Secure and Efficient Dealing with Node Capture Attack in Wireless Sensor Networks

by

SARITA AGRAWAL
201121013

A Thesis Submitted in Partial Fulfilment of the Requirements for the Degree of

DOCTOR OF PHILOSOPHY
in
INFORMATION AND COMMUNICATION TECHNOLOGY
to

DHIRUBHAI AMBANI INSTITUTE OF INFORMATION AND COMMUNICATION TECHNOLOGY

August, 2017

Declaration

I hereby declare that

i) the thesis comprises of my original work towards the degree of Doctor of Philosophy in Information and Communication Technology at Dhirubhai Ambani Institute of Information and Communication Technology and has not been submitted elsewhere for a degree,

ii) due acknowledgement has been made in the text to all the reference material used.

Sarita Agrawal

Certificate

This is to certify that the thesis work entitled SECURE AND EFFICIENT DEALING WITH NODE CAPTURE ATTACK IN WIRELESS SENSOR NETWORKS has been carried out by SARITA AGRAWAL for the degree of Doctor of Philosophy in Information and Communication Technology at *Dhirubhai Ambani Institute of Information and Communication Technology* under my supervision.

Prof. Manik Lal Das
Thesis Supervisor

Acknowledgements

I express my deepest gratitude to my spiritual Masters, whose continuous support gave me the strength to sail through this journey. I would ever be in debt to my parents for all their sacrifices, motivation and blessings. This award truly belongs to my father, who always encouraged me to face life like a soldier, and to my late mother, who always stood beside me and helped me go beyond my fears.

I am grateful to my supervisor Prof. Manik Lal Das, who believed in me and, guided me at every step systematically and patiently. His dedication, sincerity and generosity inspired me to reach at this stage. If luck is to be believed on, I would say, I am lucky to be the first research scholar of such a supervisor.

My sincere thanks to Prof. Anish Mathuria, who has an unassuming way of providing most valuable guidance in such a simple and witty manner. I feel lucky to receive guidance from Prof. Sanjay Srivastava who has always amazed me by his simplicity and clarity. I would like to acknowledge our Ex-Director Prof. Nagaraj Ramrao, Director Prof. K S Dasgupta, Dean AP - Prof. Suman Mitra, Dean Research and Development - Prof. Sanjeev Gupta and all faculty members. I thank our Registrar Mr. Soman Nair and all the staff members of DAIICT especially Ms. Deepa Poduval.

Further, I would like to thank Prof. Javier Lopez, Dr Rodrigo Roman and their team at University of Malaga, Spain for their fruitful discussion through collaboration.

I am grateful to Prof. Veena Bansal (IIT Kanpur), Prof. Ajay Jain (IIT Kanpur) and Prof. Renu Jain (Kanpur University), whose constant support encouraged me to continue my work despite all odds.

A very special thanks to my friend Seema Devra who never looked at the clock to

support me in all my endeavor.

I would like to acknowledge Mr. Narendra Yadav and his family for devoting their time and efforts to look after my parents in my absence.

I am grateful to many families of Heartfulness Institute, specially Late Mr. Rajesh Agrawal and Mr. H. L. Khar, the families of Mr. Naresh Goswami, Mrs. Kumudini Kolhe, Mr. Prakash Patel, Mr. Mukesh Barot, Mrs. Sushila Contractor, Mr. Yaman Saluja and Mrs. Renu Chaturvedi, who supported me as my extended family. I also thank Mrs Anna-marie Beaud, Ms. Dominique Veinante, Dr Catherine Choukroun and Dr. Jose Quesada and all members of Heartfulness institute in Europe, for their guidance and care during my Spain visit. I thank Mrs. Lourdes Comitre and her family who made my stay in Malaga so comfortable.

I would like to acknowledge my sister Savita Gautam and brother-in-law Ram Gautam and sisterly friends G. Keerthika, Neha Sisodiya, Dr. Rina Kumari. I thank Dr Birju Acharya, Dr. Varsha Dave and Dr. Deepak Bhandari. Last but not the least, I thank all my co-researchers in DAIICT, especially Milind Padalkar, Naveen Kumar and Ram Naresh Vangala for their support. And, thanks to the girl gang of my young researcher friends - Vandana Ravindran, Archana Nigam, Nidhi Desai, Nupur Jain, Miral Shah and Purvi Patel who helped me keep my youthfulness alive.

I thank one and all, who directly or indirectly contributed in fulfilling my dream.

Contents

Abstract **ix**

List of Symbols and Acronyms **xii**

List of Tables **xiii**

List of Figures **xiv**

1 Introduction **1**

1.1 Overview of Wireless Sensor Networks (WSNs) 2

1.1.1 Sensor Node Architecture 3

1.1.2 WSN Architecture 5

1.1.3 WSN Standards 6

1.1.4 WSN Applications 7

1.2 Security Issues in WSN 9

1.3 Motivation 11

1.4 Contribution of the Thesis 13

1.4.1 Framework of Solution to Deal with Node Capture Attack . 15

1.4.2 System and Network Model used in the Proposed Solution 17

1.5 Thesis Outline 18

2 Background and Preliminaries **21**

2.1 Overview of Security in WSN 21

2.1.1 Resilience 24

2.1.2 Key Management in WSN 27

2.1.3 Node Capture Detection and Revocation 35

2.2 Primitives Used in the Proposed Protocols 39
2.2.1 Pseudo Random Function 39
2.2.2 Bivariate Polynomials 40
2.2.3 ID based Public Key Infrastructure with Bilinear Pairings . . 40
2.2.4 Chinese Remainder Theorem 42
2.3 Trusted Platform Module 43
2.4 Conclusion 45

3 Pair-Wise Key Establishment and Key Update 46
3.1 Introduction 47
3.2 Polynomial Share Based Pair-Wise Key Establishment 49
3.3 Proposed Pair-Wise Key Establishment and Key Update Protocol . 56
3.3.1 Goals and Assumptions 56
3.3.2 Set-up and Initialization 57
3.3.3 Node Discovery and Node Authentication 57
3.3.4 Session Key Update 59
3.3.5 Security Features 60
3.3.5.1 Forward Secrecy 62
3.3.5.2 Resistance to Impersonation Attack 63
3.3.5.3 Resisting Known-key Attacks 64
3.3.5.4 Resistance to Replay Attacks 65
3.3.5.5 Resilience to Node Capture 67
3.3.5.6 Resilience to Worm hole and Sink hole Attacks . . 69
3.3.6 Comparing Performance with Existing Protocols 70
3.3.7 Experimental Results 71
3.4 Conclusion 74

4 Self-Healing and Mutual-Healing enabled Group Key Distribution 76
4.1 Introduction 76
4.2 Self Healing 77
4.3 Mutual Healing 81
4.4 Proposed Bilinear Pairing based Healing Protocol 82

4.4.1 Goals and Assumptions 82
4.4.2 Session Key Management 83
4.4.2.1 System Set-up 83
4.4.2.2 Group Key Broadcast 83
4.4.2.3 Authentication and Key Extraction 84
4.4.3 Healing 85
4.4.4 Security Analysis 88
4.4.5 Performance Analysis 91
4.4.5.1 Computation cost 91
4.4.5.2 Communication cost 92
4.4.5.3 Storage cost 93
4.5 CRT based Symmetric Key Healing Protocol 93
4.5.1 System Model 93
4.5.2 Session Key Management 94
4.5.2.1 System Setup 95
4.5.2.2 Group Key Message Construction and Distribution 95
4.5.2.3 Authentication and Key Extraction 96
4.5.3 Healing 97
4.5.4 Security Analysis 99
4.5.5 Performance Analysis 104
4.5.5.1 Computation cost 104
4.5.5.2 Communication cost 105
4.5.5.3 Storage cost 105
4.6 Comparison with Existing Schemes 105
4.6.1 Security Features 105
4.6.2 Performance 106
4.7 Experimental Results 107
4.8 Conclusion 110

