $S_{x,d}$ sends $E_{K_d}(r_{x,d} \| f_{x,d})$ to $S_{y,u}$ while $S_{y,u}$ computes $K_d = H^{u-1-d}(K_{u-1})$.

iv) Upon receiving $E_{K_d}(r_{x,d} \| f_{x,d})$, $S_{y,u}$ obtains $r_{x,d}$ and $f_{x,d}$ by decrypting $E_{K_d}(r_{x,d} \| f_{x,d})$. It then computes $S_{x,d}$'s KRK $K_{x,d-1} = H(K_d \| f_{x,d})$ and sends out $E_{K_{x,d-1}}(K_d \oplus r_{x,d})$.

iv) $S_{x,d}$ obtains the hidden KEK $K_d \oplus r_{x,d}$ by decrypting $E_{K_{x,d-1}}(K_d \oplus r_{x,d})$ using $K_{x,d-1}$. Then, $S_{x,d}$ uses the hidden KEK to retrieve $K_{x,d}$ from $E_{K_d \oplus r_{x,d}}(K_{x,d})$. $S_{x,d}$ ($S_{y,u}$) decrypts the encrypted secret information discovered in the step ii using $K_{x,d}$ ($S_{y,u}$). These secrets are then used by the given key establishment scheme to establish a pairwise key.

v) Finally, $S_{y,u}$, for instance, only retains $S_{y,u}$, $GenID_u$, $E_{K_{y,u}}(\Phi_{y,u})$, $E_{K_u \oplus r_{y,u}}(K_{y,u})$, $K_{y,u-1}$, $E_{K_u}(r_{y,u} \| f_{y,u})$ and established pairwise keys.

6.3.3 Security Analysis

Similar to the previous approaches, the main concern about this scheme is the security of KEKs. If at least one controlled node obtains a KEK K_c $(1 < c \le t)$, the attacker can distribute K_c to all the other controlled nodes. Using this key, the controlled nodes can help their neighbors to recover plaintext secret information and thus make CLEA possible among nodes from generations from 1 to c. For the attacker's side, it can try to disclose K_c by making use of compromised nodes from either deployed generations or newly added generations. For the former, the attacker compromises a cth generation node $S_{w,c}$ which has been in the network and waits for upcoming node additions. Upon receiving a HELLO message from a newly added node, $S_{w,c}$ sends $E_{K_c}(r_{w,c} \| f_{w,c})$ out and then receives $E_{K_{w,c-1}}(K_c \oplus r_{w,c})$. Using $K_{w,c-1}$ in its memory, $S_{w,c}$ obtains $K_c \oplus r_{w,c}$. Fortunately, $S_{w,c}$ cannot recover K_c since it does not know $r_{w,c}$. Therefore, the first approach fails to subvert the scheme. This also means that HOWHCBS successfully overcomes the security limitation of DOWHCBS.

Concerning the latter, the assumptions about the Class I compromise model in Chapter 2 implies that the attacker cannot gain K_c during I_k. After this interval, the

compromise of node $S_{z,c+1}$ can only help the attacker to obtain $E_{K_{z,c+1}}(\Phi_{z,c+1})$, $E_{K_{c+1}\oplus r_{z,c+1}}(K_{z,c+1})$, $K_{z,c}$, and $E_{K_{c+1}}(r_{z,c+1} \| f_{z,c+1})$, which reveal no clues to retrieve KEKs. Thus, in this case, CLEA is successfully countered. In practice, K_c might be disclosed by the continuous-time attack. The challenging question now is that: *what is the probability that a KEK K_c $(1 < c \le t)$ is compromised over the network lifetime given the loosened assumption*? To answer this question, we use the following notations:

- N_a - The number of the ath generation nodes ($a = \overline{1,t}$): N_a □ N_1 when $1 < a \le t$.
- $N(j) = \sum_{i=1}^{j} N_i$ - The total number of nodes from the first generation to the jth generation.
- L - The network lifetime.
- A_c - The event that K_c is compromised over the network lifetime.
- X_a - The event that K_a is compromised over the interval I_k.
- Y_a - The event that an ath generation node is compromised.
- Z - The event that a node is in the interval of establishing pairwise keys I_k.

Note that due to the system setup phase, K_t cannot be compromised. Therefore, K_c is compromised over the network lifetime when any K_a $(c \le a \le t-1)$ is compromised over the interval I_k. Therefore, we have:

$$P(A_c) = P(X_c \cup X_{c+1} \cup ... \cup X_{t-1}).$$

According to the union bound

$$P(X_c \cup X_{c+1} \cup ... \cup X_{t-1}) \le \sum_{\omega=c}^{t-1} P(X_\omega).$$

X_ω $(c \le \omega \le t-1)$ occurs if only if two events $Y_{\omega+1}$ and Z occur. Hence, we have:

$$P(X_\omega) = P(Y_{\omega+1}/Z)P(Z)$$

$$= \frac{N_{\omega+1}}{N(\omega+1)} \cdot \frac{I_k}{L}.$$

Combining the above equation and inequality, we have the following upper bound

$$P(A_c) \le \sum_{\omega=c}^{t-1} \frac{N_{\omega+1}}{N(\omega+1)} \cdot \frac{I_k}{L}.$$

The values of this upper bound have been plotted for 20 generations ($t = 20$) in Figure 6.14 when N_1 varies from 1000 to 10000; N_as are the same and vary from 20 to 200; I_k and L equal 120s and 9 months respectively. Figure 6.14 shows that the more nodes are added to the network, the more likely K_cs are compromised. Therefore, we can increase the number of nodes in the network deployment to significantly reduce the likelihood of compromising K_cs. Considering the entire network lifetime, the values of the upper bound increase slowly when c decreases from 20 to 2. This means that K_{20} has the lowest probability of being compromised, whereas K_2 has the highest. However, these probabilities, which are bounded approximately by 0 for K_{20} and 6.82×10^{-6} for K_2, are too small to be a cause for security concern. In addition, note that the vulnerability of HOWHCBS can be completely removed if the condition $I_k < I_c$ is guaranteed. This can be achieved if mechanisms increasing the effort to compromise nodes are utilised. These mechanisms can be a hard-to-remove cover for sensor nodes, disablement or removal of on-board programming interfaces.

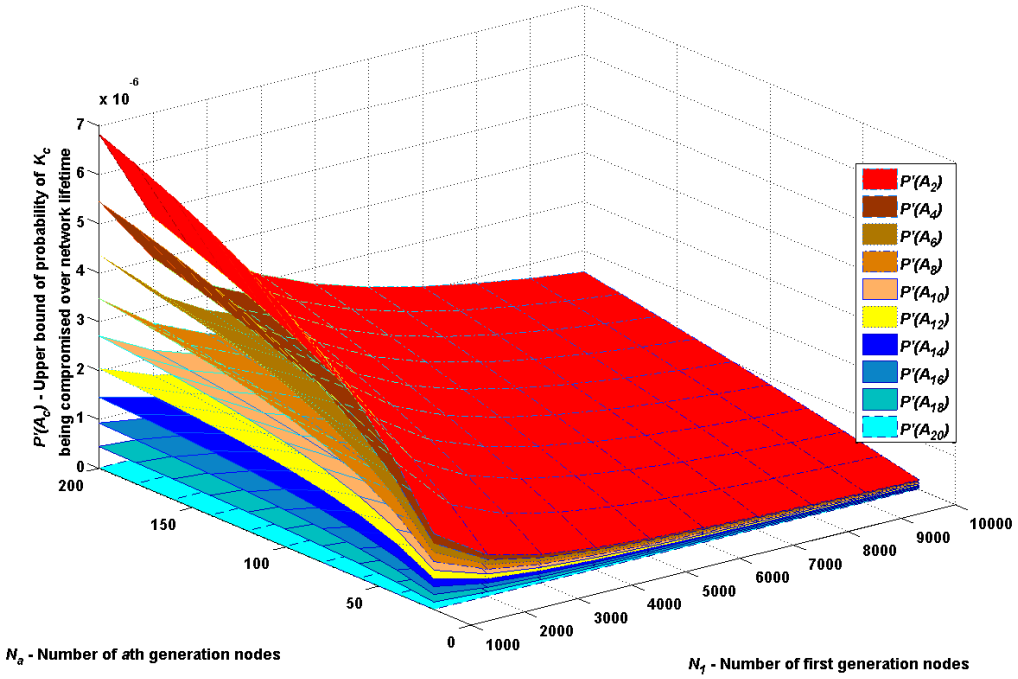


Figure 6.14: The upper bound of the probability of K_c being compromised over the network lifetime

6.3.4 Performance Evaluation

This section provides a detailed evaluation of HOWHCBS in terms of computational, communication, and storage costs.

6.3.4.1 Communication Cost

The communication cost is estimated by the number of messages sent successfully in the entire network. As presented above, a node $S_{z,s}$ must send one HELLO message when added into the network. Furthermore, $S_{z,s}$ sends a message of an encrypted hidden KEK to each kth generation ancestor node in its neighborhood with the probability p. Notice that $n(i) \propto N(i)$, therefore $n(k)$ is estimated at $n(t) \cdot \frac{N(k)}{N(t)}$. Hence, the expected number of messages sent in this case is $p \cdot n(t) \cdot \frac{N(k)}{N(t)} \cdot \frac{N_k}{N(s)}$. Moreover, $S_{z,s}$ locally broadcasts a HELLO message once after a new uth successor generation addition and an encrypted message of the random numbers once with the probability p if there is at least one uth generation node in its neighborhood. The probability that at least one uth generation node in the neighborhood can be computed as $1-\left(1-\frac{N_u}{N(t)}\right)^{\frac{n(t)\cdot N(u)}{N(t)}}$. Combining the above arguments, we have the estimated communication cost of $S_{z,s}$ over the network lifetime is

$$1+p\cdot\frac{n(t)}{N(t)}\cdot\sum_{k=1}^{s-1}\left(N_k\cdot\frac{N(k)}{N(s)}\right)+(1+p)\sum_{u=s+1}^{t}\left(1-\left(1-\frac{N_u}{N(u)}\right)^{n(t)\cdot N(u)/N(t)}\right).$$

Figure 6.15 shows the communication cost of each node from 20 generations over the network lifetime where N_1 equals 10000; N_as are the same and vary from 20 to 200. As depicted in the figure, nodes of the first generation, which has the overwhelming population, incur the lowest cost of about 2 messages each. Meanwhile, nodes of the 2nd generation, which has very small population, incur the highest cost of about 30 messages each. Therefore, the average cost of a node in the network is only a average of

7.5728 messages. This small number shows the high communication efficiency of HOWHCBS.

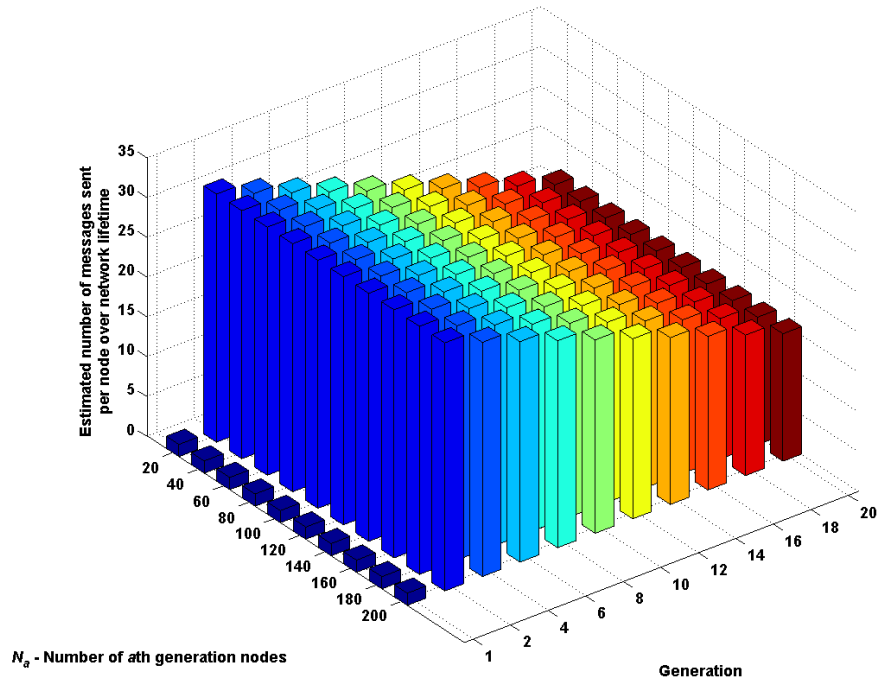


Figure 6.15: The estimated number of messages sent per node of each generation over the network lifetime

6.3.4.2 Computational Cost

After being injected into the network, a sth generation node $S_{z,s}$ incurs the costs of the following computational operations with the probability p :

- One decryption for each sth generation node in its neighborhood.
- One KEK computation for each kth ancestor generation.
- One decryption of the encrypted random numbers, one KRK computation, one hidden KEK encryption, and one decryption of the encrypted common secrets for each kth ancestor generation node.
- One decryption to get the hidden KEK, one decryption to get the SEK for each uth successor generation, and one decryption of the encrypted common secrets for each uth successor generation node.

The first item cost can be estimated as follows. The probability that a *s*th generation node is present somewhere in the network is $\frac{N_s}{N(s)}$. Then the number of *s*th generation nodes in the neighborhood is $\frac{N_s}{N(s)} \cdot n(s)$. Therefore, the first estimated cost is $\frac{N_s}{N(t)} \cdot n(t) \cdot p \cdot D$. Note that the probability that at least one *k*th ancestor generation node exists in the neighborhood is $1-\left(1-\frac{N_k}{N(s)}\right)^{\frac{n(t)\cdot N(s)}{N(t)}}$. Therefore, the cost of the second item is estimated at $\sum_{k=1}^{s-1}\left(1-\left(1-\frac{N_k}{N(s)}\right)^{\frac{n(t)\cdot N(s)}{N(t)}}\right) \cdot p \cdot (s-1-k) \cdot H$. Likewise, it is straightforward to derive the estimated cost formulas for the last two items, which are $\sum_{k=1}^{s-1} \frac{N_k}{N(t)} \cdot n(t) \cdot p \cdot \left(2 \cdot D + H + E\right)$ and $\sum_{u=s+1}^{t}\left(\left(1-\left(1-\frac{N_u}{N(u)}\right)^{\frac{n(t)\cdot N(u)}{N(t)}}\right) \cdot p \cdot 2 \cdot D + \frac{N_u}{N(t)} \cdot n(t) \cdot p \cdot D\right)$.

As a result, the total computational cost of $S_{z,s}$ over the network lifetime is estimated at

$$p \cdot \frac{n(t)}{N(t)} \cdot N(s-1) \cdot E$$

$$+ p \cdot D \cdot \left(n(t) + 2 \cdot \frac{n(t) \cdot N(s-1)}{N(t)} + 2 \cdot \sum_{u=s+1}^{t}\left(1-\left(1-\frac{N_u}{N(u)}\right)^{\frac{n(t)\cdot N(u)}{N(t)}}\right)\right)$$

$$+ p \cdot \left(\sum_{k=1}^{s-1}\left(1-\left(1-\frac{N_k}{N(s)}\right)^{\frac{n(t)\cdot N(s)}{N(t)}}\right) \cdot (s-1-k) \cdot H \right).$$

To make the above formula more interpretative, we plot Figure 6.16 to show the cost with the following setting: $N_1 = 10000$, $N_i = 100\ (i = \overline{2,t})$, $t = 20$, $n(t) = 50$, $p = 0.4$. As shown in the figure, the vast majority of nodes coming from the 1st generation incur the lowest costs of an average of 0 hashing, 0 encryption, and 25.2338 decryption operations for each node. Moreover, the greater the number of generations is, the higher

computational cost the last generation nodes have to be incurred due to the increasing number of hashing operations. The twentieth generation node incurs the highest cost of an average of 28.2668 hashing, 19.8319 encryption, and 59.6639 decryption operations. The average cost incurred by a node is of 1.6961 hashing, 2.9249 encryption, and 30.6438 decryption operations.