5 Node Capture Attack **112**
5.1 Introduction 113
5.2 Identifying Node Capture by Monitoring 114

5.3 Software and Hardware Attestation 118
5.4 Program Integrity Verification 121
5.5 Trusted Platform Module Enabled Program Integ-rity Verification (TPIV) Protocol for Node Capture Detection 124
5.5.1 Goals and Assumptions 125
5.5.2 TPIV Setup and Monitoring 126
5.5.3 Authentication and Code Verification 127
5.5.4 Security Strengths 128
5.5.4.1 High Probability of Node Capture Detection 130
5.5.4.2 Node Capture Detection by Authorized Verifier . . 132
5.5.4.3 Secrecy of Non-captured Nodes 133
5.5.4.4 Comparing TPIV with Existing Schemes 134
5.5.5 Efficiency and Experimental Results 134
5.5.5.1 Analytical Comparison 134
5.5.5.2 Improvement in Node Capture Detection Probability 135
5.5.5.3 Experimental Results 136
5.6 Conclusion 138

6 Node Revocation and Key Update 141
6.1 Introduction 141
6.2 Centralized Approach to Node Revocation 142
6.3 Distributed Voting Mechanism for Revoking a Victim Node 144
6.4 Hybrid Node Revocation Methods 147
6.5 Proposed Protocol for Node Revocation and Key Update (NRKU) 152
6.5.1 Goals and Assumptions 152
6.5.2 Initial Session Setup 152
6.5.3 Node Revocation and Key Update 153
6.5.3.1 Revocation Message Broadcast 154
6.5.3.2 Authentication and Session Key Update 154
6.5.4 Security Strengths 156
6.5.4.1 Secure Node Revocation 157
6.5.4.2 Forward and Backward Secrecy 158

6.5.4.3 Resistance to Node Collusion Attack 160
6.5.4.4 Resistance to Impersonation and Replay Attacks . 161
6.5.4.5 Existing Revocation Protocols v/s NRKU 161
6.5.5 Performance Boost with NRKU 162
6.6 Conclusion 164

7 Conclusion and Future Work **166**

Appendix 1: ProVerif Tool **187**

Appendix 2: Publications **196**

Abstract

Wireless Sensor Networks (WSN) have found enormous applications in various areas of day-to-day life such as in health-care, battle-field surveillance and disaster management. The communication amongst the sensor nodes within a WSN takes place on an unreliable wireless channel. Therefore, the nodes are vulnerable to various security attacks such as eavesdropping and message replay attack. Typically, the sensor nodes in a WSN are mostly deployed in unattended areas that render the nodes vulnerable to physical attacks. Node capture attack is one of the most precarious attacks that allows an adversary to physically capture, reprogram and redeploy a node in the network to carry out other malicious activities such as routing or cloning attacks and may badly hamper the normal functionality of the network. In this thesis, we address the node capture attack with a secure and efficient solution framework that comprises of a set of protocols for secure key establishment, detection of node capture attack and revocation of a victim of node capture attack. For secure key establishment, we worked on pair-wise key establishment and key update and propose a protocol that use multiple polynomial shares based master secret and update the pair-wise key for each session using random inputs from the pair of nodes involved. We then propose self-healing and mutual-healing enabled group key distribution protocols. First, we present a protocol using bilinear pairing and then a protocol that uses Chinese remainder theorem (CRT) based secret sharing. Detection of node capture attack is carried out using program integrity verification of suspect node by cluster heads equipped with trusted platform module (TPM). To revoke a node capture victim, we propose a node revocation and key update protocol.

We used analytical reasoning, theorem proving technique and formal analysis

with ProVerif tool to analyze the security of the proposed protocols. The analysis reveals that the pair-wise session-key establishment and key update protocol is capable of resisting impersonation, replay, known-key, sink-hole and worm-hole attacks. The protocol also ensures key freshness, mutual-key control and forward secrecy and, provides high resilience to node capture attack. The group key distribution protocols, proposed for secret key sharing within a group of nodes, equip the sensor nodes so they can recover one or more missing broadcasts from a future broadcast using self-healing. Even the missed broadcast for the current session can be obtained with the help of a neighbor node using mutual-healing. The proposed protocols ensure that only the group members authorized to take part in a session can recover the key for that session using self-healing or mutual-healing. The protocols do not allow an unauthorized neighbor to respond to a mutual-healing request. The node capture detection protocol in the proposed solution framework detects a victim of node capture attack with very high probability even when an additional memory is put into the captured node. The node capture detection protocol also ensures that the probability of a captured node revealing the personal secret of any non-captured node is negligible. The protocol allows only an authorized verifier to carry out the program integrity verification for a node suspected to be a victim of node capture attack. The node revocation and key update protocol resists node collusion and impersonation attacks while ensuring the forward and backward secrecy for secure node revocation. We experimented on ATmega328 processor using Arduino Duemilanove controller board and ArduinoISP programmer and used Castalia simulator to simulate the performance of the protocols in the real-time networks. The results show that the pair-wise key establishment and key update protocol has constant low computation overhead for session key update irrespective of the degree of the polynomials used to establish master secret. The node energy consumption for session key update is much less as compared to the one time key establishment. The proposed bilinear pairing based healing protocol reduces the cost overhead, especially the computation and storage overhead, providing additional security as compared to the existing bilinear pairing based healing protocol. With the CRT based healing protocol, we are

able to achieve the same security as the proposed bilinear pairing based healing protocol at significantly low cost overhead in terms of communication, computation and storage. The node capture detection using program integrity verification could be carried out with reduced energy consumption at both node and server end. The capture detection protocol has low communication, computation and storage overhead as compared to the existing software based program integrity verification and also reduces the overall network setup cost when compared with the hardware attestation protocols. With our node revocation and key update protocol, a resource constrained sensor node could get the key update with notably low overhead as compared to the existing revocation protocols.

List of Symbols and Acronyms

$h()$	Unkeyed cryptographic hash function
$h_k()$	Keyed cryptographic hash function with key k
$PRF_k()$	Pseudo Random Function with key k
$E_k(data)$	Encryption of $data$ with key k
$D_k(data)$	Decryption of $data$ with key k
$W \wedge V$	Logical AND operation on W and V
$W \vee V$	Logical OR operation on W and V
Attacker(S)	Attacker has access to S (S may be some term/name/variable/channel)
AttackerC(T)	Attacker can compute term T
WSN	Wireless Sensor Network
BS	Base Station
CH	Cluster Head
SKC	Symmetric Key Cryptography
PKC	Public Key Cryptography
ECC	Elliptic Curve Cryptography
MAC	Message Authentication Code
CRT	Chinese Remainder Theorem
TPM	Trusted Platform Module
PIV	Program Integrity Verification
TVS	TPM enabled Verification Server
TPIV	TPM enabled Program Integrity Verification
NRKU	Node Revocation and Key Update

List of Tables

2.1 Resiliency at various levels in a WSN 26

3.1 Security Features of Polynomial based Key Management Schemes . 69
3.2 Performance Comparison 70
3.3 Simulation Parameters 73

4.1 Comparison of Security Features 106
4.2 Notations used in Performance Comparison 106
4.3 Performance Comparison 107
4.4 Communication Cost in Sec 108

5.1 Comparing Security Features of Node Capture Detection Protocols 134
5.2 Performance Comparison of Node Capture Detection Protocols . . 135
5.3 Improvement in Capture Detection Probability with TPIV 136

6.1 Comparison of Security Features of Revocation Protocols 162
6.2 Performance Comparison of Revocation Protocols 163

List of Figures

1.1 A Generic WSN Setup 2
1.2 Sensor Node Architecture 4
1.3 Various Sensor Nodes 5
1.4 Flat WSN 6
1.5 Hierarchical WSN 6
1.6 Overview of SASNet Operational Architecture 8
1.7 Attacks in WSN 11
1.8 Framework of Proposed Solution to Deal with Node Capture Attack 16
1.9 System Model 17

2.1 Illustrating the Resiliency Strategy 25
2.2 Key Management in WSN 28
2.3 Trusted Platform Module (TPM) Sealing and Unsealing 45

3.1 Two Dimensional Grid based Polynomial Key Distribution 52
3.2 Node Discovery and Authentication 58
3.3 Session Key Update 60
3.4 Resilience to Node Capture with Key Management 68
3.5 Computation Cost of the Key Update Protocol 72
3.6 Node Energy Consumption 73

4.1 Self-Healing 98
4.2 Mutual-Healing 100
4.3 Storage Cost for Mutual-Healing 108
4.4 Computation Cost for Self-Healing 109
4.5 Computation Cost for Mutual-Healing 109

5.1 DAPP Protocol 124
5.2 Authentication and Code Verification Protocol 127
5.3 Attacker Time Line 130
5.4 TVS Optimization 137
5.5 Comparison of Computation Cost for TPIV and DAPP 137
5.6 Comparison of Energy Consumption and Communication Latency for TPIV and DAPP 139

6.1 Comparison of Energy Consumption for NRKU and RP Protocol . 163
6.2 Comparison of Computation Cost for NRKU and RP Protocol . . . 164

CHAPTER 1

Introduction

Wireless Sensor Network (WSN) has emerged as a promising technology for various real-life applications such health care management, battle-field surveillance and so on. A WSN comprises of a large number of low-cost, tiny sensor nodes that work in collaboration to accomplish application specific task. The sensor nodes communicate on a wireless medium that renders it vulnerable to various security threats such as eavesdropping and replay attacks. Sensor nodes have limited computation, communication and computation resources and are usually battery operated. Therefore, the traditional network security solutions do not fit for WSN. With the increasing use of WSNs in real world scenarios, different application requirements influence the underlying security mechanisms.

Some mission critical applications such as battle-field surveillance and forest fire detection leave the nodes unattended after deployment that gives way to physical capture of nodes resulting in more severe attacks such as node capture attack. Although, research on WSN security has advanced along with the growing use of WSNs in real-life, a major research challenge that still exists is dealing with the node capture attack.

The communication can be secured with the robust key management techniques, however, when an adversary physically captures a node, it can gain access to the cryptographic keys stored within the node. A more powerful adversary can carry out node capture attack, wherein the node is reprogrammed and redeployed in the network as an insider attacker to further disrupt the network activities. Therefore, timely detection of node capture attack in a secure and efficient manner is need of the hour. In this thesis, we present a comprehensive secure framework to

deal with node capture attack with due consideration to resource constraints of sensor nodes.

1.1 Overview of Wireless Sensor Networks (WSNs)

Wireless Sensor Networks (WSN) comprise of hundreds or thousands of tiny sensor nodes that monitor events or the environmental conditions such as vibration, temperature, and motion or track some object of interest in the physical world. The sensor nodes in a specified area work in collaboration to collect and report the observations to a central authority directly or indirectly. In a WSN, this central trusted authority, usually known as base station, serves as the data sink/processor that connects the sensor nodes to the external world. Through the base station, the sensor network data is communicated to the digital world of computer systems to make informed decisions in order to accomplish a common application specific task [140] e.g. in battle-field surveillance, forest fire detection etc. (Figure 1.1).

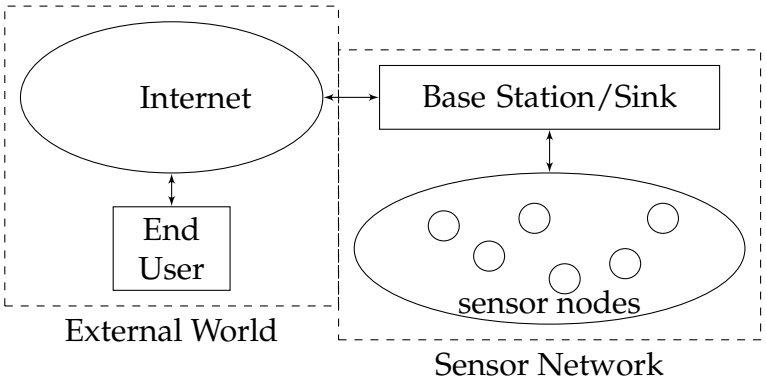


Figure 1.1: A Generic WSN Setup

WSN needs different treatment as opposed to conventional networks because of its distinctive characteristics such as autonomy, the ability of self configuration and deployment location. In a WSN, the sensor nodes are not directly controlled by any human user, the nodes either interact with other sensor nodes or with the base station. The sensor nodes are capable of setting up their services and functions in situations where no central control is available. This autonomy requires

sensor nodes to be self-configurable and be capable of maintaining themselves during the entire lifetime of the network. A WSN typically functions for periods ranging from several days to one or two years. The sensor nodes in a WSN have embedded intelligence that allows them to perform different tasks, however, the WSN protocols and services are dependent on the application requirement.

In a WSN, it is not mandatory to pre-determine or pre-engineer the position of sensor nodes thus allowing the nodes to be deployed in an ad-hoc manner. For example, WSNs can be randomly placed or air-dropped in some disaster relief operations such as forest fire detection and earth quake alerts, or in hostile, inaccessible terrains for applications like battlefield surveillance. Thus, the infrastructure-less setup of WSN with self-configurable sensor nodes is very cost effective. The sensor nodes interact with each other via radio frequency based short range wireless communication that is susceptible to various attacks. In WSN applications, where the nodes are left unattended after deployment, the security of nodes against physical attacks becomes a major concern. The sensor nodes have limited computation, communication and storage capability. Generally, in most of the WSN applications, the power source for a sensor node is an irreplaceable battery. In traditional networks, the main objective is to achieve high quality of service (QoS), however, for WSN, power conservation remains one of the primary concerns. Thus, the security protocols designed for WSNs must also focus on providing the required security with minimized overhead on resource constrained sensor nodes.

We briefly discuss the sensor node architecture, topologies, standards, applications of WSN and security issues in WSN subsequently. Then, we present the motivation behind taking up this work and discuss the contribution of the thesis.

1.1.1 Sensor Node Architecture

Sensor nodes are autonomous units with sensing facility along with limited processing, storage and communication capability and, are usually battery operated. Figure 1.2 gives a general architecture of a sensor node.