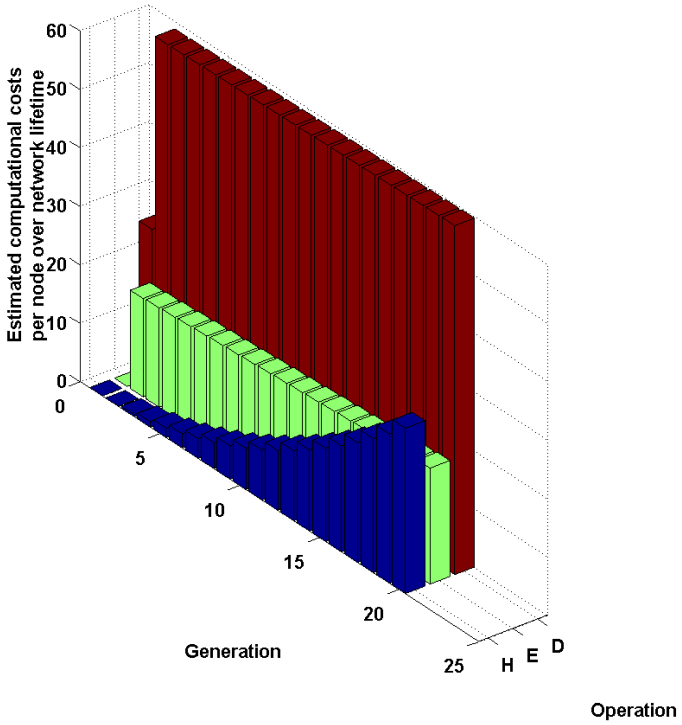


Figure 6.16: The estimated computational costs per node of each generation over the network lifetime

6.3.4.3 Storage Cost

We have simulated HOWHCBS using TOSSIM [170] together with TinyOS 1.1 [21] to evaluate the storage requirement. In this simulation, we assume that the random-pairwise keys scheme [28] is used for the key establishment in a network setting of $N = 1000$ nodes, key ring size: $m = 300$, and neighbourhood size: $n = 50$. The measurement of the ROM and RAM consumption of HOWHCBS is conducted based on MICAz platform, and the result is reported in Table 6.2. It can be seen that, given that the sizes of ROM and RAM in MICAz are 128KB and 4KB respectively, the storage requirements of about 17.1KB ROM and about 0.77KB RAM are reasonable.

Table 6.2: Storage cost of HOWHCBS (in byte)

	Full program	**AntiKSCA**
MICAz		
ROM	29950	17550
RAM	1221	792

6.4 System Implementation

In order to demonstrate the feasibility and practicality of the aforementioned schemes, a hardware system model of HOWHCBS is implemented. The implementation uses the HashFunction interface to construct the one-way hash chain and compute KRKs. This interface is provided in the Hmac-MD5 library developed by UbiSec&Sens project [171]. The implementation re-uses the SkipJack code in TinyOS 1.1 to provide encryption and decryption and the RandomMLCG library to generate random numbers. The detailed depiction of the component relationship in the implementation is given in Figure 6.17. As shown in the figure, the implementation has the following main components.

- Main: As described in Chapter 5, this component initialises the other components such as HelloCounter, Storage, PredGenPhase, SameGenPhase, SuccGenPhase, SJCipherC via StdControl interface.
- Storage: This component stores pre-loaded information such as node ID, generation ID, and encrypted key ring. It also stores information received from neighbours such as neighbour IDs, neighbour generation IDs, random numbers, and established pairwise keys. Furthermore, it supports read and write operations on sensor memory.
- HelloCounter: This component decides a node's starting and ending time of broadcasting hello messages after node deployment. It uses StorageOps interface to get the node and generation IDs from Storage and then sends it to HelloToRfm via HelloOutput interface for the hello message transmission.
- HelloToRfm: This component broadcasts the hello message which contains node ID and generation ID to neighbourhood.

- RfmToHello: This component receives hello messages from neighbourhood and save node IDs and generation IDs into Storage via StorageOps interface. It determines if common secrets exist between the receiver and senders. Based on the generation ID, it determines if the receiver is from the same or different generation to the senders and invokes the receiver's corresponding component: SameGenPhase, PredGenPhase, or SuccGenPhase as shown in Figure 6.18.

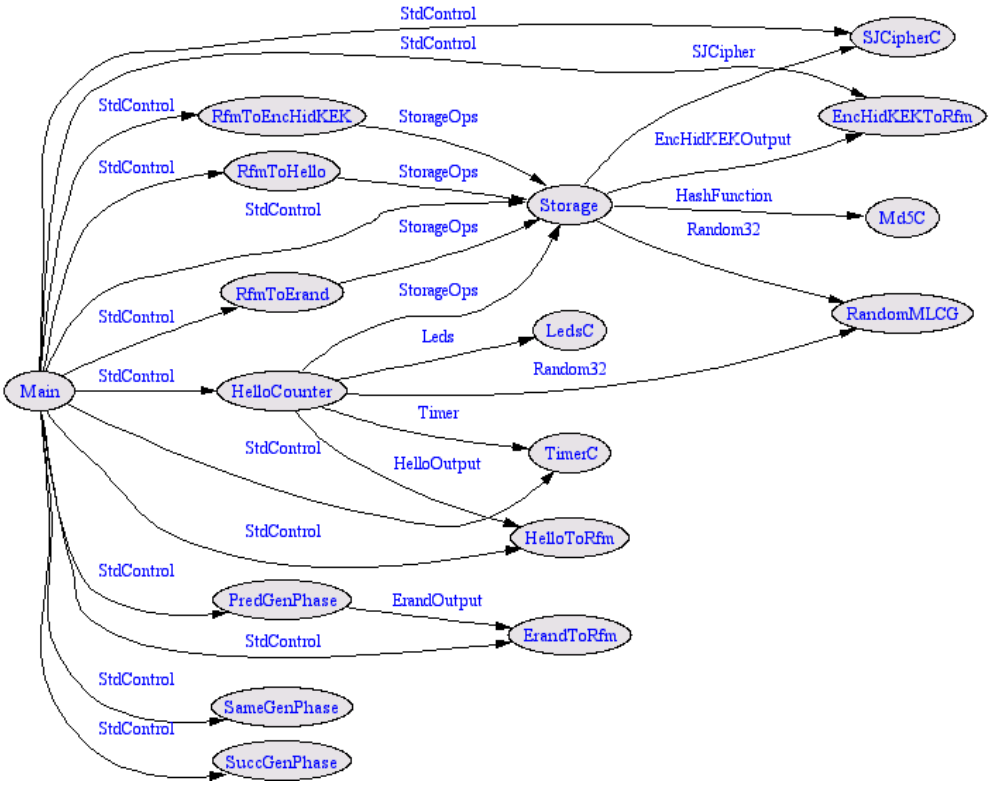


Figure 6.17: The component relationship in the implementation of HOWHCBS

- ErandToRfm: This component sends out encrypted random numbers via ErandOutput interface.
- RfmToErand: This component receives encrypted random numbers and stores them in Storage via StorageOps.
- EncHidKEKToRfm: This component sends out encrypted hidden KEKs via EncHidKEKOutput interface.
- RfmToEncHidKEK: This component receives encrypted hidden KEKs and stores them in Storage via StorageOps.

- PredGenPhase: This component is activated in a successor node. It sends out encrypted random numbers, receives the encrypted hidden KEK, obtains the SEK, retrieves the plaintext secret information, and finally performs the key establishment activity. Its final step is deleting all unnecessary keys and information.
- SameGenPhase: This component is activated when the same generation key establishment occurs. It retrieves the plaintext secret information, performs the key establishment activity, and finally deletes all unnecessary keys and information.
- SuccGenPhase: This component is activated in an ancestor node. It computes the KEK and KRK of the successor node, sends out encrypted hidden KEKs, performs the PKE activity, and finally deletes all unnecessary keys and information.
- Md5C: This component provides a hash function to compute KEKs and KRKs.
- SJCipherC: As shown in Figure 6.19, this provided component implements the Skipjack algorithm. It provides the encryption/decryption service for data before being sent out.

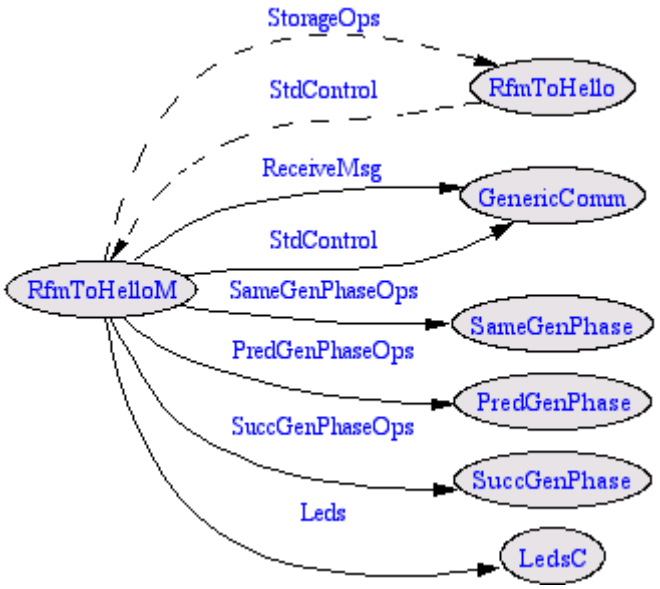


Figure 6.18: The component relationship in RfmToHello

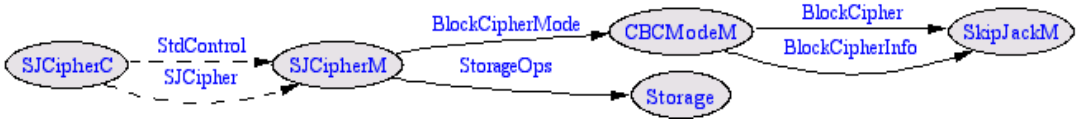


Figure 6.19: The component relationship in SJCipherC

6.5 Chapter Remarks

In this chapter, a series of anti-CLEA schemes, namely OWHCBS, DOWHCBS and HOWHCBS have been presented. These schemes exhibit a number of appealing properties. First, they are applicable to any key establishment scheme for DSNs. Second, they robustly defend key establishment schemes against CLEA at negligible extra costs. Third, they do not resort to any requirement of additional and special functionalities and resources to network entities such as aggregators and base stations, topological knowledge in advance and/or costly location-based detection algorithms. Fourth, they provide network scalability via incremental generation deployments. Finally, their efficiency and plausibility are well justified through extensive analyses, simulations, and implementation.

Chapter 7

Per Node Deployment Based Detection Countermeasures

In this chapter, a family of countermeasures is developed in an evolutionary manner to detect and defeat CLEA. As opposed to the passive approach presented in Chapter 6, the approach presented in this chapter is quite active towards CLEA and completely independent from key establishment schemes. The first two countermeasures equip sensor nodes with the ability to pinpoint controlled nodes, the source of CLEA. The final countermeasure is even more powerful than the previous ones with the capability to revoke controlled nodes.

The chapter starts with the general design goal and overview of the countermeasures followed by the naïve detection scheme. The adaptive detection scheme is then introduced to overcome the limitations of the naïve scheme. Finally, the extended adaptive detection scheme is developed by extending the adaptive scheme in order to provide sensor nodes with the revocation capability.

7.1 Design Goal and Overview

7.1.1 Design Goal

This chapter focuses on designing a family of schemes to defend against CLEA by detecting and stopping the controlled nodes' attempt to establish controlled links with uncompromised nodes in a DSN. They aim to detect this attack in a distributed fashion in which any node in the network can be a detector. Since DSNs are usually left to

operate in an unattended manner for long periods of time, the attack needs to be stamped out at the very first attempt; otherwise the attacker has time to launch other malicious attacks using the controlled yet legitimate links. Moreover, the designed schemes should be independent of specific assumptions such as priori topological and deployment knowledge, undesirable requirements of additional functionalities and resources, and key establishment schemes in order to be applicable to most, if not all DSNs. Finally, the designed schemes should be both efficient in order to save network resources and thus prolong the network lifetime and scalable to accommodate additional nodes.

7.1.2 Overview

The basic idea of these schemes is as follows. Each node maintains a counter to keep track of the node deployment/addition activity in its neighbourhood. This counter increases by 1 after a constant time interval or on the event of node deployment/addition. Each incoming node attempts to join the network by broadcasting a HELLO message which contains a newest constant counter value. The neighbouring nodes admit the incoming node only if their maintained counter values are equal to the one in the received HELLO message. Otherwise, the neighbours know that the received HELLO message is replayed by controlled nodes. Therefore, CLEA is detected and the controlled nodes are black-listed by the receivers.

7.2 Naïve Detection Scheme

This scheme is the simplest one of the proposed schemes and is discussed for the sake of completeness. The presentation of this scheme provides a conduit for understanding the later proposed schemes.

7.2.1 Assumptions

In addition to the assumptions given in Chapter 2, the following assumptions are made when the naïve scheme is developed:

i) The network is protected by broadcast authentication schemes which have two basic properties: data authentication and source authentication. More specifically, data authentication ensures that the messages from senders (security server or sensor nodes) are unchanged en route to receivers (sensor nodes) and thus provides message integrity as well. Source authentication assures the receivers that messages originate from the claimed senders. These broadcast authentication schemes can be achieved using either symmetric-key approaches or public-key approaches as discussed in chapter 4. For the rest of the thesis, without loss of generality, let us denote $\langle Msg, AUTH_i(Msg)\rangle$ as the authenticated version of a broadcast message Msg from node S_i.

ii) Each of all network entities is equipped with a clock. These clocks are precisely synchronised with each other. Furthermore, each entity also maintains a counter associated with the clock whose value increases after each constant time interval ΔT as demonstrated in Figure 7.1.

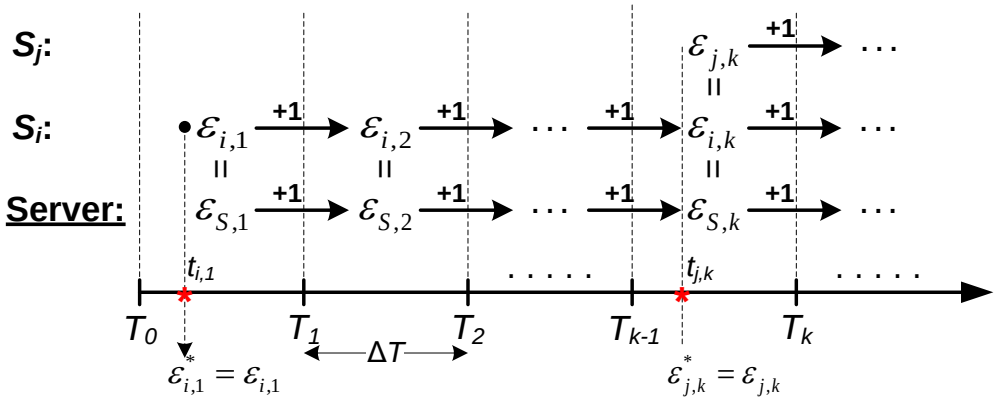


Figure 7.1: Counter value increment in sensor nodes following the naïve scheme

iii) Each time interval ΔT_k can accommodate only one deployment which can only occur at any arbitrary point of time over the interval except T_{k-1} and T_k. This assumption is justifiable because given a specific application of the DSN, the value of ΔT can be fine-tuned based on the following inputs: the deployment lifetime and the frequency of the deployment. For applications of a high deployment rate, the deployment lifetime is divided into small intervals with the small value of ΔT, e.g. several hours or days, in such a manner that only one

deployment point falls into one interval. Meanwhile, for ordinary applications in which the frequency is low, the value of ΔT can be increased, e.g. several weeks or months.

iv) Each newly deployed and legitimate node only makes requests for secure link establishment with its neighbors during a small window of time δT which is long enough for the node to finish all its secure link establishment activities. The proper value of δT can be determined via the extrapolation from the result presented in [53]. Thereafter, it does not initiate the secure link establishment requests. Instead, it simply waits for the requests from later deployed nodes within its neighborhood.

7.2.2 Initialisation of Security Server and Sensor Nodes

The security server is initialised with a counter C_s whose value is initialised to equal 1 at T_0^S and incremented by 1 after each interval ΔT. That is, the value of C_s at the τ-th time interval is $\varepsilon_{S,\tau} = \varepsilon_{S,\tau-1} + 1$. The maximum value of C_s is $t+1$ where t is the number of deployments over the network lifetime. The timer of the security server also stops when C_s reaches $\varepsilon_{S,t+1}$.

Each sensor node S_i which is deployed over the τ-th time interval ($\tau >= 1$) is initialised by the security server with the following information:

- Sensor node ID S_i.
- A counter C_i, whose value $\varepsilon_{i,\tau}$ is initialised to equal $\varepsilon_{S,\tau}$ ($\varepsilon_{i,\tau} = \varepsilon_{S,\tau}$). This value increases by 1 after each interval ΔT starting from $T_{\tau-1}^n = T_{\tau-1}^S$: $\varepsilon_{i,\tau+1} = \varepsilon_{i,\tau} + 1$. The maximum value of C_i is $t+1$ as well and the timer of S_i also stops when C_i reaches $\varepsilon_{S,t+1}$.
- $HelloMsg_{i,\tau} = \langle S_i \,||\, \varepsilon_{i,\tau}^*, AUTH_i(S_i \,||\, \varepsilon_{i,\tau}^*) \rangle$: The HELLO message consists of the node ID, the counter value of the security server over the τ-th time interval, that is $\varepsilon_{i,\tau}^* = \varepsilon_{S,\tau}$, and the authentication code/tag for $S_i \,||\, \varepsilon_{i,\tau}^*$.

7.2.3 Pre-programming of Sensor Nodes

One might argue that the above initialisation approach does not allow for node pre-programming. This is because the fact that the exact deployment time interval of a sensor node has to be known in advance in order to obtain the right HELLO message with a valid counter value from security server. However, this argument does not make sense for two reasons. On the one hand, an overlapped pre-programming strategy does allow sensor nodes to be pre-programmed to a certain extent. This strategy can be developed as follows. Since the deployment time of the first deployed nodes in the majority is known in advance, they can be pre-programmed before deployment. The rest of the network (computed by subtracting the number of the first deployed nodes from the estimated overall number of nodes) is divided into a small number of deployment groups, e.g. 3 groups indexed from 2 to 4. The groups 2, 3, 4 are then pre-programmed with HELLO messages to be deployed in the intervals ΔT_2, ΔT_3, ΔT_4 respectively. If in the real deployment situation, there is no deployment during, ΔT_3, for example, the nodes supposed to be deployed over ΔT_3 and the leftover nodes from group 2 are re-pre-programmed for future intervals such as ΔT_5, ΔT_6, etc. In such situation, the security server is used to decide which time interval we are in so that we can deploy the right pre-programmed groups. In this way, we can enable the flexibility in pre-programming of sensor nodes as well as the deployment time.

On the other hand, in many cases, the pre-programming of sensor nodes is not desirable due to two reasons. First, we do not know the number of nodes of each deployment beforehand. Therefore, if we want to pre-program nodes, we have to estimate the number of nodes for each deployment. However, the estimation is often not accurate. Hence, the re-pre-programming has to be repeated all the time. Second, for some applications in which the deployment occurs over a long period of time such as several months or years, it is not efficient to reserve (via pre-programming) a large number of sensor nodes for one application using the proposed schemes. Real-time programming enables these nodes to be used for other applications while they are not being used by the current application. In this way, the efficiency of resource use is improved.

7.2.4 Detection of CLEA at Sensor Nodes

When a legitimate sensor node S_i receives a request of secure link establishment $HelloMsg_{j,\varphi}$ from S_j during the ϕ-th time interval, it executes the following steps to verify the authenticity, integrity, and legitimacy of the request:

- Examines the authenticity and integrity of the request: The request is rejected if it is not authentic and intact, otherwise S_i believes that $HelloMsg_{j,\varphi}$ is generated by the security server.
- Compares $\varepsilon_{i,\phi}$ with $\varepsilon^*_{j,\varphi}$: Owing to the synchronisation between the server counter and sensor node counters, there can be only two possibilities. First, S_j has just been deployed in the same time slot ϕ. This means $\phi = \varphi$ and $\varepsilon_{i,\phi} = \varepsilon^*_{j,\varphi}$. In this case, S_i believes that S_j is a newly deployed legitimate node and permits S_j to establish a secure link with itself. Second, S_i and S_j are in the different time slots. This means $\phi > \varphi$ and $\varepsilon_{i,\phi} > \varepsilon^*_{j,\varphi}$. In this case, S_i believes that the $HelloMsg_{j,\varphi}$ is replayed by the controlled node S_j. S_i simply rejects the request and puts S_j on its blacklist.

7.2.5 Discussion

At first glance, the naïve scheme seems to provide an efficient mechanism for detecting and thwarting CLEA. However, the following problems are exposed upon further examination.

- *Small* ΔT: In practice, if ΔT is set to be short enough, it is very likely that $\varepsilon_{i,\phi}$ $(= \varepsilon_{i,\varphi})$ is greater than $\varepsilon^*_{j,\varphi}$ even if S_i and S_j are newly deployed in the same time slot. The reason is that there may be a delay Δ_d in receiving the $HelloMsg_{j,\varphi}$ induced by unreliable transmission of $HelloMsg_{j,\varphi}$. During Δ_d, $\varepsilon^*_{j,\varphi}$ remains unchanged while $\varepsilon_{i,\phi}$ ($\varepsilon_{i,\varphi}$) keeps increasing as long as $\Delta_d \geq T_\varphi - t_{j,\varphi}$. This results in this inequality: $\varepsilon_{i,\phi} > \varepsilon^*_{j,\varphi}$. In consequence, newly deployed legitimate nodes might fail to join the network.

- *Large* ΔT : At the opposite extreme, we can increase ΔT to be large enough such that $\varepsilon_{i,\phi}$ equals $\varepsilon^*_{j,\varphi}$ regardless of the delay Δ_{d} . This means the small ΔT problem can be overcome and newly deployed legitimate nodes can join the network. Unfortunately, the large ΔT again introduces another problem. Due to the large ΔT , the gap between $t_{j,\varphi}$ and T_{φ} might be large. Therefore, $\varepsilon_{i,\phi}$ might remain unchanged and equal $\varepsilon^*_{j,\varphi}$ over this long period. Consequently, it is very likely that the attacker has enough time to compromise some nodes and then launch CLEA by replaying HELLO messages with pre-deployed nodes during this period. Furthermore, controlled nodes can also establish secure links with later deployed nodes without restraint.
- *Counter de-synchronisation*: Due to the drift and error of timers at server and node sides, $\varepsilon_{S,*}$ might increase faster (more slowly) than $\varepsilon_{i,*}$ such that $\varepsilon^*_{j,\varphi}$ is greater (smaller) than $\varepsilon_{i,\phi}$ regardless of the delay Δ_{d} . This results in network failure to accommodate newly deployed nodes.

An enhanced scheme to address the above issues by introducing two modes of timing for the counters is presented as below.

7.3 Adaptive Detection Scheme

7.3.1 Assumptions

In essence, the assumptions of the adaptive detection scheme are similar to those of the naïve scheme. However, there are some minor modifications made to the previous assumptions as follows:

i) The clocks of all sensor nodes and the security server are not precisely but loosely synchronised with each other using secure time synchronisation protocols [168, 169]. Specifically, the clock difference between the security server and any node is bounded by δ_{diff} .

ii) Only one deployment can occur at any arbitrary point of time within this interval $\left[\max\left\{T_i^S + \delta_{diff} - \Delta_d, T_i^S\right\}, T_{i+1}^S - \delta_{diff} - \Delta_d\right)$, where $i \geq 1$. For $i = 0$, the interval for the deployment is within $\left[T_0^S, T_1^S - \delta_{diff} - \Delta_d\right)$, where $T_0^S = T_0^n$.

When these assumptions are incorporated into the adaptive scheme, it is easy to see that the small ΔT and counter de-synchronisation problems in the naïve scheme are eliminated.

7.3.2 Initialisation of Security Server and Nodes

The procedure of initialising the security server and sensor nodes is identical to that of the naïve scheme. The security sever is initialised with a counter whose value increases by 1 after each constant interval ΔT. A node deployed over ΔT_k is pre-loaded with a node ID, a counter whose value is initialised to equal the security server's counter value over ΔT_k and increased by 1 after each ΔT, and a HELLO message whose sampled counter value is also equal to the security server's counter value over ΔT_k and remains unchanged. The HELLO message is deleted after the key establishment process in the node finishes.

7.3.3 Deployment of Sensor Nodes and Detection of CLEA

Sensor nodes to be deployed over ΔT_i have been initialised at some time before the beginning of ΔT_i as described in Section 7.2.3. For example, sensor nodes to be deployed over the first interval ΔT_1 (group 1) have been initialised before T_0^S. As shown in Figure 7.2, at $t_{S,1}$, the initialized sensor nodes of group 1 are powered on and deployed over the sensor field. After that they start broadcasting HELLO messages to their neighborhoods. Now let us call $t_{i,1}$ the time point where node S_i of group 1 receives the first legitimate HELLO message, then $\Delta_d = t_{i,1} - t_{S,1}$ is the delay of the S_i's receipt of the HELLO message. This delay is induced mainly by the unreliable transmission of the message. Because the sensor nodes are deployed and powered up at

the same time, Δ_ds observed by different sensor nodes are approximately the same. To guarantee that almost if not all HELLO messages are received within δT, a small time window for secure link establishment, we can make use of the work presented in [53] to determine the suitable value for δT. For example, if the number of node in a neighborhood is 20, the practical value for δT should be around 10 seconds.

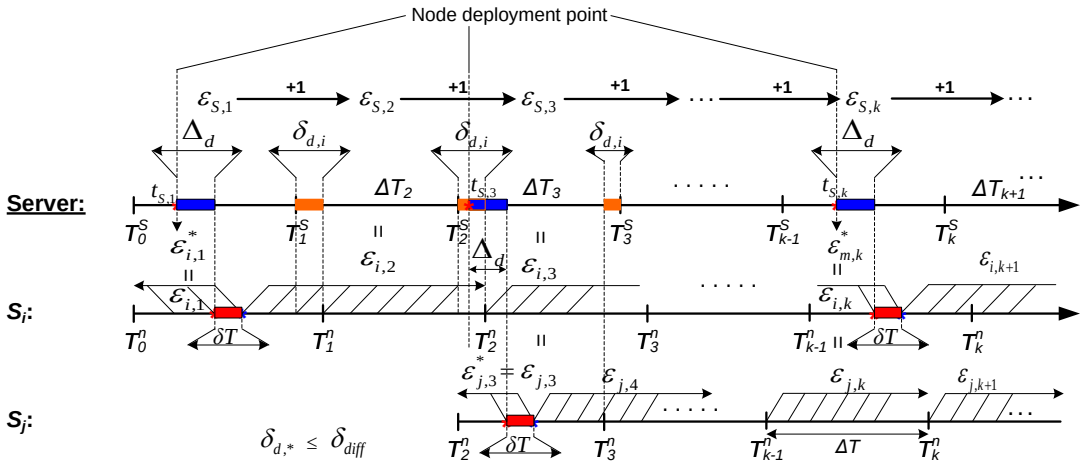


Figure 7.2: The counter value increment in sensor nodes following the adaptive detection scheme

Let us consider a sensor node S_n under two scenarios during ΔT_ϕ: newly deployed and pre-deployed. For the first scenario exemplified by node S_i over ΔT_1 and node S_j over ΔT_3, to join the network, S_n broadcasts its $HelloMsg_{n,\phi}$ to its neighbors. Right after δT, S_n deletes its HELLO message and increases its counter value by 1 regardless of whether ΔT_ϕ is over or not. Hence the counter value of S_n now is $\varepsilon_{n,\phi+1}$. From this point to the end of ΔT_ϕ (T_ϕ), if S_n receives any $HelloMsg_{r,\mu}$ ($\mu \le \phi$) from S_r, it believes that the $HelloMsg_{r,\mu}$ is replayed by CLEA due to this inequality $\varepsilon_{n,\phi+1} > \varepsilon^*_{r,\mu}$. Thus, it rejects the request and puts S_r on its blacklist.

The second scenario exemplified by S_i over ΔT_k in Figure 7.2 implies the situation that S_n has already joined the network. During ΔT_ϕ, S_n might or might not receive a request for establishing a secure link. It will not open the δT window of ΔT_ϕ until it receives

the first authentic and CLEA free request from S_m, for example. If it is the case, S_n believes that a number of sensor nodes are newly deployed and some of these are its new neighbours which want to establish secure links with itself. It then opens the δT window. Assume that the $HelloMsg_{r,\varphi}$ from S_r is one of the requests S_n receives over δT. S_n executes the same process as detailed in the naïve scheme including examining the authenticity and integrity of the request and comparing $\varepsilon_{n,\phi}$ with $\varepsilon^*_{r,\varphi}$ to detect CLEA. After the δT window, S_n increase its counter value to $\varepsilon_{n,\phi+1}$ in order not to accept any link establishment requests during the rest of ΔT_ϕ. This counter value remains unchanged during $\Delta T_{\phi+1}$ if there is not another deployment over this interval as illustrated in Figure 7.2. Thereafter it keeps increasing after each ΔT until the next deployment.

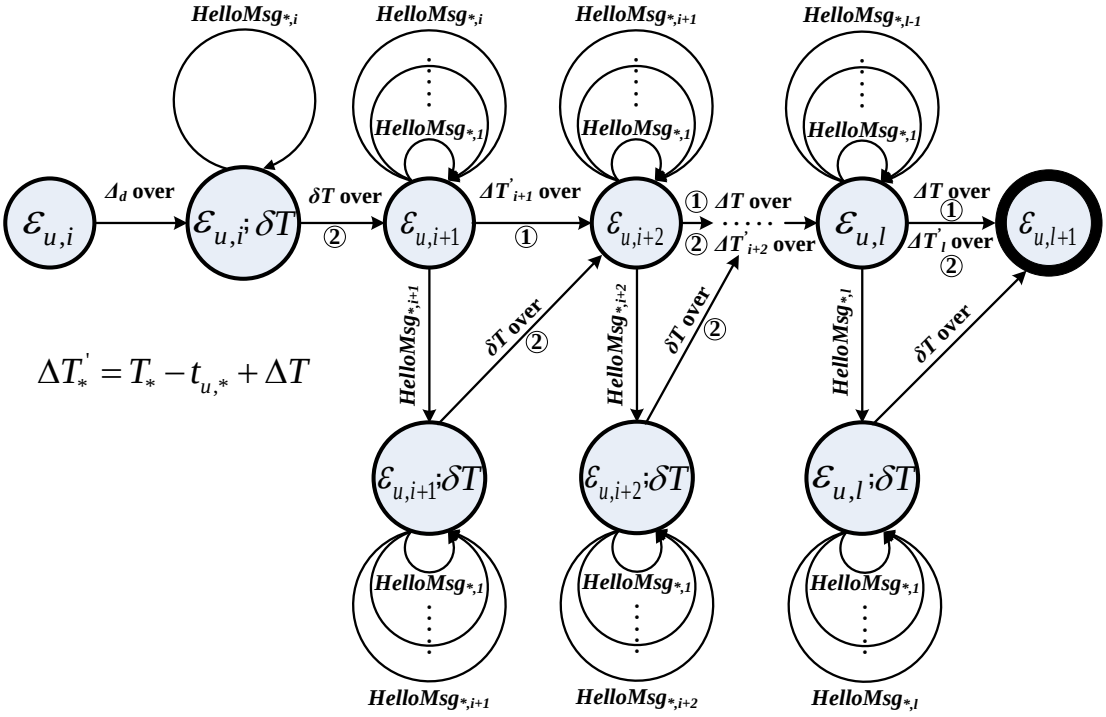


Figure 7.3: Counter value transition in S_u deployed over ΔT_i

Figure 7.3 illustrates the state transition of the counter of a sensor node S_u deployed over the ith time interval. After Δ_d, S_u is in the δT time window to process incoming HELLO messages from sensor nodes deployed over ΔT_i while its counter value is still

$\varepsilon_{u,i}$. When δT is over, the counter value increases by 1 to $\varepsilon_{u,i+1}$ and S_u is put into the interval ΔT_{i+1}. In this state, S_u will consider any out-of-date HELLO messages, which come from the intervals between ΔT_1 to ΔT_i, to be the indication of CLEA. If it will not receive any legitimate HELLO messages from upcoming sensor nodes over ΔT_{i+1}, it will wait for the period of $\Delta T'_{i+1} = T_i - t_{u,i} + \Delta T$ before increasing its counter value to $\varepsilon_{u,i+2}$ and then continue with the counter value increment after each ΔT as usual. If S_u receives any legitimate $HelloMsg_{*,i+1}$ from a node deployed in ΔT_{i+1}, it switches to the δT time window mode and repeats the same process as described above.

7.3.4 Security Analysis and Discussion

Effectively, the adaptive scheme introduces two modes for increasing counter values. A node increases its counter values after either the small window δT (*short-time mode*) or the large time interval ΔT (*long-time mode*) based on whether deployments occur or not. By taking advantage of the flexibility of these two modes, the scheme can remove the large ΔT problem exposed in the naïve scheme. Therefore, combining with the assumptions in section 7.3.1, it is straightforward to see that the adaptive scheme successfully overcomes the problems of the naïve scheme.

The scheme works well in the stabilised network in which all nodes have been deployed and no nodes are coming into the network. When there are nodes coming into the network, two sophisticated attacks might happen. The following are to discuss the possibility of these attacks on the adaptive scheme.

- Forward CLEA: This attack can be launched based on the assumption that the attacker already compromised some pre-deployed nodes from which controlled nodes were generated. Although these controlled nodes cannot be used to establish controlled links with pre-deployed legitimate nodes, they might be used to establish controlled links with nodes deployed later. Nevertheless, the following reasons render the attack impractical. First, the attacker cannot predict the locations of newly deployed nodes. Therefore, it cannot deploy the controlled nodes exactly at these locations in advance to wait for upcoming nodes. Second, the scope of this attack is limited because a large number of

sensor nodes are already deployed in the first deployment. The attacker cannot mount the forward CLEA on these nodes. Last but not least, note that when launching the attack, controlled nodes have to introduce their presence, which has not been known by their neighbours before, to their neighbourhoods. Therefore, if each sensor node uses some watchdog mechanism (e.g. [172]) to keep a frequently updated neighbour list, they can detect the presence of the controlled nodes. Using some distributed voting scheme (e.g. [28]), pre-deployed nodes can alert incoming nodes of the controlled nodes' presence. Based on the alert, the incoming nodes stop keyed link establishment with the controlled nodes and put them into their blacklists.