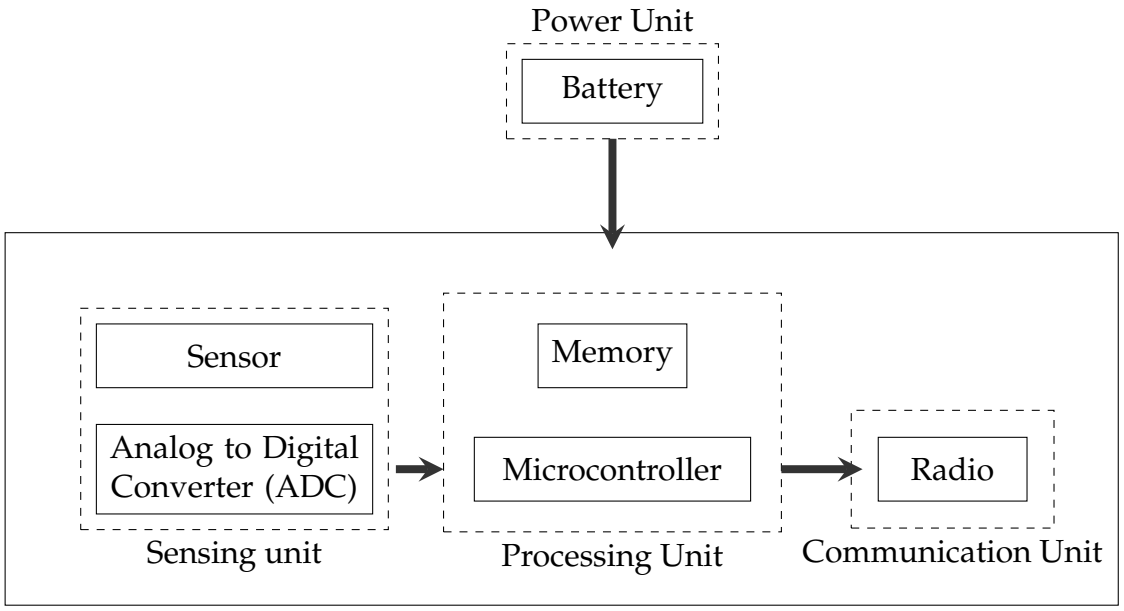


Figure 1.2: Sensor Node Architecture

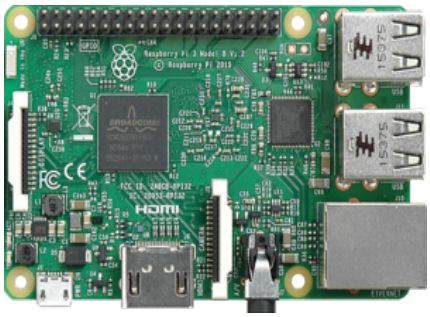
With the help of sensor hardware, a bridge is built between the abstract world and the physical world. A sensor node comprises of an on-board processor with memory, a sensing unit, a radio unit and a power unit. The sensors in the sensing unit produce a continuous signal related to measured physical world data such as temperature and vibrations. The analog signals from sensors are converted into digital signals by analog to digital converter (ADC) and fed into the processing unit wherein simple computations can be carried out before sending the data to the next level nodes or to the base station through radio unit. A sensor node consumes maximum energy for communication. It is observed that the cost of executing 3 million instructions by a 100 MIPS per word processor is approximately same as that of the energy cost required to transmit 1 KB of data to a distance of 100 meters [103]. Thus, to save on energy cost, in most of the cases, rather than sending the raw data, the nodes transmit partially processed data needed by the higher level nodes.

In last two decades, numerous sensor nodes of varying capabilities have been made available in the market by different vendors, for example IRIS [54], Waspmote [128] and Raspberry [106] (Refer Figure 1.1.1). IRIS has MEMSIC Atmel ATmega1281 processor with 16 Mhz clock, 128 KB Flash and 8KB RAM as stor-

age. IRIS can communicate at the data rate of 250 Kbps. With Libelium Atmel ATmega1281 processor having 14.7456 MHz clock, Waspmote works at 250 Kbps data rate and has 128 KB Flash memory. Raspberry Pi is the latest trend with 1.2 GHz 64/32-bit quad-core ARM Cortex-A53. Raspberry Pi has 1 GB RAM and uses Wi-Fi 802.11n for communication. The choice of a sensor node largely depends upon the application requirements and the budget.

(a)
IRIS

(b)
Wasp



(c) Raspberry Pi

Figure 1.3: Various Sensor Nodes

1.1.2 WSN Architecture

The sensor nodes in a WSN can be arranged in different topologies depending on the underlying application needs. For applications such as monitoring the temperature conditions in a small area, a WSN may be a flat network with base station (BS) and all nodes at the same level. In a flat WSN, all the sensor nodes communicate the sensed data directly to the base station. (Refer Figure 1.4).

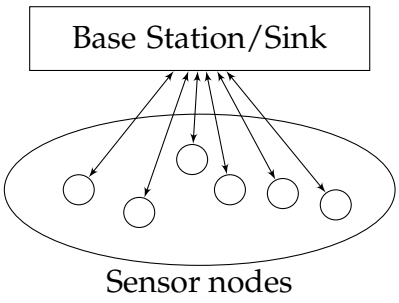


Figure 1.4: Flat WSN

However, for large applications, such as surveillance of battlefield, where the network is spread over a large geographical area covering a few kilometres, it may be a hierarchical network as shown in Figure 1.5. In a hierarchical WSN, the nodes communicate with the base station through some more powerful intermediate nodes known as cluster heads.

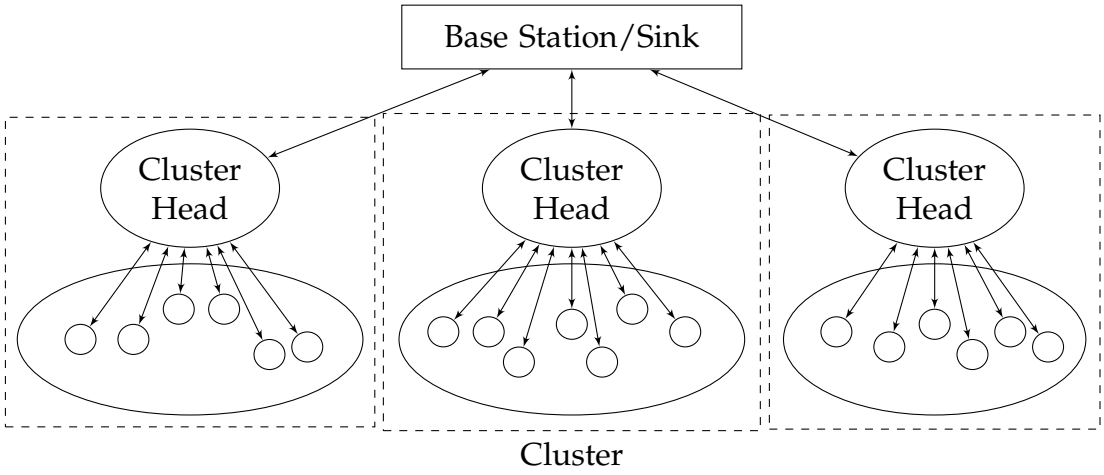


Figure 1.5: Hierarchical WSN

1.1.3 WSN Standards

The underlying standards used in wireless sensor network for communication are mainly IEEE 802.15.4 standard and ZigBee [140]. The physical and medium access control layers are defined in IEEE 802.15.4, while the network and application layers are defined in ZigBee. These two protocol stacks compromising of IEEE 802.15.4 and ZigBee together can support long lasting applications with low

data rate on battery powered wireless devices such as sensor nodes. The physical layer of the IEEE 802.15.4 standard provides packet transmission on the physical medium and activation/deactivation of the radio transceiver and, operates on three different license free radio frequency bands:

1. 868 - 868.6 MHz with a data rate of 20 kbps (in Europe)
2. 902 - 928 MHz with a data rate of 40 kbps ((in North America)
3. 2400 - 2483.5 MHz with a data rate of 250 kbps (worldwide).

Data and management services are provided to the upper layers by the medium access control layer of IEEE 802.15.4. The medium access control packet transmission and reception across the physical layer is enabled by data service. The management service comprises of synchronization of communications, association and disassociation of devices to the network and, management of guaranteed time slots. The medium access control layer also implements basic security mechanisms leaving the advanced security features, such as key management and device authentication, to the upper layers. The medium access control layer security services are based on symmetric keys that are securely generated, transmitted, and stored by the upper layers. Access control, frame integrity, data encryption, and sequential freshness are some medium access control layer security services and are optional. ZigBee is built upon the IEEE 802.15.4 standard wherein the network layer provides support to various network topologies such as tree, star, and peer-to-peer while the framework for communication and distributed application development is provided by the application layer.

1.1.4 WSN Applications

WSNs have found enormous applications in almost every walk of the life [6]. Some of the commonly used applications of WSN are discussed below:

- *Environmental Applications:* Sensor networks are deployed for habitat monitoring, wherein the conditions of plants and wild animals in the wild life can be monitored by sensor nodes, for example, Great Duck Island project [82], ZebraNet [56] and Luster [110]. Water quality monitoring in the field of hydrochemistry, air quality monitoring for air-pollution control, monitoring

of chemical or biological hazards in chemical plants also use WSNs. Sensor nodes may also be scattered to detect forest fire or seismic activities in disaster prone areas as an early warning mechanism.

- *Military Applications:* The presence of enemy troops and vehicles can be tracked by deploying the WSN for battlefield surveillance. For example, the SASNet [70] as an agile surveillance system with an aim to provide improved operational flexibility and usability in military surveillance (Refer Figure 1.6).

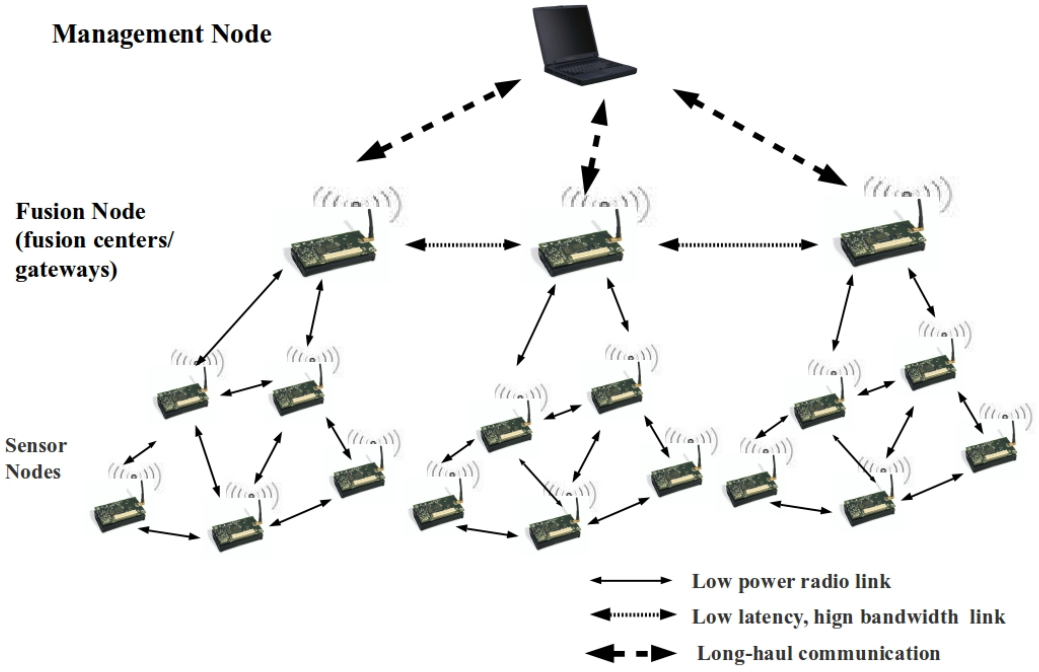


Figure 1.6: Overview of SASNet Operational Architecture

Sensitive objects and buildings such as communication centers and headquarters can be protected with the help of WSNs. Sensor nodes can also be utilized for intelligent guidance and coordination by mounting them on unmanned robotic vehicles, submarines etc. The deployment of WSNs can be done for detecting the possibility of terrorist attacks and remote sensing of

chemical, nuclear or biological weapons as well. Battle damage assessment can also be carried out using WSN.

- *Health Care Applications:* Health care industry has been looking for the means to provide quality health care at reduced costs and also the focus is being shifted to prevention and early detection of ailments. WSNs are capable of providing continuous monitoring even remotely and can be deployed with low set up overhead. Through remote monitoring with the help of sensor nodes deployed at the patient's home, the doctors can keep track of the patient's behavior and move and provide immediate medical attention when required. Vital signs within an elderly person or patient can be monitored by equipping the person with wearable sensors integrated into wireless body area networks. This provides real time health updates of a patient. A practical implementation of one such system is AlarmNet (Assisted Living and Residential Monitoring Network) [130] that offers ubiquitous and flexible health care to elderly personnel and patients as per their needs.

Apart from these applications, WSNs can also be used in road transport applications to track vehicles and traffic control, in structural monitoring applications to monitor bridge strength and building life, in warehouse management, in industrial process control and so on. Thus, WSNs can give us a promising solution for many real-life problems.

1.2 Security Issues in WSN

Since last two decades, WSNs have been implemented in diverse areas such as battlefield surveillance, forest fire and earth quake detection, health care, home automation applications and so on [113]. A WSN needs to be protected against malicious activities that can adversely affect the network functionality. As the sensor nodes in a WSN are resource-constrained in terms of computational capabilities, battery power, memory, and communication bandwidth, to implement the traditional security protocols and cryptographic algorithms is challenging. Most of the applications using WSNs are remotely handled and therefore, the security

of such networks is the major concern while designing the network.

In a WSN, an adversary can eavesdrop on the public wireless communication channel and gather the information about the activities happening in the network. Although, because of the inherent passive nature of the eavesdropping attack, the network behavior is not directly affected, the information acquired from passive attacks can be used to perform active attacks. An adversary can generate fake events, modify the messages, and can even introduce bogus control information as an active attacker. In WSNs, routing is an essential service for the data and information to reach the sink. As the communication happens on unreliable wireless channel, the traffic may be disrupted by performing various routing attacks such as replay attack, selective forwarding, sink hole and worm hole attacks. The attacker may use jamming and hello flood leading to denial of service [81].

The unattended deployment of WSN in various applications further pose more security threats including node subversion and node capture. Thus, WSN demands security solutions based on application security goals that vary from application to application. For example, for a WSN deployed inside a hospital to track the doctors and patients, we do need to ensure the integrity and confidentiality of the information about patient's health, but the individual sensor nodes deployed within the hospital building are physically protected. However, a WSN deployed in battlefield surveillance to detect and track targets and to send the real time information of enemy mobility to the command center, there is a threat of node capture attack wherein, for further malicious activities, the enemy may physically capture the unattended nodes, reprogram and redeploy those nodes in the network.

Figure 1.7 summarizes the commonly possible threats to WSN security. The effects of one or more attacks can render the services of a WSN useless and therefore can not be ignored. In mission critical applications such as health-care monitoring and forest fire detection, there may be life-threatening results due to malfunctioning of the network.

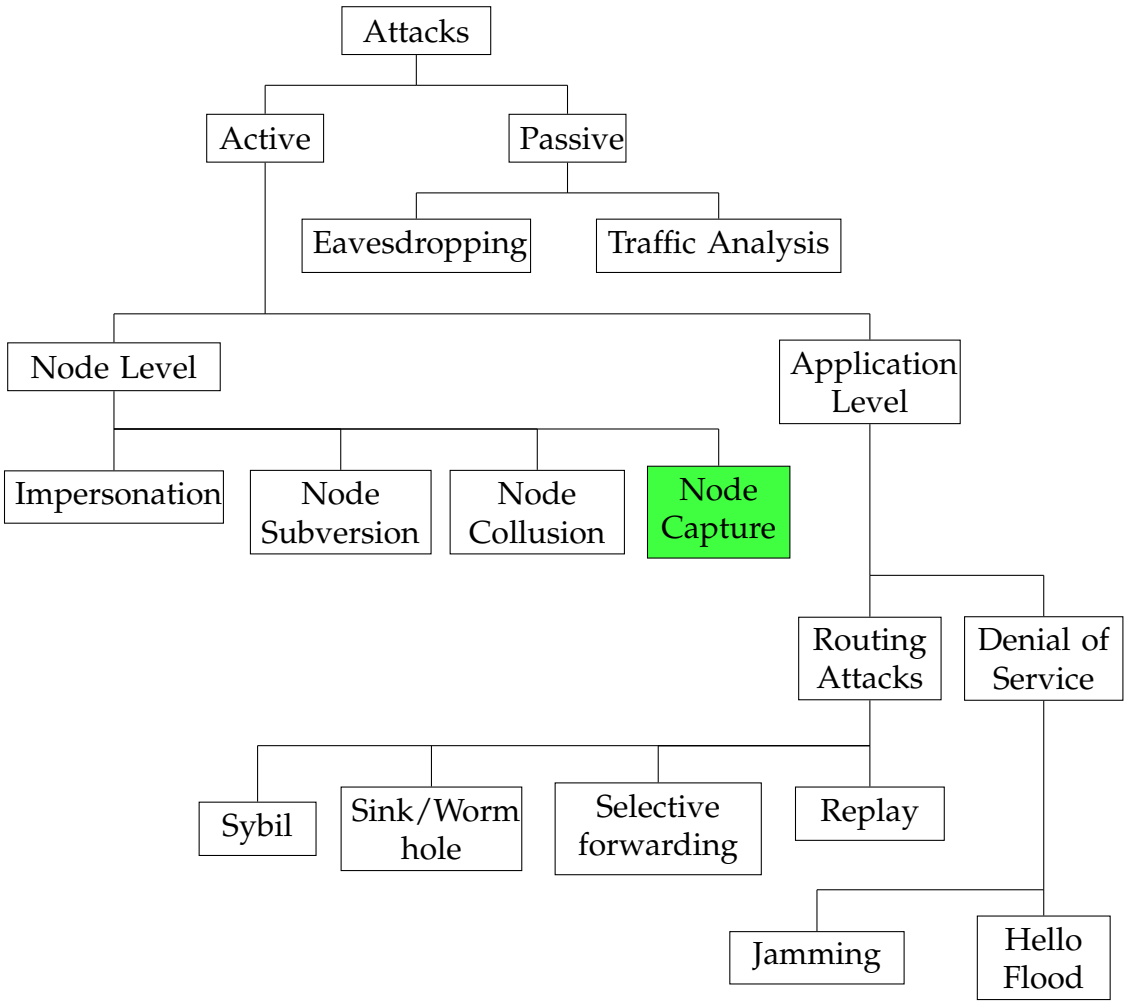


Figure 1.7: Attacks in WSN

1.3 Motivation

Many real life applications demand the deployment of WSNs in hostile environments. For example, military surveillance missions are carried out with the aim of providing alerts to the military commands about the objects of interest such as enemy troops and vehicle movement in hostile territories. In such scenarios, surreptitious action is needed as threat to human life is always present. Thus, deploying WSNs as unmanned vigilance agent tremendously helps the military in the surveillance of the region. VigilNet [51] is one such WSN application for military that is used to obtain and verify the capabilities of the enemy and the hostile target positions in a battle-field.