- Backward CLEA: The backward CLEA allows controlled nodes to establish controlled links with pre-deployed nodes. It can happen with the aid of the continuous-time attack. In other words, the attacker can compromise nodes just after their deployment and then generate controlled nodes over some interval ΔT_c. Thereafter, the attacker can make use of the controlled nodes to establish controlled links with legitimate nodes throughout the network until the end of ΔT_c provided the window δT of these legitimate nodes has not expired. Fortunately, the likelihood of this attack is mitigated since the number of controlled links established over ΔT_c is limited owing to the fact that the controlled nodes can not establish controlled links after δT and the number of nodes deployed in each later deployment is small. Furthermore, this attack can be virtually eliminated if some mechanisms which increase the effort to compromise nodes are applied. As discussed in [47], the time to compromise sensor nodes can be significantly increased by some basic mechanisms such as using hard-to-remove covers, removing or disabling nodes' programming interface, protecting the programming interface with passwords. A more complicated method of launching the backward CLEA could be employed as follows. The attacker first deploys many controlled sensors in the network. After a new sensor node S_i is deployed, the attacker records the HELLO message sent from S_i and replays the message at all the controlled nodes within δT time window. Because the HELLO message has the valid counter value, it tricks the neighbours of the controlled nodes to believe that a legitimate new sensor S_i is

deployed in their neighbourhoods. Then the attacker with unlimited time to compromise S_i can obtain the secret information and then enable the controlled nodes to impersonate S_i through the established controlled links. Nevertheless, this attack is not possible due to two reasons. First, as discussed in Chapter 2, in key establishment schemes for sensor networks, a node ID is usually tied to set of unique secret keys used to establish pairwise/group keys. In this attack, the controlled nodes try to use secret keys which are not originally associated with S_i to establish pairwise/group keys with their neighbours within the δT window. As a consequence, the controlled nodes and their neighbours cannot agree upon common keys. In other words, the controlled nodes fail to establish controlled links with their neighbours over the δT window. Second, the δT window is very short. The attacker can compromise S_i to obtain the S_i's secret keys and let the controlled nodes know the keys so that they can fully impersonate S_i. But by the time the controlled nodes get to know the S_i's secret keys, the δT window has already been closed.

7.4 Extended Adaptive Detection Scheme

As analysed before, the adaptive scheme might not be totally immune from CLEA in the worst-case scenarios. Obviously, in an application scenario which demands rigorous security, this susceptibility makes the scheme unappealing. Fortunately, we can extend this scheme to the extended adaptive detection scheme to address this issue. The latter differs from the former in its assumptions and detection of CLEA at sensor nodes. The revocation of controlled nodes in the extended scheme is achieved by adapting the idea of LSM in [9] or its variants in [77, 102].

7.4.1 Assumptions

To develop the extended scheme, the assumptions made in the adaptive scheme are extended by adding two more assumptions as follows.

i) Sensor nodes are equipped with the knowledge of geographic positions as assumed in [9, 77, 101, 102, 173] using localisation schemes [160, 161, 174-177].

ii) In contrast to the naïve and adaptive schemes, sensor nodes in the extended scheme are capable of authenticating any outgoing messages and verifying any pass-by/incoming message using either symmetric-key or asymmetric-key authentication techniques [138, 146, 178-181].

7.4.2 Detection of CLEA at Sensor Nodes

The detection process is executed after each deployment of sensor nodes. All pre-deployed nodes in the neighbourhood of a newly deployed node as well as the newly deployed node itself are under suspicion of being controlled nodes. This must be resolved. After having been deployed, a node makes a request for the link establishment by broadcasting its HELLO message locally. Upon receiving the request, each neighbour in the newly deployed node's neighbourhood first verifies the authenticity of the request and then checks the request freshness by comparing whether its counter value equals the one in the request. If the authenticity or freshness is unconfirmed the neighbour will then believe that the requester is a controlled node otherwise it accepts the request by sending out an authenticated message (location claim) containing its ID and location (e.g., geographic coordinates) in response. Upon hearing this message, each neighbouring node of the neighbour follows the approach in LSM, for example, to detect the controlled nodes. Specifically, each location claim receiver probabilistically forwards the claim to a randomly chosen set of witness nodes. By requiring each node en route to store a copy of this claim, the scheme effectively draws r line segments per location claim across the network. If a conflicting location claim ever traverses the segments, the node at the intersection will detect the conflict and initiate an authenticated revocation broadcast.

In addition, as mentioned above, it might be the case that the newly deployed node is also a controlled node. To deal with this situation, the scheme requires that the newly deployed node also has to follow the detection process by broadcasting its location claim locally after receiving the location claims from its neighbours. If the node fails to perform this task, it is alleged to be a controlled node by its neighbours even though it has passed the counter value check previously.

Figure 7.4 illustrates how controlled nodes are detected by the extended scheme. It is assumed in this figure that S_x and S_k belonging to the same deployment are newly scattered to different geographic locations of the sensor field. From the security viewpoint, any neighbour of S_x and S_k including themselves can be a controlled node.

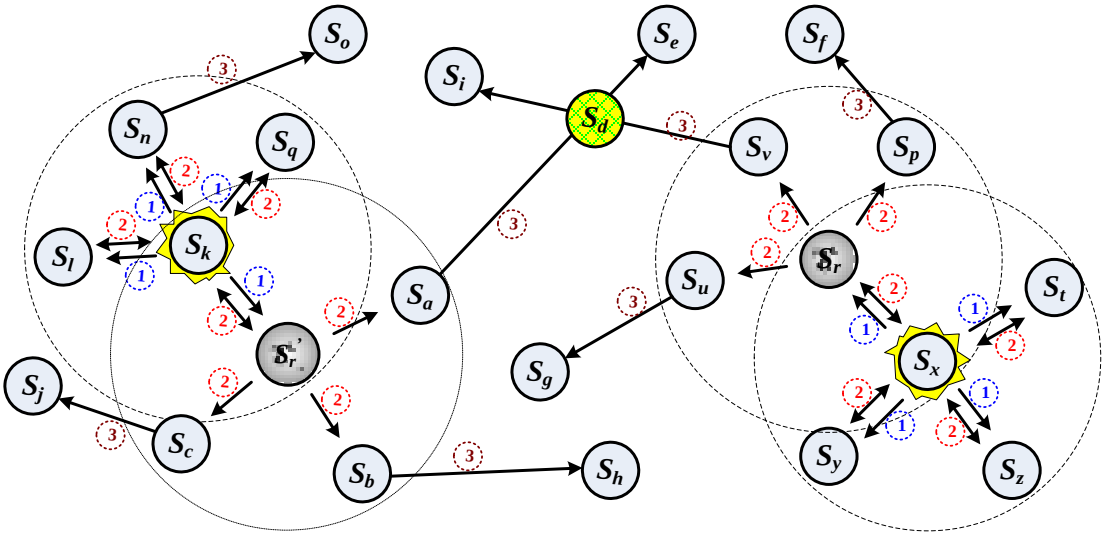


Figure 7.4: Example of detection of CLEA at sensor nodes

To initiate the secure link establishment and detection processes, S_x and S_k's very first step indicated by number 1 is to broadcast their HELLO messages to the neighbourhoods. Second, after confirming the legitimacy of the HELLO messages, each neighbour of S_x (S_k), for example S_r (S_r'), performs the secure link establishment with S_x (S_k). Thereafter, as indicated by number 2, S_x (S_k) and their neighbours involve themselves in the detection process. Specifically, S_r (S_r'), for example, broadcasts its authenticated location claim l_r (l_r') locally. Each of the node's neighbours, S_a (S_v) for instance, then probabilistically forwards l_r (l_r') to the randomly selected witnesses as denoted by number 3. Upon receiving the location claim, each node along the forwarding route, S_d for instance, verifies the signature/MAC on l_r, checks for a conflict with the claims already in its buffer, stores a copy of l_r in its buffers, and then forwards l_r to the witness. If S_d discovers a conflict, i.e. finds another location claim l_r' for the same node ID $S_r' = S_r$ such that $l_r' \neq l_r$, it then floods the network with

unforgeable evidence (the conflicting set of authenticated location claims) of S_r's attempt to launch CLEA. This results in a distributed revocation of S_r.

One might argue that the controlled nodes such as $S_r^{'}$ refuse to follow the scheme (i.e., by not broadcasting its location claim). By this way, the controlled nodes may avoid the conflict of location claims and thus evade the detection. However, acting in this manner, the controlled nodes also miss the only chance of launching CLEA which is their ultimate goal.

There might be the case that the collision takes place at a controlled node. Nevertheless, this does not imply that another collision will not occur at some legitimate nodes in the network. Furthermore, since all scheme decisions are made locally and probabilistically, the attacker cannot anticipate the location of the collision, thus the probability of a collision occurring at a controlled node will be negligible.

7.4.3 Security Analysis and Discussion

In this section, security analysis and discussion are given based on the following questions:

i) What is the probability (P_c) that a witness detects CLEA or the detection rate of the extended scheme?
ii) How does the extended scheme counteract the masked-replication attack?
iii) How does the extended scheme thwart the node revocation attack?
iv) What are other possible security vulnerabilities of the extended scheme?

7.4.3.1 Finding Detection Rate

Suppose that ζ controlled nodes using the same node ID S_λ are dispersed in different locations with ζ authenticated location claims l_1, l_2,…, l_ζ, we would like to find the probability P_c that a witness detects CLEA or in other words, the probability that two conflicting location claims of S_λ are received by the same witness. Note that one collision of two conflicting location claims (l_p and l_q) forms sufficient evidence for the

revocation of all ζ controlled nodes, because the collision triggers a network-wide flood of the duplicate claims and any other node that has heard location claim l_n ($n \neq p,q$) with the same ID S_λ will also revoke S_λ. To determine P_c, note that any location claim is stored at $r.\alpha$ nodes in the network, thus the probability that a node receives the location claim after the kth deployment is $r.\alpha / N(k)$. Hence, the probability $\overline{P_{c_1}}$ that the $r.\alpha$ receivers of claim l_1 do not receive any of the $r.\alpha$ copies of claim v_2 is given by:

$$\overline{P_{c_1}} = \left(1 - \frac{r.\alpha}{N(k)}\right)^{r.\alpha} \tag{1}$$

Likewise, the probability $\overline{P_{c_2}}$ that the $2.r.\alpha$ receivers of claims l_1 and l_2 do not receive any of the $r.\alpha$ copies of claim l_3 is given by:

$$\overline{P_{c_2}} = \left(1 - \frac{2.r.\alpha}{N(k)}\right)^{r.\alpha} \tag{2}$$

Adopting the approach to the birthday problem [99], the probability $\overline{P_c}$ of no collisions among the location claims is calculated by:

$$\overline{P_c} = \prod_{\omega=1}^{\zeta-1}\left(1 - \frac{\omega.r.\alpha}{N(k)}\right)^{r.\alpha} \tag{3}$$

Applying the standard estimation that $e^x \geq 1 + x$ leads to the following inequality:

$$\overline{P_c} \leq \prod_{\omega=1}^{\zeta-1} e^{\frac{-\omega.r^2.\alpha^2}{N(k)}} \tag{4}$$

$$\leq e^{\frac{-r^2.\alpha^2}{N(k)}\sum_{\omega=1}^{\zeta-1}\omega} \tag{5}$$

$$\leq e^{\frac{-r^2.\alpha^2}{N(k)}.\frac{\zeta.(\zeta-1)}{2}} \tag{6}$$

Due to the fact that $N(k) \leq N(t)$, we have:

$$\overline{P_c} \leq e^{\frac{-r^2.\alpha^2}{N(t)}.\frac{\zeta.(\zeta-1)}{2}} \tag{7}$$

Finally, the probability of detecting ζ controlled nodes is given by:

$$P_c \geq 1 - e^{\frac{-r^2.\alpha^2}{N(t)} \cdot \frac{\zeta.(\zeta-1)}{2}} \quad (8)$$

For better interpretation, the relationship among the lower bound of the detection probability of a single duplicate of a controlled node ($P_{c_{lb}}$), the number of line segments per node, and the average number of nodes on each line segments is plotted. The influence of r and α on $P_{c_{lb}}$ is examined in Figures 7.5 and 7.6, respectively, while $N(t)$ varies from 1000 to $10,000$. As shown in the figures, $P_{c_{lb}}$ almost reaches 100% when r and α are greater than 4 and 40 respectively. For example, if $r = 6$, $N(t) = 10,000$, $\alpha = 50$, the extended scheme will detect even a single duplicate of S_λ with probability greater than 99.99%.

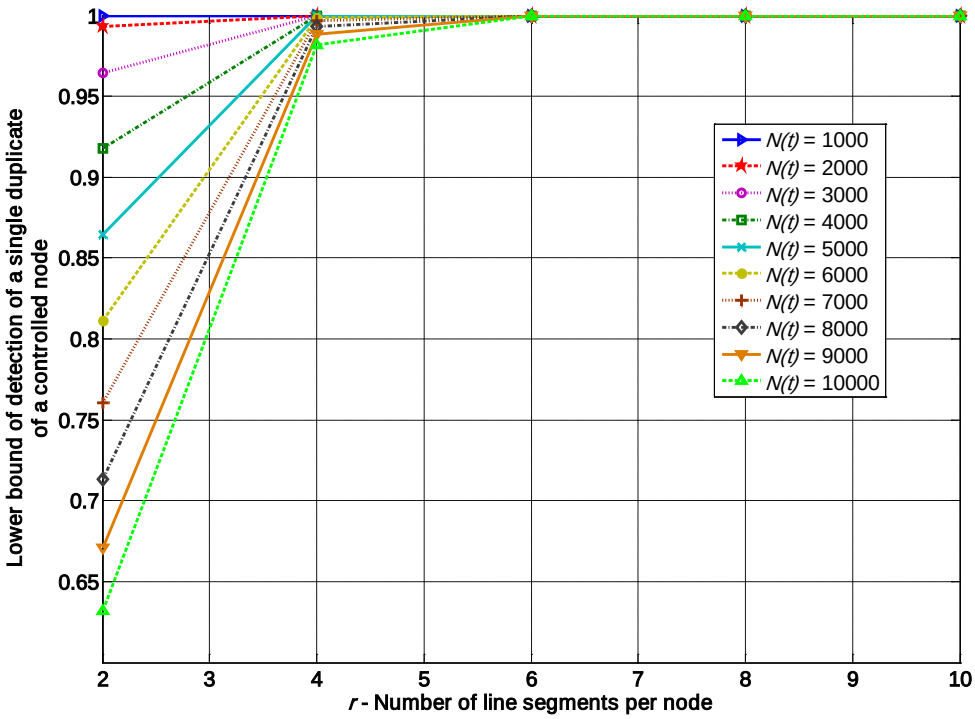


Figure 7.5: Influence of r on $P_{c_{lb}}$ with different network sizes

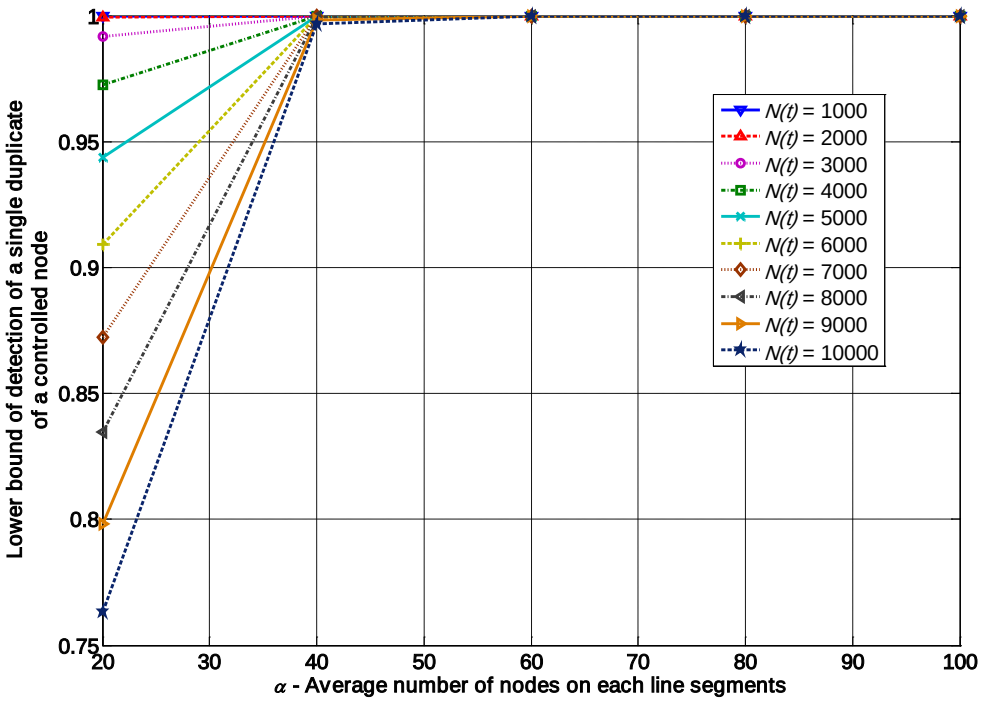


Figure 7.6: Influence of α on $P_{c_{lb}}$ with different network sizes

7.4.3.2 Counteracting Masked-Replication Attack

The masked-replication attack is mentioned in [9] as a major threat to the detection protocols and can be defeated by the pseudo-neighbour nominating technique [9]. In the extended scheme, the probability of a newly deployed node being compromised just after its deployment is negligible. Hence, it is justifiable to assume that at the moment when the neighbours of newly deployed nodes initiate a round of detection, there are at least two legitimate nodes (newly deployed nodes) as the neighbours of two geographically distant controlled nodes which use the same ID. These legitimate nodes continue the detection procedure to detect CLEA. Therefore, the extended scheme is inherently not susceptible to the masked-replication attack without resort to the pseudo-neighbour nominating technique.