Over the years, various security solutions have been proposed to address specific security issues of WSN, since the inherent resource constraints of sensor nodes restrict the use of conventional network security mechanisms [6]. Despite securing the communication with the help of secret keys [81] [144], the unattended deployment of sensor nodes in harsh environments for such mission critical applications pose a major threat of node capture attack [39].

The node capture attack has different level of severity depending upon the adversary capability and the time available with an adversary to carry out an attack [10]. An adversary aims for node capture attack because it provides an effective way to be present in the network and control the network activities as an insider. For example, in a large sensor network deployed for battle-field surveillance, the sensor nodes are randomly air-dropped in the field and are left unattended. In such scenario, the enemy may get hold of one or more nodes physically and can easily extract cryptographic keys for using them to communicate on the associated links. In such hostile environments, even an attacker can not remain present all the time, therefore, the attacker may change the program running on a sensor node and redeploy the node in the network as an insider attacker. From a set of captured nodes, the adversary can recover enough cryptographic information and further compromise the secure communication links, subsequently causing substantial damage to the entire WSN. This mandates the need for detecting the node capture attack victim as early as possible.

Typically, it is assumed that in physical attacks, an attacker has unsupervised access to a node for an extended period of time. However, in normal WSN operation, nodes keep communicating with their neighboring nodes and if a node is continuously absent then it is an unusual condition that neighbors can notice. Some of the existing protocols to detect a victim of the node capture attack, based on monitoring of the nodes through base station, cluster head, group manager or peer nodes, are proposed in [39] [87] [76]. Even with the continuous monitoring, it is possible that the malicious neighbors collude and the detection is bypassed, resulting in node capture.

In order to prevent an attacker from deploying a reprogrammed node for carrying

out insider attacks, various attestation protocols have been proposed such as [112] which require strict time measurement and in multi-hop wireless networks, it is impractical to achieve the same. The program integrity verification protocol in [99] is vulnerable to impersonation attack and needs the continuous involvement of base station. The program integrity verification with distributed authentication in [25] exposes all node program codes to adversary on verifier compromise and does not address memory addition attack. The hardware based protocols as proposed in [117], require specialized hardware such as Trusted Platform Module (TPM) [122] at all the sensor nodes resulting in high network setup cost in large sensor networks. Therefore, a secure and efficient protocol for timely detection of node capture attack is required.

When a node is detected to be a victim of node capture attack, it must be revoked from the network in order to prevent further damage to the network. A practical and feasible approach to revoke a victim node is to exclude it from the further network activities. For this purpose, the secret shared within the group of non-captured nodes can be updated. Since, sensor nodes are resource constrained, the node revocation process should place significantly low overhead on the nodes.

From the above discussion, it can be observed that the node capture attack is a major security threat especially to the WSNs deployed in harsh unattended environments. Although various proposals for WSN security are considered in the literature, a complete solution to deal with node capture attack in secure and efficient manner taking into account key establishment, detection of node capture attack and revocation of the captured node is not considered.

Our motivation to take up this work is to provide a complete solution to securely and efficiently deal with node capture attack.

1.4 Contribution of the Thesis

The proposed solution integrates the key establishment, detection of node capture attack and revocation of victim of node capture. We address the problem of node capture detection in clustered wireless sensor networks, as most of the sen-

sor node applications deploy large sensor network with hierarchical (clustered) topology. Moreover, the clustered network does not have the threat of single point of failure and is suitable for large sensor network deployed in a vast geographical area for applications such as battlefield surveillance and forest fire detection.
The contribution of the thesis is summarized as below:

- **Secure key establishment.** The wireless sensor network demands confidentiality, authenticity and/or integrity of the messages depending on the application needs. For example, an environment monitoring application keeping track of the surrounding temperature and pressure maintains the authenticity and integrity of the information, but does not need confidentiality. However, the health care application keeping the patients' health records maintains confidentiality as well to preserve the privacy of the patient. For secure communication, it is required for the nodes to share a secret pair-wise key and/or group key that can be dynamically updated after each communication session. We have considered a clustered wireless sensor network wherein the communication is secured at two different levels:
  - At base station-cluster head level, we have proposed the pair-wise key establishment amongst base station and cluster heads using multiple polynomial share based key. Multiple polynomial share based key can be established between any pair of nodes irrespective of their physical locations and provides strong resilience to node capture attack as well.
  - Within a cluster, the nodes communicate amongst themselves using a group key distributed through a session wise broadcast by cluster head. The proposed group key distribution is enabled with self-healing and mutual healing property.
- **Detection of node capture attack.** A node that is suspected to be a victim of node capture attack undergoes the program integrity verification. As soon as a cluster head notices the unusual absence of any node from the network, the suspect node is challenged to prove the integrity of its program code. If node is captured, reprogrammed and redeployed in the network, the pro-

posed program integrity verification protocol detects the node capture victim with very high probability.

- **Secure revocation of victim node.** When a capture attack is confirmed on a node, it must be revoked from the network to prevent further damage caused by the victim node as an insider attacker. The proposed node revocation and key update protocol allows a cluster head to exclude the victim node from sharing the further group session keys within its cluster.

1.4.1 Framework of Solution to Deal with Node Capture Attack

In Figure 1.8, the framework of the proposed solution is presented. Once the network is deployed, the nodes form a cluster by responding to the request beacon of the nearest cluster head. A pair-wise key is established within the pair of CHs and also between a CH and the base station (BS). For each session, within a pair of BS-CH or CH-CH, the pair-wise key is dynamically updated [4].
For communication at the level of cluster, a cluster head distributes the session key to all the nodes authorized to participate in the session within its cluster using self-healing and mutual-healing enabled group-key distribution protocol [3].
Each cluster head monitors its cluster nodes by keeping track of the transmissions from its cluster nodes. If no transmission is heard from a node for more than a set threshold time, the monitoring cluster head suspects the node to be captured and the suspect node is requested to prove the integrity of its program code. Program integrity verification protocol [5] is executed between the cluster head and the suspect node.
If the suspect node fails to prove its program integrity or it has moved out of that cluster without informing its cluster head, it is considered to be captured and the cluster head then either proceeds for node revocation and key renewal process or it decides to repair the node by reprogramming it with the valid program code.