One could argue that the masked replication attack can be made possible when the newly deployed node is surrounded by all controlled nodes. Since, in this case, the controlled nodes will suppress the location claim from that node and thus successfully

conceal the attack from being detected. Luckily, this argument is not convincing when this issue is examined in more detail. When the controlled nodes receive the HELLO message from the newly deployed node, they have two options. First, they can refuse to follow the extended scheme by not broadcasting their location claims. The deployed node obviously detects this behaviour since it has not received the location claims from the neighbours over a certain amount of time (including delay induced by unreliable transmission of the location claims). As a result, the deployed node refuses to establish relationship with the no-reply controlled nodes. In other words, by refusing to broadcast their location claims in response to the HELLO message receipt, the controlled nodes refuse the only chance to attack the deployed node via controlled link establishment as well. Second, they have to follow the scheme by broadcasting their location claims to their neighbourhood (or the two-hop neighbours of the newly deployed node). It is easy to see that the masked attack is only successful if all the two-hop neighbours are controlled nodes as well. Now let us assume that the "if" condition is true and the number of nodes in a neighbourhood is about 25 (excluding the nodes sitting in the overlapped regions). Then, the average number of one-hop controlled nodes would be approximately 25. Each of these controlled nodes again would have about 25 controlled nodes in its neighbourhood. Hence, the number of controlled nodes needed to mask the detection initiated by only one newly deployed node is about $25 \times 25 = 625$. This number is not reasonable since it does not match the assumption in Chapter 2. With this number of controlled nodes, apparently all security mechanisms become vulnerable.

7.4.3.3 Thwarting Node Revocation Attack

The extended scheme might face the node revocation attack as they allow sensor nodes to blacklist and revoke nodes which send the outdated HELLO messages. The attacker might try to revoke a legitimate node S_i by recording and later replaying S_i' old HELLO message to a few other sensor nodes. These sensor nodes will expel S_i from the network when they receive the out-of-date message. However, when this attack is investigated closely, it will be found out that the pre-deployed legitimate node S_i cannot be expelled by the attack. There are two cases to be considered. First, the receivers of the S_i's old HELLO message are S_i's neighbours. In this case, the receivers have already believed that S_i is their legitimate neighbour.

Hence, the receipt of the outdated HELLO message from S_i simply gives a signal for the receivers to believe that S_i is in fact being under the attack. Second, the receivers of the S_i's old HELLO message are not S_i's neighbours. In this case, each of the receivers simply checks its neighboring list in its memory. If it finds out that S_i is not on the list, it just simply ignores the HELLO message.

7.4.3.4 Other Security Vulnerabilities

Another security threat to the extended scheme is the jamming attack. It may lead to the failure in the secure link establishment between a requester (newly deployed node) and a responder (pre-deployed or newly deployed node) by monitoring the HELLO message receipt of the responder and then jamming the wireless channel for as short as δT after the node receives the message. Doing that, the attacker can prevent the responder from establishing secure link with the newly deployed sensor nodes. Nonetheless, the jamming attacks are beyond the scope of this thesis. There has been a multitude of countermeasure against the jamming attack on sensor networks in the literature which have been already discussed in chapter 4 such as [88, 89, 155, 156]. For example, the proposed scheme can be combined with the randomized differential direct sequence spread spectrum [156] to provide jamming resistant wireless channels between newly deployed nodes and pre-deployed nodes.

7.5 Evaluations and Further Comparison

In this section, we first evaluate the performance of the proposed schemes using LSM as a benchmark since the idea of LSM has been used as an example to demonstrate the extended scheme in Section 7.4.2. We then compare the proposed approaches with several other approaches in terms of security and performance.

7.5.1 Performance Evaluation

7.5.1.1 Analytical Evaluation

Since the communication overhead of the proposed schemes is overwhelmingly induced by location claim multicast, communication overhead can be roughly defined as the average number of location claims sent and received by nodes in the network. Then, for the entire network, the naïve and adaptive schemes require no communication. Regarding the extended scheme, according to the assumption in Chapter 2, there is neither backward CLEA nor forward CLEA on the first deployment. Therefore, nodes of this deployment incur no communication overhead for location claim multicast. Note that nodes deployed later are to either grow the network or replace failing and unreliable deployed nodes. Therefore, N_i ($i = \overline{2,t}$) is much less than and directly proportional to $N(t)$. It is quite reasonable to assume that N_i equals $O(\sqrt{N(t)})$ because, for example, when $N(t) = 10000$ then N_i is around 100. This is an appropriate value for the purpose of either growing the network or replacing failing nodes. Since each newly deployed node has a fairly small constant number of nodes as their neighbors, then the total number of the neighbors of the newly deployed nodes is estimated to be $O\left(\sqrt{N(t)}\right)$. During the detection process, the extended scheme effectively draws constant r line segments per one location claim broadcast from one neighbor or one newly deployed node and the length of the segment is about $O\left(\sqrt{N(t)}\right)$. As a result, the extended scheme demands $O(N(t))$ communication. All of these overheads are less than that of LSM which grows at a rate of $O(N(t)\sqrt{N(t)})$.

The detection of CLEA involves the verification of the HELLO message in the naïve and adaptive schemes, and the signature/MAC generation and verification of location claims in the extended scheme and LSM. Therefore, the computation overhead is estimated by counting the average number of generation and verification operations in the network. Accordingly, for the entire network, the naïve and adaptive schemes only need $O(N(t))$ message verification while the extended scheme involves $O(N(t))$ authenticated message generation and verification. LSM requires highest computation

which involves generating and verifying public-key signatures and grows at $O(N(t)\sqrt{N(t)})$.

The location claim stored at each node is used as the measurement unit of storage overhead. In this sense, each node in the naïve and adaptive schemes incurs no storage cost while the one in the extended scheme and LSM has to store $\Theta(1)$ and $O(\sqrt{N(t)})$ respectively.

The estimation of performance cost is summarised in Table 7.1 as follows. Note that the communication and computation costs are estimated for the overall network while the storage cost is estimated per node.

Table 7.1: Summary of performance overheads

	Communication	**Computation**	**Storage**
Naïve	0	$O(N(t))^1$	0
Adaptive	0	$O(N(t))^1$	0
Extended	$O(N(t))$	$O(N(t))^2$	$\Theta(1)$
LSM	$O(N(t)\sqrt{N(t)})$	$O(N(l)\sqrt{N(t)})^2$	$O(\sqrt{N(t)})$

1. Message verification; 2. Signature/MAC generation and verification.

7.5.1.2 Simulations

In order to verify the accuracy of the analytical predictions, simulations are conducted to measure the performance requirements of the two last proposed schemes and compared them with LSM. In the simulations, $N(t)$ nodes varying between 1,000 and 10,000 are distributed uniformly. Each node has approximately 40 neighbours on average. Each location claim creates six line segments. For the adaptive and extended schemes, we further assume that $t = 21$, $N_1 = 0.9 \cdot N(t)$, $N_i = (N(t) - N_1)/(t-1)$, $i = \overline{2,t}$. The simulations have been repeated 100 times and the results shown in Figures 7.7, 7.8, 7.9 report the average values.

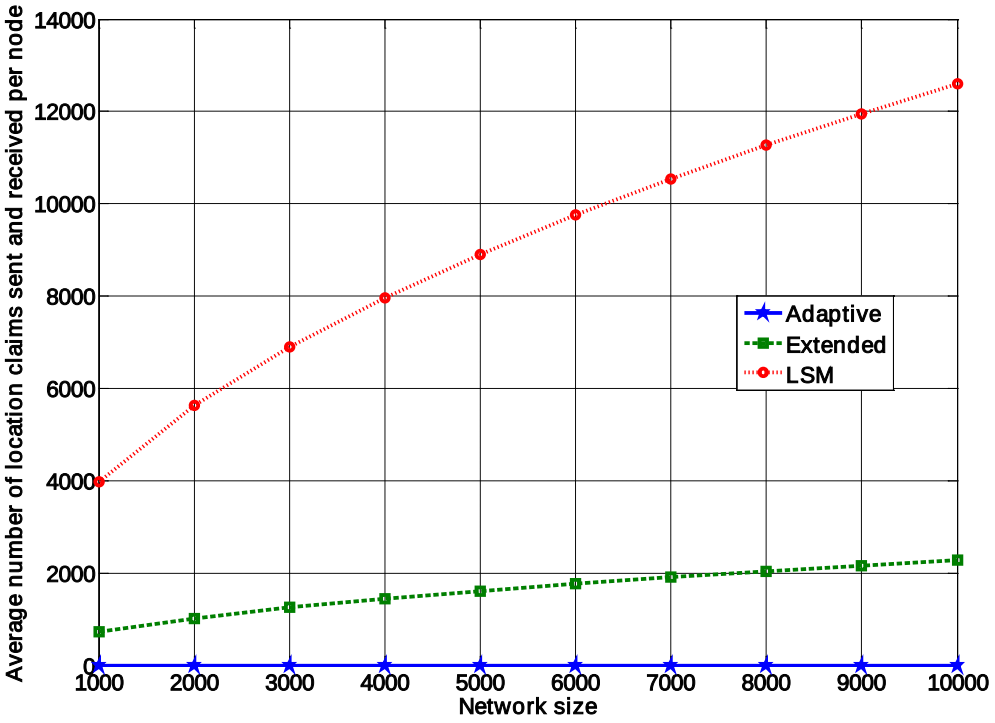


Figure 7.7: Average amount of communication per node in three approaches

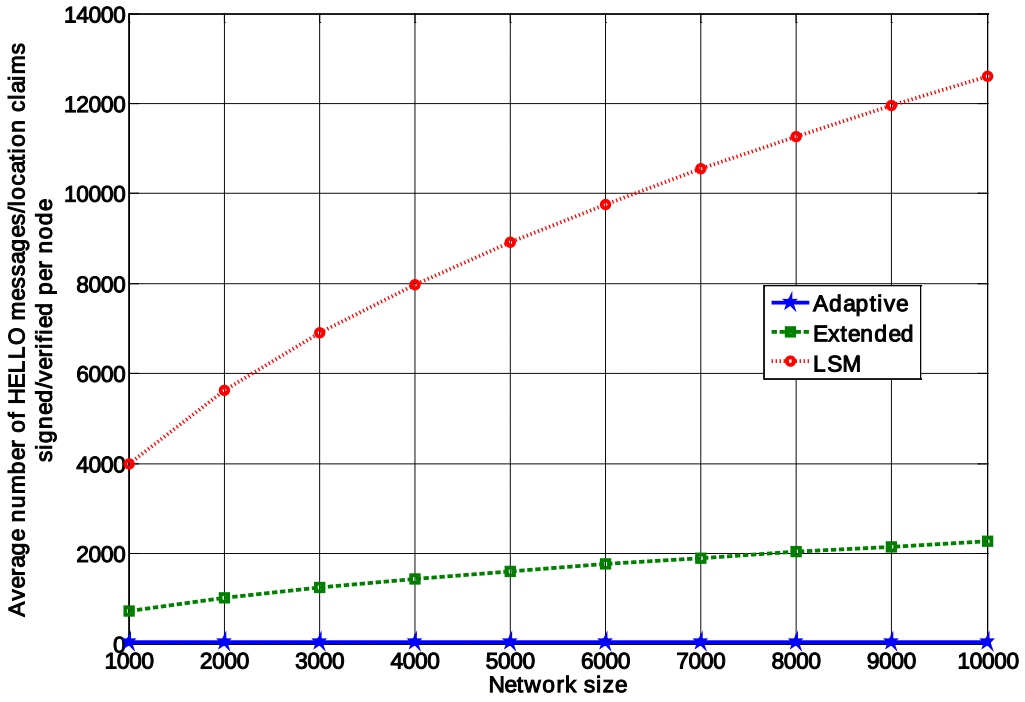


Figure 7.8: Average computation overhead per node in three approaches

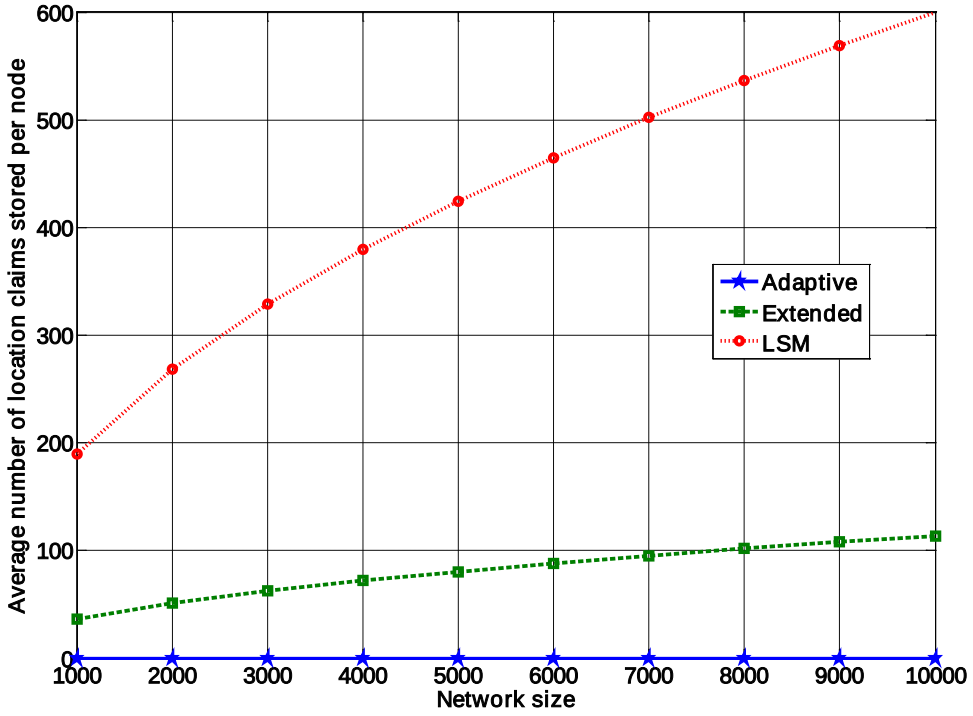


Figure 7.9: Average memory storage per node in three approaches

As shown in the Figures 7.7 to 7.9, the simulations closely match the theoretical anticipation. Note that the communication and computation overheads of LSM in Figures 7.7 and 7.8 is computed for t rounds of detection only. Consequently, if the network lifetime is longer and therefore the number of detection rounds is much greater than t, LSM then incurs a greater communication overhead. From the figures, it can therefore be affirmed that the proposed schemes significantly outperform LSM.

7.5.2 Further Comparison

This section compares the proposed schemes with several other counteractive approaches in terms of performance and security. The results are shown in Table 7.2. Accordingly, the schemes outperform the other approaches: distributed detection approaches [9, 77] (DDs for short), SET [78], Blom based protocol [79] (Blom KPS for short), group deployment knowledge based detection protocol [101] (Group DD for short), localised multicast schemes [102] (Cell DD for short), group deployment knowledge based key pre-distribution schemes [34, 66, 85, 182] (Group KPSs for short),

key infection scheme [55] (Key Infection), and one-way hash chain based protection schemes [56, 57, 183] (PROTECT for short) in many aspects. Firstly, they require a significantly fewer number of nodes to be involved in each detection round and fewer number of detection rounds over the network lifetime in comparison with DDs, SET, Blom KPS, Group DD, Cell DD. As a result, they incur the far lower cost of computation, communication, and memory storage than the other approaches do.

Table 7.2: Comparison with other schemes in terms of security and performance

Property	Naïve & Adaptive (Ours)	Extended (Ours)	DDs	SET	Blom KPS	Group DD	Cell DD	Group KPSs	Key Infection	PROTECT (Ours)
Early & real-time prevention	√	√						√		√
Early detection & revocation		√								
Distributed	√	√	√			√	√	N/A	N/A	N/A
Number of nodes involved	Small	Medium	All	All	All	Medium	Medium	N/A	N/A	N/A
Number of detection rounds	Small	Small	Large	Large	Large	Large	Large	N/A	N/A	N/A
Location information		√	√			√	√			
Deployment & topology knowledge						√	√	√		
Independent of PKE schemes	√	√	√				√			√
Network scalability	√	√		√	√	√	√	√	√	√
Time sync.	√	√	√			√	√			
Efficiency	High	Medium	Low	Low	Low	Low	Low	High	High	High

Secondly, the proposed schemes provide better network scalability than do DDs. They can easily accommodate new nodes added to the network without delay, whereas as for DDs there is some undesirable delay in accommodating and requiring new nodes to follow the detection process in the network. This delay enables the controlled nodes to either launch CLEA between time slots or refuse to follow the protocols at the specified time. Thirdly, unlike Blom-KPS and SET, the proposed schemes adopt the fully distributed detection approach to CLEA, thus they neither introduce a single point of

failure nor cause undue communication burden among the nodes surrounding the base station that results in the shortened network lifetime. Fourthly, the proposed schemes work very well with any types of network/node deployment and topology. Meanwhile, Group DD, Cell DD and Group KPSs are dependent upon a very particular assumption of deployment knowledge and network topology which is not always applicable. Fifth, the proposed schemes are more secure than Key Infection because they require deployed nodes to verify the legitimacy of new nodes before admitting them to be their neighbours. Meanwhile, in Key Infection, new nodes can join the network freely by simply initiating the request for the key establishment with deployed nodes. Finally, all of the proposed schemes provide early and real-time attack prevention. Furthermore, the extended scheme allows for early attack detection and revocation. In contrast, these features are not available in DDs, SET, Blom KPS, Group DD and Cell DD. Meanwhile the attack detection and revocation is not provided in Group KPSs and PROTECT.

Although the proposed schemes exhibit various advanced features, they also have some limitations which exists in other approaches as well. All of the schemes have to depend on secure time synchronisation which is also the necessity for DDs, Group DD, and Cell DD. The extended scheme needs the location information for the detection, so do the DDs, Group DD, and Cell DD.

7.6 Chapter Remarks

This chapter introduced a family of evolutionary schemes for defence against and detection of CLEA and revocation of controlled nodes. It started with the naïve scheme as a conduit for the development of the adaptive and extended schemes. The adaptive scheme exhibits greatly desired performance at the cost of somewhat weaker security resilience while the extended scheme trades off little performance for greater security gains which include controlled node revocation and detection of both backward and forward CLEAs. The plausibility of the proposed schemes in respect of security features and performance overheads has been demonstrated through analytical analyses, simulations, and extensive comparison with other schemes.

Chapter 8

Conclusions and Future Work

This chapter begins with a high level summary of the main themes and contributions of this thesis. Despite the presented contributions, there remain a few research issues for future work. These issues are described in the second section of this chapter.

8.1 Conclusions

In this dissertation, four related themes with respect to CLEA on distributed sensor networks have been studied as follows:

- In the first theme, we first provided a brief introduction to DSNs together with their advantageous characteristics as well as limitations. We then presented a few specific security-critical and real-world applications to emphasise the necessity of security measures for the success of DSN applications. Lastly, we examined the factors that facilitate CLEA in DSNs. The first factor is that DSNs are usually left unattended after deployment. Due to this unattended condition, the attacker can surreptitiously penetrate into the network to capture a limited number of nodes. The second factor is the requirement for sensor nodes to meet low-cost and resource-constrained features for large-scale and cost-effective deployments of DSNs. Because of this requirement, sensor nodes usually lack tamper-proof hardware. Consequently, the attacker can easily compromise captured nodes to extract secret information. The secret information disclosure renders traditional security mechanisms such as encryption and authentication ineffective under CLEA. The third factor is the attacker's capability to abuse the compromised nodes and secret information repetitively without being detected. The outcome of this examination is the following set of features used to identify

and characterise the attack: prerequisite for node compromise, limit on the number of compromised nodes, repetitive and multiple usage of compromised nodes and secret information, control of large portion of secured links, and attack via controlled nodes.

- For the second theme, two literature reviews of existing key establishment schemes and countermeasures against CLEA were conducted which established that CLEA is a serious threat and emphasised the necessity of new countermeasures. The first review produced a taxonomy of key establishment schemes for DSNs. Based on the taxonomy, it classified the key establishment schemes according to their level of vulnerability to CLEA. The classification indicates that no key establishment schemes are totally immune from CLEA. Because the key establishment schemes are usually used as the building block of the other security measures, their vulnerability definitely will lead to the susceptibility of the entire security system. The second review started with the description of NRA and KSCA as the concrete instances of CLEA followed by existing indirect and direct countermeasures against these two attacks which may be utilised to thwart CLEA. This review indicated that each of these countermeasures has its own limitations which make it ineffective against CLEA.
- The third theme explored fundamental security services in DSNs. They are cryptographic primitives, message authentication, DoS prevention for broadcast authentication, anti-jamming techniques, secure localisation, and secure time synchronisation. For each service, we carried out a brief survey of its up-to-date research results. These results can be used as building blocks for the development of the new countermeasures against CLEA.
- The fourth theme investigated the enabling factors of CLEA to establish that CLEA is a real security threat to DSNs. The investigation first considered the feasibility of node compromise. It has been found out that in most cases, if not all, the attacker can compromise sensor nodes easily due to the following reasons: i) most of the current sensor hardware has been designed and implemented without acknowledging these weaknesses and vulnerabilities in the design requirements for each node; ii) most sensor nodes are comprised of “off the shelf” components and programmed using widely available open-source software; iii) tamper resistant hardware has yet to be available on sensor nodes. Even when protected by anti-tamper technologies, sensor nodes are still

susceptible wide variety of the physical tampering methods such as non-invasive attacks, semi-invasive attacks, and invasive attacks. The investigation then discussed the methods for repetitive and multiple use of compromised nodes and secret information. They are secret information cloning, short-distance swapping, long-distance swapping, mixed-distance swapping. Lastly, the investigation made the first successful implementation of a CLEA prototype. This implementation uses four MICAz motes using the short-distance swapping method, the random-pairwise keys scheme in the TinyOS environment.

- The fifth theme proposed a series of the secret information protection-based schemes. These schemes creatively applied the one-way hash chain to the protection of secret information pre-loaded in sensor nodes. Due to this protection, the attacker cannot obtain secret information to establish controlled links with legitimate sensor nodes and thus will fail to launch CLEA. The proposed schemes exhibit a number of appealing features: applicability to any key establishment schemes, robustness, independence of any special requirements, and the guarantee of network scalability. Their efficiency and plausibility are well justified through extensive analyses, simulations, and implementation on the current sensor platform.
- The last theme developed a family of the attack detection-based schemes. They rely on two types of counter values for the attack detection. The first counter value is maintained by each sensor node internally while the second one is sent via a link establishment request to receiving sensor nodes. Over a specific interval, the attack is detected when the two values are not matched. The last scheme also has the ability to identify and revoke the controlled nodes or the source of attack. The efficiency and plausibility of the proposed schemes in respect of security features and performance overheads has been demonstrated through analyses, simulations, and extensive comparison with other schemes.

In summary, the main contributions of this thesis are as follows:

- A comprehensive survey of key establishment schemes which are vulnerable to CLEA.
- A comprehensive survey of existing countermeasures against CLEA and the identification of their limitations.

- A detailed review of hardware tamper-proof technologies as well as their vulnerability.
- A detailed description of how compromised nodes and secret information can be utilised repetitively in multiple times at different locations and the prototype implementation of CLEA in the form of KSCA.
- Three new defensive schemes against CLEA using secret information protection.
- Three new active scheme against CLEA via controlled node detection and revocation.

8.2 Future Work

The development of the detection-based schemes needs a multicast authentication scheme where each sensor node can send authenticated messages to a number of selected sensor nodes in the network. Based on the discussion of message authentication in DSNs we concluded that no existing multicast authentication schemes are as effective and efficient as desired. Therefore, to further support the detection-based approach, we intend to develop a new multicast authentication scheme which leverages the symmetric-key cryptographic techniques to achieve optimal performance in both security and overheads.

As discussed in section 7.4.3.4, the detection-based approach might be susceptible to the jamming attack due to their dependence of the short time window δT . This dependence can be relaxed in order to provide the approach with resistance to jamming attacks by integrating the detection-based approach with the protection-based approach. This integration is left as future work.

Finally, the schemes proposed in this dissertation work efficiently and robustly based on the underlying assumption that all sensor nodes in DSNs are static after their deployment. However, in environments where sensor nodes are mobile, the detection-based approach becomes ineffective. Therefore, we plan to extend our detection-based schemes and develop new algorithms for mobile sensor networks.

References

[1] M. Ilyas and I. Mahgoub, *Handbook of sensor networks: compact wireless and wired sensing systems*: CRC Press, Boca Raton, USA, 2005.

[2] System Energy Efficiency Lab, "CitiSense - Adaptive Services for Community-Driven Behavioral and Environmental Monitoring to Induce Change," San Diego, http://seelab.ucsd.edu/health/overview.shtml, August 17, 2010.

[3] S. Churchill, "CitiSense: Cellular Environmental Monitoring," DailyWireless.org, San Diego, http://www.dailywireless.org/2009/12/12/citisence-cellular-environmental-monitoring/, August 17, 2010.

[4] S. LOHR, "Smart Dust? Not Quite, but We're Getting There," The New York Times, http://www.nytimes.com/2010/01/31/business/31unboxed.html, August 17, 2010.

[5] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks,* vol. 38, pp. 393-422, 2002.

[6] K. Romer and F. Mattern, "The design space of wireless sensor networks," *IEEE Wireless Communications,* vol. 11, pp. 54-61, 2004.

[7] K. Pister, J. Kahn, B. Boser, and S. Morris, "SMART DUST Autonomous sensing and communication in a cubic millimeter," http://robotics.eecs.berkeley.edu/~pister/SmartDust/, August 17, 2010.

[8] TechNewsDaily, "World's Smallest Solar-Powered Sensor Runs Almost Forever," http://www.technewsdaily.com/worlds-smallest-solar-powered-sensor-runs-almost-forever-0188/, August 18, 2010.

[9] B. Parno, A. Perrig, and V. Gligor, "Distributed Detection of Node Replication Attacks in Sensor Networks," in *Proc. IEEE symp. on Security and Privacy*, pp. 49-63, 2005.

[10] e-SENSE website, "Capturing Ambient Intelligence for Mobile Communications through Wireless Sensor Networks," http://www.ist-esense.org/, August 18, 2010.

[11] P. Casari, A. Castellani, A. Cenedese, C. Lora, M. Rossi, L. Schenato, and M. Zorzi, "The "Wireless Sensor Networks for City-Wide Ambient Intelligence (WISE-WAI)" Project," *Sensors,* vol. 9, pp. 4056-4082, 2009.

[12] L. Benini, E. Farella, and C. Guiducci, "Wireless sensor networks: Enabling technology for ambient intelligence," *Microelectronics Journal,* vol. 37, pp. 1639-1649, 2006.

[13] E. Aarts and S. Marzano, *The new everyday: views on ambient intelligence*. Rotterdam: OIO Publishers, 2003.

[14] E. H. L. Aarts, J. M. Rabaey, and W. Werner, *Ambient intelligence*: Springer, 2005.

[15] Q. Wang, Y. Zhu, and L. Cheng, "Reprogramming wireless sensor networks: challenges and approaches," *IEEE Network,* vol. 20, pp. 48-55, May/June 2006.

[16] A. Dunkels, N. Finne, J. Eriksson, and T. Voigt, "Run-time dynamic linking for reprogramming wireless sensor networks," in *Proc. 4th int'l conf. on Embedded Networked Sensor Systems (SenSys'06)*, Boulder, Colorado, USA, November 2006.

[17] UC Berkeley and MLB Co, "Tracking vehicles with a UAV-delivered sensor network," http://robotics.eecs.berkeley.edu/~pister/29Palms0103/, August 18, 2010.

[18] B. J. FEDER, "Wireless Sensor Networks Spread to New Territory," The New York Times, http://www.nytimes.com/2004/07/26/business/26sensor.html, August 18, 2010.

[19] A. Ledeczi, "Shooter Localization," http://w3.isis.vanderbilt.edu/projects/nest/applications.html, August 18, 2010.

[20] Crossbow Technology Inc., "MICA2 Datasheet," https://www.eol.ucar.edu/rtf/facilities/isa/internal/CrossBow/DataSheets/mica2.pdf, August 18, 2010.

[21] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler, "TinyOS: An Operating System for Wireless Sensor Networks," in *Ambient Intelligence*, W. Weber, J. Rabaey, and E. Aarts, Eds. Berlin: Springer, 2005.

[22] M. Maróti, P. Völgyesi, S. Dóra, B. Kusý, A. Nádas, Á. Lédeczi, G. Balogh, and K. Molnár, "Radio interferometric geolocation," in *Proc. 3rd int'l conf. on Embedded Networked Sensor Systems*, San Diego, California, USA, 2005.

[23] B. Kusy, A. Ledeczi, M. Maroti, and L. Meertens, "Node density independent localization," in *Proc. 5th int'l conf. on Information Processing in Sensor Networks*, Nashville, Tennessee, USA, 2006.

[24] J. Pershall, C. Loxley, and J. Strong, "Shell and HP to Develop Ultrahigh-resolution Seismic Sensing Solution," HP, LONDON, http://www.hp.com/hpinfo/newsroom/press/2010/100215xa.html, August 18, 2010.

[25] Fraunhofer-Gesellschaft, "Sniffing out terrorists," Physorg.com, http://www.physorg.com/news182602309.html, August 18, 2010.

[26] R. D. Pietro, L. V. Mancini, and A. Mei, "Random Key-assignment for Secure Wireless Sensor Networks," in *Proc. 1st ACM workshop on Security of Ad hoc and Sensor Networks (SASN '03)*, Fairfax, Virginia, 2003.

[27] L. Eschenauer and V. Gligor, "A Key-management Scheme for Distributed Sensor Networks," in *Proc. 9th ACM conf. on Computer and Communications Security*, Washington, DC, USA, pp. 41-47, 2002.

[28] H. Chan, A. Perrig, and D. Song, "Random Key Predistribution Schemes for Sensor Networks," in *Proc. IEEE symp. on Security and Privacy*, pp. 197-213, 2003.

[29] W. Du, J. Deng, Y. S. Han, P. K. Varshney, J. Katz, and A. Khalili, "A Pairwise Key Pre-distribution Scheme for Wireless Sensor Networks," *ACM Trans. Inf. Syst. Secur.*, vol. 8, pp. 228-258, 2005.

[30] D. Liu, P. Ning, and R. Li, "Establishing Pairwise Keys in Distributed Sensor Networks," *ACM Trans. Inf. Syst. Secur.*, vol. 8, pp. 41-77, 2005.

[31] T. D. Tran and C. S. Hong, "Efficient ID-based Threshold Random Key Pre-distribution Scheme for Wireless Sensor Networks," *IEICE Trans. Communications*, vol. E91-B, pp. 2602-2609, August 2008.

[32] A. K. Das, "An efficient random key distribution scheme for large-scale distributed sensor networks," *Security and Communication Networks*, vol. Early View, p. n/a, 2009.

[33] S.-P. Chan, R. Poovendran, and M.-T. Sun, "A key management scheme in distributed sensor networks using attack probabilities," in *Proc. IEEE Global Telecom. conf. (GLOBECOM '05)*, St. Louis, MO, USA, pp. 1007-1011, 2005.

[34] W. Du, J. Deng, Y. S. Han, S. Chen, and P. K. Varshney, "A Key Management Scheme for Wireless Sensor Networks Using Deployment Knowledge," in *Proc. IEEE INFOCOM 2004*, pp. 586-597, 2004.

[35] W. Zhang, M. Tran, S. Zhu, and G. Cao, "A Random Perturbation-Based Scheme for Pairwise Key Establishment in Sensor Networks," in *Proc. 8th ACM*

int'l symp. on Mobile Ad hoc Networking and Computing, Montreal, Quebec, Canada, pp. 90-99, 2007.

[36] F. Delgosha and F. Fekri, "A multivariate key-establishment scheme for wireless sensor networks," *Trans. Wireless. Comm.*, vol. 8, pp. 1814-1824, 2009.

[37] D. J. Malan, M. Welsh, and M. D. Smith, "A public-key infrastructure for key distribution in TinyOS based on elliptic curve cryptography," in *Proc. 1st annual IEEE Communications Society conf. on Sensor and Ad Hoc Communications and Networks (SECON'04)*, pp. 71-80, 2004.

[38] C. Lederer, R. Mader, M. Koschuch, J. Großschädl, A. Szekely, and S. Tillich, "Energy-Efficient Implementation of ECDH Key Exchange for Wireless Sensor Networks," in *Proc. 3rd IFIP WG 11.2 int'l workshop on Information Security Theory and Practice, Smart Devices, Pervasive Systems, and Ubiquitous Networks*, pp. 112-127, August 2009.

[39] S. Khajuria and H. Tange, "Implementation of Diffie-Hellman key exchange on wireless sensor using elliptic curve cryptography," in *Proc. 1st int'l conf. on Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology (Wireless VITAE 2009)*, pp. 772-776, 2009.

[40] J. Großschädl, A. Szekely, and S. Tillich, "The energy cost of cryptographic key establishment in wireless sensor networks," in *Proc. 2nd ACM symp. on Information, Computer and Communications Security*, Singapore, 2007.

[41] T. Moore, "A Collusion Attack on Pairwise Key Predistribution Schemes for Distributed Sensor Networks," in *Proc. 4th IEEE int'l conf. on Pervasive Computing and Communications (PerCom'06)*, 2006.

[42] P. Tague and R. Poovendran, "Modeling adaptive node capture attacks in multi-hop wireless networks," *Ad Hoc Networks*, vol. 5, pp. 801-814, 2007.

[43] P. Tague and R. Poovendran, "Modeling node capture attacks in wireless sensor networks," in *Proc. 46th annual Allerton conf. on Communication, Control, and Computing*, Illinois, USA, pp. 1221-1224, 2008.