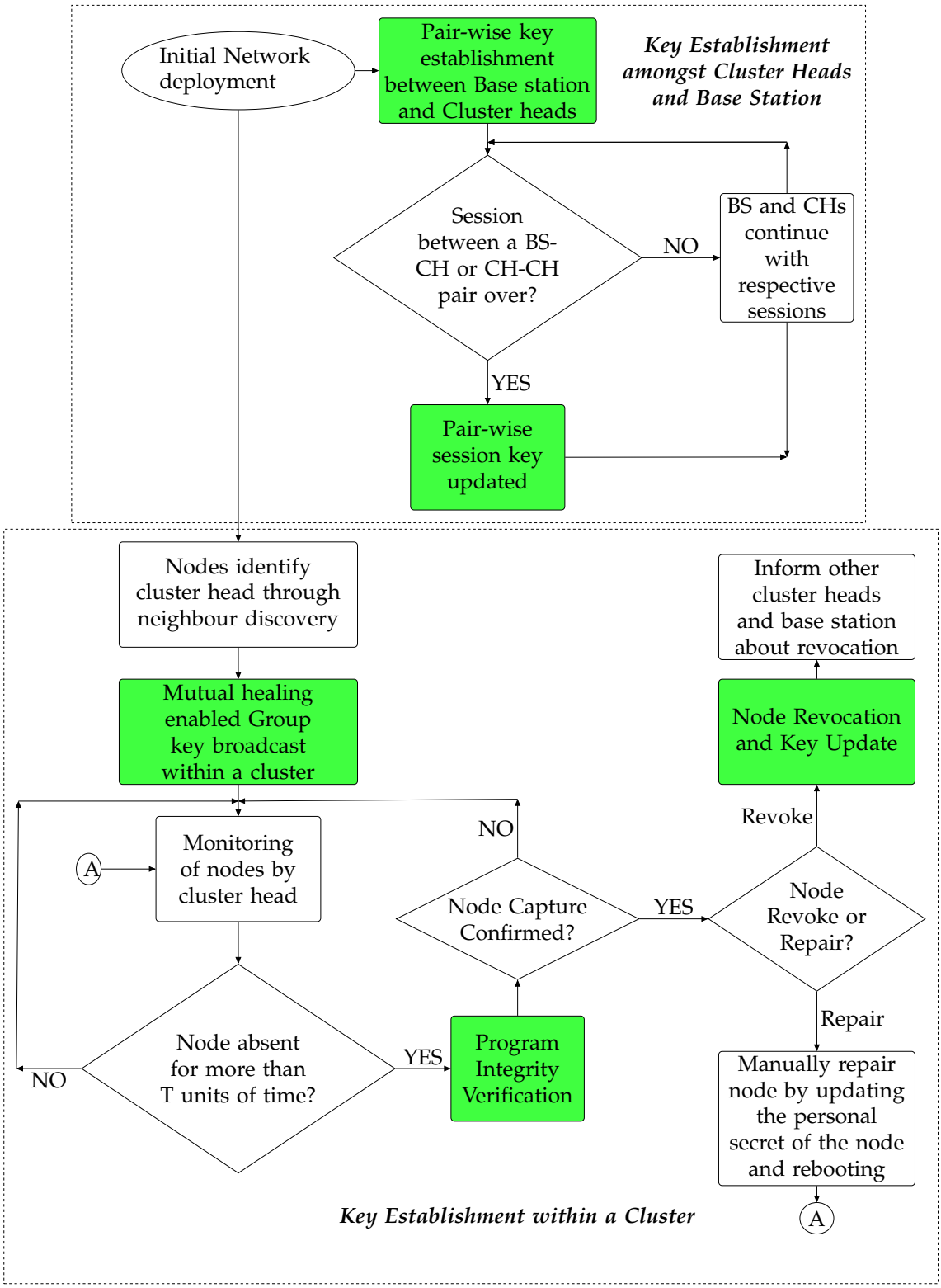


Figure 1.8: Framework of Proposed Solution to Deal with Node Capture Attack

1.4.2 System and Network Model used in the Proposed Solution

The system and network model used in the proposed solution is as depicted in Figure 1.9.

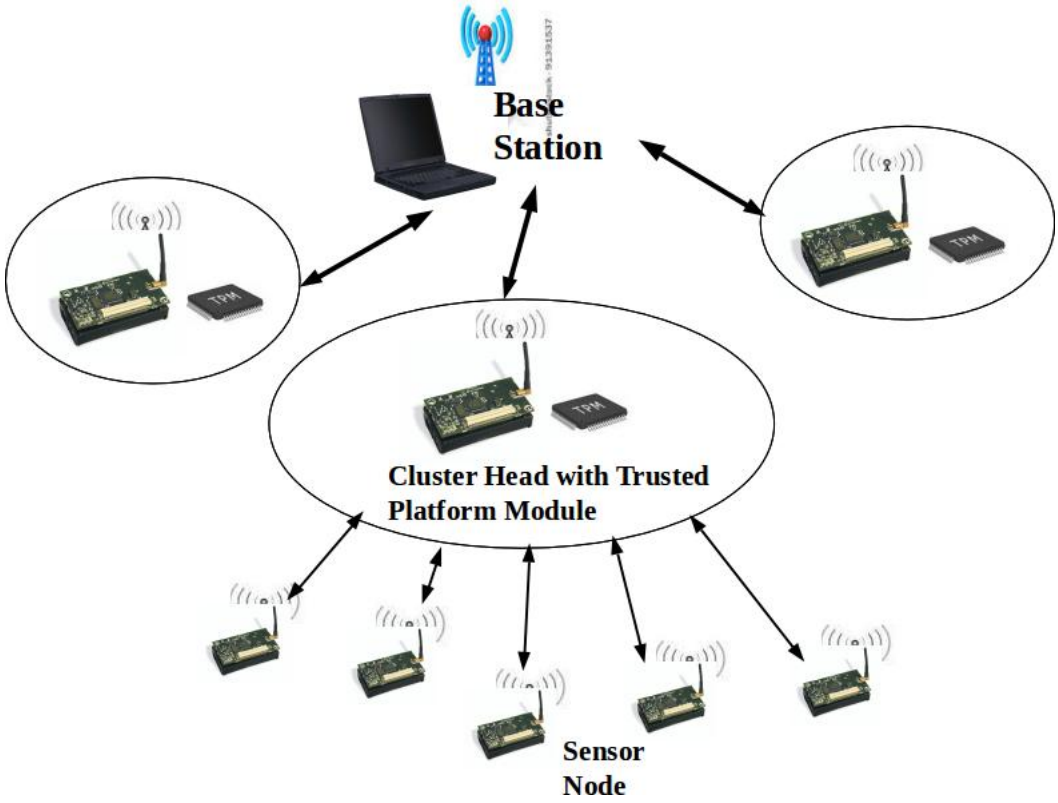


Figure 1.9: System Model

Base station is the trusted link between the WSN and the external world and serves as the central authority in the WSN. In the clustered WSN, some resourceful cluster heads (CH) link the base station with nodes. Prior to deployment, base station assigns unique identities to the cluster heads and the nodes. The program memory in each node has sufficient free space, after loading boot code, application code and flash down-loader, to store a unique random incompressible bit string that serves as a unique secret of the node. Each cluster head is equipped with a trusted platform module (TPM) [122] in which the program memory content of all the nodes are sealed. Within a cluster, the nodes communicate using a group session key. When a cluster head changes its location, reformation of the clusters take place. A node moving away from its cluster informs its cluster head, who in turn informs the move to other nodes in its cluster as well as to the head of new

cluster which the moving node decides to join.

Adversary Assumptions. We consider an active adversary which can intercept, remove or update packets or insert new packets into the network apart from just eavesdropping on the communication channel to listen to the on-going communication. The adversary can also physically capture, reprogram and redeploy a node in the network to launch further insider attacks. An attempt from an adversary to put additional memory in the captured node or to inject malicious code into the node will be detected by the proposed program integrity verification.
The security claims made in the proposed protocols are analyzed with the help of theorem proving techniques and by carrying out the formal analysis using ProVerif tool [14].

1.5 Thesis Outline

In chapter 2, we give the preliminaries for the protocols that are building blocks of the proposed solution to deal with node capture attack in WSNs. First, the overview of security in WSN is discussed with an emphasis on secure key management, node capture detection and node revocation. The primitives used in the various protocols discussed in the thesis namely pseudo random function, bivariate polynomials, bilinear pairing and Chinese remainder theorem based secret sharing are defined next. At the end, we give an overview of trusted platform module (TPM) that are associated with cluster heads in our capture detection protocol.

In Chapter 3, we present the authenticated pairwise key establishment and session key update protocol. We give an overview of the pair-wise key establishment protocols proposed for WSNs such as SNEP. Next, polynomial share based key establishment schemes that do not require the presence of third party are described. Then, we discuss in detail the protocol for pair-wise key establishment used for communication amongst base station and cluster head in our solution

framework wherein the initial pair-wise master secret is established using shares from multiple bi-variate polynomials. The protocol allows the subsequent session key update using master secret and random inputs from the participating pair of nodes.