[44] C. Hartung, J. Balasalle, and R. Han, "Node compromise in sensor networks: The need for secure systems," University of Colorado at Boulder January 2005.

[45] J. R. Douceur, "The sybil attack," in *Proc. int'l workshop on Peer-to-Peer Systems*, pp. 251-260, 2002.

[46] A. Becher, Z. Benenson, and M. Dornseif, "Tampering with Motes: Real-World Physical Attacks on Wireless Sensor Networks," in *Proc. 3rd int'l conf. Security in Pervasive Computing (SPC'06)*, pp. 104-118, 2006.

[47] C. Krauß, M. Schneider, and C. Eckert, "On handling insider attacks in wireless sensor networks," *Inf. Secur. Tech. Rep.*, vol. 13, pp. 165-172, 2008.

[48] Y. C. Hu, M. Jakobsson, and A. Perrig, "Efficient Constructions for One-Way Hash Chains," in *Applied Cryptography and Network Security*, pp. 423-441, 2005.

[49] Crossbow Technology Inc., "MICAz datasheet," http://www.openautomation.net/uploadsproductos/micaz_datasheet.pdf, August 18, 2010.

[50] Y. W. Law, "Key management and link-layer security of wireless sensor networks : Energy-efficient attack and defense," in *Electrical Engineering, Mathematics and Computer Science* Enschede, Netherlands: University of Twente, p. 136, 2005.

[51] R. Watro, D. Kong, S.-F. Cuti, C. Gardiner, C. Lynn, and P. Kruus, "TinyPK: securing sensor networks with public key technology," in *Proc. 2nd ACM workshop on Security of Ad hoc and Sensor Networks (SASN '04)*, Washington DC, USA, pp. 59-64, 2004.

[52] S. Zhu, S. Setia, and S. Jajodia, "LEAP: efficient security mechanisms for large-scale distributed sensor networks," in *Proc. 10th ACM conf. on Computer and Communications Security (CCS'03)*, Washington D.C., USA, pp. 62-72, 2003.

[53] S. Zhu, S. Setia, and S. Jajodia, "LEAP+: Efficient Security Mechanisms for Large-Scale Distributed Sensor Networks," *ACM Trans. Sen. Netw.*, vol. 2, pp. 500-528, 2006.

[54] D. Dolev and A. C. Yao, "On the security of public key protocols," *IEEE Trans. Information Theory*, vol. 29, pp. 198-208, 1983.

[55] R. Anderson, H. Chan, and A. Perrig, "Key Infection: Smart Trust for Smart Dust," in *Proc. 12th IEEE int'l conf. on Network Protocols (ICNP'04)*, pp. 206-215, 2004.

[56] T. D. Tran and J. I. Agbinya, "A framework for confronting key-swapping collusion attack on random pairwise key pre-distribution schemes for distributed sensor networks," in *Proc. 5th IEEE int'l conf. on Mobile Ad Hoc and Sensor Systems (MASS'08)*, Atlanta, Georgia, USA, pp. 815-820, 2008.

[57] T. D. Tran and J. I. Agbinya, "Combating key-swapping collusion attack on random pairwise key pre-distribution schemes for wireless sensor networks," *Security and Communication Networks, in press.*

[58] W. Zhang and G. Cao, "Group Rekeying for Filtering False Data in Sensor Networks: A Predistribution and Local Collaboration-Based Approach," in *Proc. IEEE INFOCOM 2005*, pp. 503-514, 2005.

[59] L. Sang and A. Arora, "Spatial Signatures for Lightweight Security in Wireless Sensor Networks," in *Proc. IEEE INFOCOM*, pp. 2137-2145, 2008.

[60] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone, *Handbook of applied cryptography*: CRC Press, 1996.

[61] M. Eltoweissy, M. Moharrum, and R. Mukkamala, "Dynamic key management in sensor networks," *IEEE Communications Magazine*, vol. 44, pp. 122-130, 2006.

[62] R. Rivest, "The MD5 Message Digest Algorithm," *RFC 1321, Internet Engineering Task Force*, 1992.

[63] D. Eastlake and P. Jones, "US secure hash algorithm 1 (SHA1)," *RFC 3174, Internet Engineering Task Force*, Sep. 2001.

[64] D. Huang, M. Mehta, D. Medhi, and L. Harn, "Location-aware key management scheme for wireless sensor networks," in *Proc. 2nd ACM workshop on Security of Ad hoc and Sensor Networks*, Washington DC, USA, pp. 29-42, 2004.

[65] F. Anjum, "Location dependent key management using random key-predistribution in sensor networks," in *Proc. 5th ACM workshop on Wireless Security*, Los Angeles, California, pp. 21-30, 2006.

[66] D. Liu and P. Ning, "Improving Key Predistribution with Deployment Knowledge in Static Sensor Networks," *ACM Trans. Sen. Netw.*, vol. 1, pp. 204-239, 2005.

[67] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler, "SPINS: security protocols for sensor networks," *Wireless Network*, vol. 8, pp. 521-534, 2002.

[68] M. F. Younis, K. Ghumman, and M. Eltoweissy, "Location-Aware Combinatorial Key Management Scheme for Clustered Sensor Networks," *IEEE Trans. Parallel and Distributed Systems*, vol. 17, pp. 865-882, 2006.

[69] A. S. Wander, N. Gura, H. Eberle, V. Gupta, and S. C. Shantz, "Energy analysis of public-key cryptography for wireless sensor networks," in *Proc. 3rd IEEE*

int'l conf. on Pervasive Computing and Communications (PerCom'05), pp. 324-328, 2005.

[70] J. Brown, X. Du, and K. Nygard, "An Efficient Public-Key-Based Heterogeneous Sensor Network Key Distribution Scheme," in *Proc. IEEE Global Telecomm. conf. (GLOBECOM'07)*, Washington, DC, USA, pp. 991-995, 2007.

[71] G. Gaubatz, J. P. Kaps, E. Ozturk, and B. Sunar, "State of the art in ultra-low power public key cryptography for wireless sensor networks," in *Proc. 3rd IEEE int'l conf. on Pervasive Computing and Communications Workshops (PerCom'05)*, pp. 146-150, 2005.

[72] A. Liu and P. Ning, "TinyECC: A Configurable Library for Elliptic Curve Cryptography in Wireless Sensor Networks," in *Proc. 7th int'l conf. on Information Processing in Sensor Networks (IPSN '08)*, St. Louis , Missouri , USA, pp. 245-256, 2008.

[73] Q. Jing, J. Hu, and Z. Chen, "C4W: An Energy Efficient Public Key Cryptosystem for Large-Scale Wireless Sensor Networks," in *Proc. IEEE int'l conf. on Mobile Adhoc and Sensor Systems (MASS'06)* pp. 827-832, 2006.

[74] C. Haowen, V. D. Gligor, A. Perrig, and G. Muralidharan, "On the distribution and revocation of cryptographic keys in sensor networks," *IEEE Trans on Dependable and Secure Computing*, vol. 2, pp. 233-247, 2005.

[75] C. Castelluccia and A. Spognardi, "RoK: A robust key pre-distribution protocol for multi-phase wireless sensor networks," in *Proc. 3rd int'l conf. on Security and Privacy in Communications Networks and Workshops (SecureComm 2007)*, Nice, France, pp. 351-360, 2007.

[76] Y. Zhou and Y. Fang, "A Two-Layer Key Establishment Scheme for Wireless Sensor Networks," *IEEE Trans. Mobile Computing*, vol. 6, pp. 1009-1020, 2007.

[77] M. Conti, R. D. Pietro, L. V. Mancini, and A. Mei, "A randomized, efficient, and distributed protocol for the detection of node replication attacks in wireless sensor networks," in *Proc. 8th ACM int'l symp. on Mobile Ad hoc Networking and Computing*, Montreal, Quebec, Canada, 2007.

[78] H. Choi, S. Zhu, and T. F. La Porta, "SET: Detecting node clones in sensor networks," in *Proc. 3rd int'l conf. on Security and Privacy in Communications Networks and Workshops (SecureComm 2007)*, 2007.

[79] R. Brooks, P. Y. Govindaraju, M. Pirretti, N. Vijaykrishnan, and M. T. Kandemir, "On the Detection of Clones in Sensor Networks Using Random Key Predistribution," *IEEE Trans. Systems, Man, and Cybernetics - Part C: Applications and Reviews,* 2007.

[80] H. Fu, S. Kawamura, M. Zhang, and L. Zhang, "Replication Attack on Random Key Pre-distribution Schemes for Wireless Sensor Networks," *Computer Communications,* vol. 31, pp. 842-857, 2008.

[81] C. Karlof and D. Wagner, "Secure routing in wireless sensor networks: attacks and countermeasures," *Ad Hoc Networks,* vol. 1, pp. 293-315, 2003.

[82] K.-W. Fan, S. Liu, and P. Sinha, "Structure-Free Data Aggregation in Sensor Networks," *IEEE Trans. Mobile Computing,* vol. 6, pp. 929-942, 2007.

[83] C. Hua and T.-S. P. Yum, "Optimal Routing and Data Aggregation for Maximizing Lifetime of Wireless Sensor Networks," *IEEE/ACM Trans. Networking,* vol. 16, pp. 892-903, 2008.

[84] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, "Directed diffusion for wireless sensor networking," *IEEE/ACM Trans. Networking,* vol. 11, pp. 2-16, 2003.

[85] D. Liu and P. Ning, "Location-based pairwise key establishments for static sensor networks," in *Proc. 1st ACM workshop on Security of Ad hoc and Sensor Networks*, 2003.

[86] T. Ito, H. Ohta, N. Matsuda, and T. Yoneda, "A key pre-distribution scheme for secure sensor networks using probability density function of node deployment," in *Proc. 3rd ACM workshop on Security of Ad hoc and Sensor Networks (SASN '05),* Alexandria, VA, USA, 2005.

[87] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks," in *Proc. IEEE INFOCOM,* pp. 1567-1576 vol.1563, 2002.

[88] W. Xu, K. Ma, W. Trappe, and Y. Zhang, "Jamming sensor networks: attack and defense strategies," *IEEE Network,* vol. 20, pp. 41-47, May-June 2006.

[89] A. Mpitziopoulos, D. Gavalas, C. Konstantopoulos, and G. Pantziou, "A survey on jamming attacks and countermeasures in WSNs," *IEEE Communications Surveys & Tutorials,* vol. 11, pp. 42-56, 2009.

[90] A. Alarifi and W. Du, "Diversify sensor nodes to improve resilience against node compromise," in *Proc. 4th ACM workshop on Security of Ad hoc and Sensor Networks (SASN'06)*, Alexandria, Virginia, USA, pp. 101-112, 2006.

[91] M. J. Atallah, E. D. Bryant, and M. R. Stytz, "A survey of anti-tamper technologies," *The Journal of Defense Software Engineering*, pp. 12-16, 2004.

[92] S. Blythe, B. Fraboni, S. Lall, H. Ahmed, and U. de Riu, "Layout reconstruction of complex silicon chips," *IEEE Journal of Solid-State Circuits*, vol. 28, pp. 138-145, 1993.

[93] R. J. Anderson, "Why cryptosystems fail," *Commun. ACM*, vol. 37, pp. 32-40, 1994.

[94] R. J. Anderson and M. Kuhn, "Tamper Resistance - a Cautionary Note," in *Proc. 2nd USENIX workshop on Electronic Commerce*, pp. 1–11, 1996.

[95] R. J. Anderson and M. G. Kuhn, "Low Cost Attacks on Tamper Resistant Devices," in *Proc. 5th int'l workshop on Security Protocols*, pp. 125-136, 1998.

[96] H. Handschuh, P. Paillier, and J. Stern, "Probing Attacks On Tamper-Resistant Devices," in *Cryptographic Hardware and Embedded Systems*, pp. 727-727, 1999.

[97] S. Skorobogatov, "Low temperature data remanence in static RAM," *University of Cambridge, Computer Laborary, Technical Report UCAM-CL-TR-536*, vol. 536, June, 2002.

[98] D. Samyde, S. Skorobogatov, R. Anderson, and J.-J. Quisquater, "On a New Way to Read Data from Memory," in *Proc. 1st int'l IEEE Security in Storage workshop*, 2002.

[99] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*: MIT press, 2001.

[100] B. Karp and H. T. Kung, "GPSR: greedy perimeter stateless routing for wireless networks," in *Proc. 6th annual int'l conf. on Mobile Computing and Networking (MobiCom)*, Boston, Massachusetts, United States, 2000.

[101] J.-W. Ho, D. Liu, M. Wright, and S. K. Das, "Distributed detection of replica node attacks with group deployment knowledge in wireless sensor networks," *Ad Hoc Networks*, vol. 7, pp. 1476-1488, 2009.

[102] B. Zhu, S. Setia, S. Jajodia, S. Roy, and L. Wang, "Localized Multicast: Efficient and Distributed Replica Detection in Large-Scale Sensor Networks," *IEEE Trans. Mobile Computing*, vol. 9, pp. 913-926, 2010.

[103] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker, "GHT: a geographic hash table for data-centric storage," in *Proc. 1st ACM int'l workshop on Wireless Sensor Networks and Applications (WSNA'02)*, Atlanta, Georgia, USA, pp. 78-87, 2002.

[104] S. Zhu, S. Setia, S. Jajodia, and P. Ning, "Interleaved hop-by-hop authentication against false data injection attacks in sensor networks," *ACM Trans. Sen. Netw.*, vol. 3, p. 14, 2007.

[105] M. Bellare, R. Canetti, and H. Krawczyk, "Keying Hash Functions for Message Authentication," in *Proc. 16th annual int'l conf. on Advances in Cryptology*, 1996.

[106] B. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, pp. 422-426, 1970.

[107] L. Fan, P. Cao, J. Almeida, and A. Z. Broder, "Summary cache: a scalable wide-area web cache sharing protocol," *IEEE/ACM Trans. Netw.*, 2000.

[108] R. R. Brooks, *Disruptive Security Technologies with Mobile Code and Peer-to-Peer Networks*. Boca Raton, FL, USA: CRC Press, Inc., 2004.

[109] Z. Li and G. Gong, "Randomly directed exploration: An efficient node clone detection protocol in wireless sensor networks," in *Proc. IEEE int'l conf. on Mobile Adhoc and Sensor Systems (MASS '09)*, Macau SAR, China, pp. 1030-1035, 2009.

[110] J.-W. Ho, M. Wright, and S. K. Das, "Fast Detection of Replica Node Attacks in Mobile Sensor Networks Using Sequential Analysis," in *Proc. IEEE INFOCOM*, Rio de Janeiro, Brazil, pp. 1773-1781, 2009.

[111] A. Wald, *Sequential analysis*. New York: Dover Publications, 2004.

[112] Y. Zeng, J. Cao, S. Zhang, S. Guo, and L. Xie, "Random-walk based approach to detect clone attacks in wireless sensor networks," *IEEE J. Selected Areas in Communications*, vol. 28, pp. 677-691, 2010.

[113] J. Polastre, R. Szewczyk, and D. Culler, "Telos: enabling ultra-low power wireless research," in *Proc. 4th int'l symp. on Information Processing in Sensor Networks*, Los Angeles, California, pp. 364-369 2005.

[114] J. Schiller, A. Liers, H. Ritter, R. Winter, and T. Voigt, "ScatterWeb - Low Power Sensor Nodes and Energy Aware Routing," in *Proc. 38th Hawaii int'l conf. on System Sciences*, pp. 1-9, 2005.

[115] C. M. Maunder and R. E. Tulloss, *The test access port and boundary scan architecture*. Los Alamitos, California: IEEE Computer Society Press, 1990.

[116] R. Szewczyk and D. Gay, "Debugging nesC code with the AVR JTAG ICE," http://www.tinyos.net/tinyos-1.x/doc/nesc/debugging.html, August 18, 2010.

[117] O. Kömmerling and M. G. Kuhn, "Design principles for tamper-resistant smartcard processors," in *Proc. USENIX workshop on Smartcard Technology (Smartcard '99)*, Chicago, Illinois, pp. 9-20, 1999.

[118] S. P. Skorobogatov and R. J. Anderson, "Optical Fault Induction Attacks," in *Proc. 4th int'l wrkshop on Cryptographic Hardware and Embedded Systems*, pp. 2-12, 2003.

[119] J. MARKOFF, "Vulnerability Is Discovered in Security for Smart Cards," The New York Times, SAN FRANCISCO, http://www.nytimes.com/2002/05/13/technology/13SMAR.html?pagewanted=all, August 18, 2010.

[120] Y. C. Hu, A. Perrig, and D. B. Johnson, "Packet leashes: a defense against wormhole attacks in wireless networks," in *Proc. IEEE INFOCOM*, pp. 1976-1986, 2003.

[121] R. Poovendran and L. Lazos, "A graph theoretic framework for preventing the wormhole attack in wireless ad hoc networks," *Wirel. Netw.*, vol. 13, pp. 27-59, 2007.

[122] J. Spencer, *The Strange Logic of Random Graphs* vol. 22: Springer-Verlag, 2000.

[123] Cygwin, "What Is Cygwin?," http://www.cygwin.com/, August 18, 2010.

[124] Sun Microsystems, "Sun Java," http://java.sun.com/, August 18, 2010.

[125] Sun Microsystems, "Java Communications," http://java.sun.com/products/javacomm/, August 18, 2010.

[126] D. Gay, P. Levis, R. v. Behren, M. Welsh, E. Brewer, and D. Culler, "The nesC language: A holistic approach to networked embedded systems," in *Proc. ACM SIGPLAN 2003 conf. on Programming Language Design and Implementation*, San Diego, California, USA, pp. 1-11, 2003.

[127] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, pp. 120-126, 1978.

[128] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of Computation*, vol. 48, pp. 203-209, 1987.

[129] V. Miller, "Use of elliptic curves in cryptography," *Advances in Cryptology (CRYPTO'85), LNCS*, vol. 218, pp. 417-462, 1985.

[130] G. d. Meulenaer, F. Gosset, F.-X. Standaert, and O. Pereira, "On the Energy Cost of Communication and Cryptography in Wireless Sensor Networks," in *Proc. IEEE int'l conf. on Wireless & Mobile Computing, Networking & Communication (WiMob'08)*, 2008.

[131] Crossbow, "Wireless Sensor Networks Product Reference Guide," Crossbow Technology Inc., 2007.

[132] E. F. Brickell, D. E. Denning, S. T. Kent, D. P. Maher, and W. Tuchman, "Skipjack Review, Interim Report: The Skipjack Algorithm," July 1993.

[133] R. Rivest, "The RC5 encryption algorithm," in *Proc. 2nd int'l workshop on Fast Software Encryption*, Leuven, Belgium, pp. 86-96, 1994.

[134] D. E. Standard, "Federal Information Processing Standards Publication 46," *National Bureau of Standards, US Department of Commerce*, 1977.

[135] J. Daemen and V. Rijmen, *The design of Rijndael: AES--the advanced encryption standard*: Springer Verlag, 2002.

[136] J.-P. Kaps, G. Gaubatz, and B. Sunar, "Cryptography on a Speck of Dust," *Computer*, vol. 40, pp. 38-44, 2007.

[137] M. Y. Rhee, *Internet Security: Cryptographic Principles, Algorithms and Protocols*: Wiley, 2003.

[138] T. Heer, S. Götz, O. G. Morchon, and K. Wehrle, "ALPHA: an adaptive and lightweight protocol for hop-by-hop authentication," in *Proc. int'l conf. On Emerging Networking Experiments And Technologies (ACM CoNEXT '08)*, Madrid, Spain, 2008.

[139] W. Du, R. Wang, and P. Ning, "An efficient scheme for authenticating public keys in sensor networks," in *Proc. 6th ACM int'l symp. on Mobile Ad hoc Networking and Computing*, Urbana-Champaign, IL, USA, pp. 58-67, 2005.

[140] R. C. Merkle, "Protocols for Public Key Cryptosystems," in *Proc. IEEE symp. on Security and Privacy*, Oakland, CA, USA, p. 122, 1980.

[141] M. O. Rabin, "Digitalized Signatures and Public-Key Functions as Intractable as Factorization," Massachusetts Institute of Technology, Cambridge, MA, United States 1979.

[142] C. P. Schnorr, "Efficient signature generation by smart cards," *Journal of Cryptology*, vol. 4, pp. 161-174, 1991.

[143] NIST, "Digital Signature Standard (DSS) (FIPS 186-3)," U. S. D. o. Commerce, Ed. Gaithersburg, MD, USA: Federal Information Processing Standards Publications, 2009.

[144] D. Johnson, A. Menezes, and S. Vanstone, "The elliptic curve digital signature algorithm (ECDSA)," *Int'l Journal Info. Sec.*, vol. 1, pp. 36-63, 2001.

[145] T. Elgamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Trans. Info. Theo.*, vol. 31, pp. 469-472, 1985.

[146] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Proc. Advances in Cryptology (CRYPTO 84)*, Santa Barbara, California, United States, pp. 47-53, 1985.

[147] J. M. David, W. Matt, and D. S. Michael, "Implementing public-key infrastructure for sensor networks," *ACM Trans. Sen. Netw.*, vol. 4, pp. 1-23, 2008.

[148] M. Bellare, C. Namprempre, and G. Neven, "Security Proofs for Identity-Based Identification and Signature Schemes," in *Advances in Cryptology (EUROCRYPT 2004)*. vol. 3027 Interlaken, Switzerland: Springer Berlin, pp. 268-286, 2004.

[149] X. Cao, W. Kou, L. Dang, and B. Zhao, "IMBAS: Identity-based multi-user broadcast authentication in wireless sensor networks," *Comput. Commun.*, vol. 31, pp. 659-667, 2008.

[150] Y. Huang, W. He, K. Nahrstedt, and W. C. Lee, "DoS-resistant broadcast authentication protocol with low end-to-end delay," in *Proc. IEEE INFOCOM workshops*, pp. 1-6, 2008.

[151] R. Wang, W. Du, and P. Ning, "Containing denial-of-service attacks in broadcast authentication in sensor networks," in *Proc. 8th ACM int'l symp. on Mobile Ad hoc Networking and Computing*, Montreal, Quebec, Canada, pp. 71 - 79, 2007.

[152] Q. Dong, D. Liu, and P. Ning, "Pre-authentication filters: providing dos resistance for signature-based broadcast authentication in sensor networks," in *Proc. 1st ACM conf. on Wireless Network Security (WiSec'08)*, Alexandria, VA, USA, 2008.

[153] S. Zhu, S. Xu, S. Setia, and S. Jajodia, "LHAP: A Lightweight Hop-by-Hop Authentication Protocol for Ad-Hoc Networks," in *Proc. 23rd int'l conf. on Distributed Computing Systems*, Providence, Rhode Island, USA, 2003.

[154] L. C. Baird, W. L. Bahn, M. D. Collins, M. C. Carlisle, and S. C. Butler, "Keyless Jam Resistance," in *Proc. IEEE Information Assurance and Security Workshop (IAW '07)*, pp. 143-150, 2007.

[155] C. Pöpper, M. Strasser, and S. Čapkun, "Jamming-resistant broadcast communication without shared keys," in *Proc. 18th USENIX Security symp.*, Montreal, Canada, 2009.

[156] Y. Liu, P. Ning, H. Dai, and A. Liu, "Randomized Differential DSSS: Jamming-Resistant Wireless Broadcast Communication," in *Proc. IEEE INFOCOM*, San Diego, CA, USA, pp. 1-9, 2010.

[157] D. Liu, P. Ning, and W. K. Du, "Attack-resistant location estimation in sensor networks," in *Proc. 4th int'l symp. Information Processing in Sensor Networks (IPSN'05)*, California, USA, pp. 99-106, 2005.

[158] D. Liu, P. Ning, and W. Du, "Detecting Malicious Beacon Nodes for Secure Location Discovery in Wireless Sensor Networks," in *Proc. 25th int'l conf. Distributed Computing Systems (ICDCS'05)*, Ohio, USA, pp. 609-619, 2005.

[159] T. Park and K. G. Shin, "Attack-tolerant localization via iterative verification of locations in sensor networks," *ACM Trans. Embed. Comput. Syst.*, vol. 8, pp. 1-24, 2008.

[160] Z. Li, W. Trappe, Y. Zhang, and B. Nath, "Robust statistical methods for securing wireless localization in sensor networks," in *Proc. 4th int'l symp. on Information Processing in Sensor Networks (IPSN '05)*, UCLA, Los Angeles, California, USA, pp. 91-98, 2005.

[161] D. Liu, P. Ning, A. Liu, C. Wang, and W. K. Du, "Attack-resistant location estimation in wireless sensor networks," *ACM Trans. Information and System Security (TISSEC)*, vol. 11, p. 22, 2008.

[162] L. Fang, W. Du, and P. Ning, "A beacon-less location discovery scheme for wireless sensor networks," in *Proc. IEEE INFOCOM*, Miami, FL, USA pp. 161-171, 2005.

[163] L. Lazos, R. Poovendran, and S. Capkun, "ROPE: robust position estimation in wireless sensor networks," in *Proc. 4th int'l symp. Information Processing in Sensor Networks (IPSN'05)*, UCLA, Los Angeles, California, USA, pp. 324-331, 2005.

[164] Y. Yang and Y. Sun, "Securing Time-Synchronization Protocols in Sensor Networks: Attack Detection and Self-Healing," in *Proc. Global Telecomm. conf. (GLOBECOM '08)*, Miami, Florida, USA, pp. 1-6, 2008.

[165] S. Ganeriwal, S. Čapkun, C.-C. Han, and M. B. Srivastava, "Secure time synchronization service for sensor networks," in *Proc. 4th ACM workshop on Wireless Security*, Cologne, Germany, 2005.

[166] S. Ganeriwal, Christina Pöpper, Srdjan Čapkun, and M. B. Srivastava, "Secure Time Synchronization in Sensor Networks," *ACM Trans. Inf. Syst. Secur.*, vol. 11, pp. 1-35, 2008.

[167] H. Song, S. Zhu, and G. Cao, "Attack-resilient time synchronization for wireless sensor networks," *Ad Hoc Networks*, vol. 5, pp. 112-125, 2007.

[168] K. Sun, P. Ning, and C. Wang, "TinySeRSync: secure and resilient time synchronization in wireless sensor networks," in *Proc. 13th ACM conf. on Computer and Communications Security (CCS'06)*, Alexandria, Virginia, USA, 2006.

[169] X. Hu, T. Park, and K. G. Shin, "Attack-Tolerant Time-Synchronization in Wireless Sensor Networks," in *Proc. IEEE INFOCOM*, Phoenix, Arizona, USA, pp. 41-45, 2008.

[170] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: accurate and scalable simulation of entire TinyOS applications," in *Proc. 1st int'l conf. on Embedded Networked Sensor Systems*, Los Angeles, California, USA, 2003.

[171] UbiSec&Sens, "Ubiquitous Sensing and Security in the European Homeland," http://www.ist-ubisecsens.org/index.php, August 18, 2010.

[172] S. Marti, T. J. Giuli, K. Lai, and M. Baker, "Mitigating routing misbehavior in mobile ad hoc networks," in *Proc. annual int'l conf. on Mobile Computing and Networking (ACM MobiCom '00)*, Boston, Massachusetts, USA, pp. 255-265, 2000.

[173] T. D. Tran and J. I. Agbinya, "Early and Lightweight Distributed Detection of Node Replication Attack in Sensor Networks," in *Proc. IEEE Wireless Communications and Networking Conference 2010 (WCNC 2010)*, Sydney, Australia, 2010.

[174] L. Doherty, K. S. J. Pister, and L. E. Ghaoui, "Convex position estimation in wireless sensor networks," in *Proc. IEEE INFOCOM*, pp. 1655-1663, 2001.

[175] J. Newsome and D. Song, "GEM: Graph EMbedding for routing and data-centric storage in sensor networks without geographic information," in *Proc. 1st int'l conf. on Embedded Networked Sensor Systems (SenSys '03)*, pp. 76-88, 2003.

[176] A. Caruso, S. Chessa, S. De, and A. Urpi, "GPS free coordinate assignment and routing in wireless sensor networks," in *Proc. IEEE INFOCOM*, Miami, FL, USA pp. 150-160, 2005.

[177] S. Čapkun and J.-P. Hubaux, "Secure positioning in wireless networks," *IEEE J. Selected Areas in Communications*, vol. 24, pp. 221-232, 2006.

[178] C.-M. Yu, C.-S. Lu, and S.-Y. Kuo, "A constrained function based message authentication scheme for sensor networks," in *Proc. Wireless Communications and Networking Conference (IEEE WCNC '09)*, Budapest, Hungary, pp. 2180-2185, 2009.

[179] H. Chen, "Efficient compromising resilient authentication schemes for large scale wireless sensor networks," in *Proc. 3rd ACM conf. on Wireless Network Security*, Hoboken, New Jersey, USA, pp. 49-54, 2010.

[180] C. Cocks, "An identity based encryption scheme based on quadratic residues," *Cryptography and Coding*, vol. 2260, pp. 360-363, 2001.

[181] F. Hess, "Efficient identity based signature schemes based on pairings," in *Proc. 9th annual int'l workshop on Selected Areas in Cryptography*, St. John's, Newfoundland, Canada, pp. 310-324, 2002.

[182] Y. Zhen and G. Yong, "A key pre-distribution scheme using deployment knowledge for wireless sensor networks," in *Proc. 4th int'l symp. on Information Processing in Sensor Networks*, Los Angeles, CA, USA, pp. 261-268, 2005.

[183] T. D. Tran and J. I. Agbinya, "Revisiting Key-swapping Collusion Attack on Distributed Sensor Networks," in *Proc. 4th int'l conf. on Sensor Technologies and Applications (SENSORCOMM 2010)*, Venice, Italy, pp. 381-388, 2010.

[184] M.J.B. Robshaw, "Stream Ciphers," *RSA Laboratories Technical Report TR-701*, version 2.0, July, 1995.

[185] R.L. Rivest, " The RC4 Encryption Algorithm," *RSA Data Security, Inc.*, March 1992.

[186] P. Rogaway and D. Coppersmith, "A software-optimized encryption algorithm," in *Proc. Fast Software Encryption – Cambridge Security Workshop*, Cambridge, U.K., pp. 56-63, 1994.

[187] T. Seigenthaler, "Decrypting a class of stream ciphers using ciphertext only," *EEE Trans. Computers*, vol. C-34, no. 1, pp. 81-85, 1985.

[188] K. Zeng, C.H. Yang, D.Y. Wei, and T.R.N. Rao, "On the linear consistency test in cryptanalysis with applications," in *Proc. Advances in Cryptology – Crypto'89*, pp. 167-174, 1990.

[189] K. Zeng, C.H. Yang, D.Y. Wei, and T.R.N. Rao, "An improved linear syndrome algorithm in cryptanalysis with applications," in *Proc. Advances in Cryptology – Crypto'90*, pp. 34-47, 1990.

[190] J. Golić, "Linear cryptanalysis of stream ciphers," in *Proc. Fast Software Encryption*, vol. 1008, pp. 154-169, 1995.

Appendix

Glossary

In this thesis, the following terms are defined and used:

AES	Advanced Encryption Standard
ALPHA	Adaptive and Lightweight Protocol for Hop-by-hop Authentication
ATSP	Attack-tolerant Time Synchronisation Protocol
BS	Base Station
CBC	Cipher Block Chaining encryption mode
CBC-MAC	Cipher Block Chaining Message Authentication Code
CFB	Cipher FeedBack encryption mode
CLEA	Controlled Link Establishment Attack
COTS	Commercial Off-The-Shelf
CSS	Chirp Spread Spectrum
DES	Data Encryption Standard
DOWHCBS	Diversified One-Way Hash Chain Based Scheme
DSA	Digital Signature Algorithm
DSN	Distributed Sensor Network
DSSS	Direct Sequence Spread Spectrum
ECB	Electronic Code Book encryption mode
ECC	Elliptic Curve Cryptography
ESMIS	Exclusive Subset Maximal Independent Set
ECDH	Elliptic Curve Diffie–Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
ECIES	Elliptic Curve Integrated Encryption Scheme
FHSS	Frequency Hopping Spread Spectrum

HOWHCBS	Hidden One-Way Hash Chain Based Scheme
IBC	Identity-Based Cryptography
ID	Sensor node identifier
IDEA	International Data Encryption
IED	Improvised Explosive Device
KEK	Key Encryption Key
KPS	Key Pre-distribution Scheme
KRK	Key Recovery Key
KSCA	Key-Swapping Collusion Attack
LED	Light-Emitting Diode
LM	Localised Multicast
LSM	Line-Selected Multicast
MAC	Medium Access Control, Message Authentication Code
MARS	MARS encryption algorithm
MD4	Message Digest algorithm 4
MD5	Message Digest algorithm 5
MEMS	Micro-ElectroMechanical System
NRA	Node Replication Attack
OFB	Output FeedBack encryption mode
OMAC	One-key Message Authentication Code
OWHC	One-Way Hash Chain
OWHCBS	One-Way Hash Chain Based Scheme
OWHF	One-Way Hash Function
PKC	Public Key Cryptography
PKE	Pairwise Key Establishment
PKG	Private Key Generator
PMAC	Parallelizable Message Authentication Code
RAM	Random-Access Memory
RAWL	RAndom WaLk
RC5	RC5 encryption algorithm
RD-DSSS	Randomised Differential Direct Sequence Spread Spectrum
RED	Randomised, Efficient, and Distributed protocol
RM	Randomised Multicast

ROM	Read-Only Memory
RSA	Rivest-Shamir-Adleman algorithm
SEAL	Software-optimized Encryption ALgorithm
SEK	Secret Encryption Key
SET	SET operation based detection scheme
SHA	Secure Hash Algorithm
SKE	Symmetric Key Cryptography
SLDR	Subset LeaDeR
SPN	Substitution-Permutation Network
SPRT	Sequential Probability Radio Test
TDMA	Time Division Multiple Access
TRAWL	Table-assisted Random WaLk
TTL	Time To Live
UAV	Unmanned Aerial Vehicle
UDSSS	Uncoordinated Direct Sequence Spread Spectrum