In Chapter 4, the self healing and mutual healing enabled group key distribution is presented. We first give the need of self-healing and the brief overview of the existing self-healing protocols. We then discuss the importance of mutual-healing and describe the bilinear pairing based mutual healing protocol available in the existing literature. Then, we present the improvement over existing bilinear pairing based protocol and discuss the significantly improved self-healing and mutual-healing using Chinese remainder theorem based group key distribution that we have used for communication within clusters in our solution framework.

In Chapter 5, we present the vital contribution to this thesis in terms of a secure and efficient solution to detect node capture attack. We start the discussion with identifying node capture attack by monitoring and then present the brief overview of the existing software and hardware attestation approaches to detect node capture attack. Next, we discuss the program integrity verification approach and present the proposed solution of node capture detection that uses trusted platform module enabled cluster heads to verify the integrity of a node's program. With the help of light weight hash operations and pseudo random functions, the protocol is able to detect the victim of node capture with high probability even in presence of a strong adversary capable of putting additional memory in the node.

In Chapter 6, we identify the requirement of node revocation after node capture detection and discuss the various centralized, distributed and hybrid protocols proposed in the literature to address the issue of revocation. Then, we present a secure node revocation and key update protocol that puts very low overhead on the resource constrained sensor nodes. The proposed protocol revokes a victim node by not allowing the victim node to obtain the group session keys thereafter.

In Chapter 7, the conclusion for the thesis is given, wherein we briefly present the importance and merits of the proposed protocols for pair-wise key establishment, healing enabled group key distribution, node capture detection using program integrity verification and node revocation and key update used in the solution framework for secure and efficient dealing with node capture attack in wireless sensor networks. At the end of this chapter, we give the future scope of this research.

CHAPTER 2

Background and Preliminaries

WSNs are gaining popularity in various aspects of real life applications due to their ease of deployment even in difficult terrains such as battlefields and earth quake prone areas. Along with the growing use of WSNs, security of WSNs has been a matter of concern. The communication on unreliable wireless medium and deployment of WSNs in harsh terrains depending on application specific needs leave WSN vulnerable to various security threats including routing attacks and physical attacks. We studied various aspects of WSN security and realized that node capture attack is one of the most critical attacks on WSN security. In this chapter, we give an overview of WSN security with the emphasis on secure key establishment and, node capture detection and node revocation. We also present the basics of security primitives and the overview of trusted platform module used in the proposed protocols.

2.1 Overview of Security in WSN

WSNs give a promising approach for various applications mostly in hostile and emergency environments e.g. battlefield surveillance, forest fire detection, health care and so on. Most of such applications using WSNs are mission critical. The wireless communication in sensor networks is exposed to attacks such as eavesdropping, traffic monitoring and analysis. Furthermore, due to their deployment mostly in hostile environments, the sensor nodes are vulnerable to tampering or even physical destruction. Therefore, the security is a major concern while designing such networks. To integrate the required security features, the secu-

rity mechanisms require one or more cryptographic primitives namely symmetric key cryptography (SKC), asymmetric key cryptography and hash functions. With SKC, both communicating parties must share the same security credentials which means the same key is used for encryption and decryption. Asymmetric key cryptography, also known as public key cryptography (PKC) uses two distinct keys, a private key that is kept secret, and a public key that is known publicly. In PKC, the encryption done with the public key can be decrypted with the corresponding private key. Cryptographic hash functions [88] provide "digital fingerprints" of the given data. Other cryptographic primitives such as message authentication code (MAC) can be built using cryptographic hash functions. The cryptographic primitives have been extensively used in the network security. However, due to the inherent constraints of memory, computation power, energy and requirement of keeping the cost low, the existing network security solutions can not fit in WSN as it is. Therefore, the efforts have been made to secure the WSNs in an efficient and economical way. In traditional networks, the traffic pattern is dominantly end-to-end communication wherein the intermediate routers do not require to access the message bodies. Therefore, the authenticity, integrity and confidentiality of messages are usually ensured using end-to-end security mechanism such as SSL (Secure Sockets Layer) [118]. However, in WSNs, in-network processing such as data aggregation may be carried out by the intermediate nodes (cluster heads). In such cases, if message integrity is left to be checked only at the final destination, an adversary may inject packets en-route. To address this issue, a link-layer security architecture, TinySec, was proposed in [58] that can be integrated into existing applications. Two variants of security options are proposed in TinySec. In authentication only mode, the data load remains unencrypted and the entire packet is authenticated using a MAC. MAC is a cryptographically secure checksum of a message that is used to achieve message authenticity and integrity in the communication using shared symmetric key. A sender party computes MAC over the data being transmitted using the shared secret key and transmits this MAC along with the data. At the receiver's end, the MAC is computed on the received data using the same shared key and the computed MAC is compared

with the received MAC value. If both values match, the received data is accepted as authenticated and correct. MACs are keyed hash functions that are hard to forge without the secret key. With authenticated encryption (TinySec-AE), data payload is encrypted and a MAC, computed over the packet header and the encrypted data, is used to authenticate the packet.

In general, the threat to security of a WSN is influenced by three major factors [63] - the topology of the network, the density of the network and, the underlying key management scheme governing the manner in which keys are shared among the nodes. Moreover, the security requirements may vary depending on the application specific needs. The health care applications are amongst the real life examples of WSN application, wherein the medical sensors are put on a patient's body or the environmental sensors are installed at different places within the hospital premises. These sensor nodes may be captured by an attacker, who can alter the programs in sensor nodes according to his needs to carry out malicious activities and can later place such captured nodes back into the network. Such attack may even put a patient's life into danger. Similarly, in a WSN deployed for earthquake alert, a capture attack victim functioning with adversarial motives can disrupt the alert-mechanism that may result in heavy loss to the movable property and humans. By securing the node location with the help of secure localization, the possibility of physical capture of nodes can be reduced. Various secure localization schemes such as Serloc [71] and [94] have been proposed. In [94], secure localization of sensor nodes is considered from two perspectives. First, an adversary may pretend to be an unknown or anchor node and compromise other nodes in order to interfere with the localization process. Therefore, secure node authentication process is needed. Secondly, the integrity of the localization information is required as an attacker may delete, replay or change the location information. For authentication and integrity protection, secure key management is essential. Although, with secure localization and robust key management mechanism in place, we can increase the resilience to node capture attack, the possibility of an adversary launching node capture attack exists. Therefore, a secure and efficient way of dealing with node capture attack remains as one of the most challenging

problems in wireless sensor network security.

In the subsequent sections, we discuss concept of resilience and then the background study carried out to address the issues of secure key management, detection of node capture attack and node revocation.

2.1.1 Resilience

The term *Resilience* has been used by different communities in different contexts. The dictionary defines resilience as toughness or the capacity to recover quickly from difficulties. In general engineering systems, fast recovery from a degraded system state is often termed as resilience. Dependable computing community defined resilience as the persistence of service delivery that can justifiably be trusted, when facing changes. In computer networking, resilience is defined as the ability to provide and maintain an acceptable level of service in the face of faults and challenges to normal operation. Threats and challenges for services can range from simple misconfiguration over large scale natural disasters to targeted attacks. Resilience is seen as the combination of trustworthiness (dependability, security, performability) and tolerance (survivability, disruption tolerance, and traffic tolerance) [123]. Resilient networks aim to provide acceptable service to applications. The required level and the properties of resilience is determined by the needs of users and the application service provided by the network. A network's resilience depends upon the ability of the entities to defend and protect themselves in the face of challenges and in the presence of adversary. A network can be made resilient by implementing various security mechanisms such as authentication, confidentiality and integrity in the communication.

A resiliency strategy is implemented with the help of a four step process [132]:

- *defend* against challenges and threats
- *detect* when an adversarial attack has taken place
- If an attack has happened, *remediate* to minimize the impact of the attack

- when attacker has left the network, *recover* to normal operations

Figure 2.1 below illustrates a Resiliency strategy.

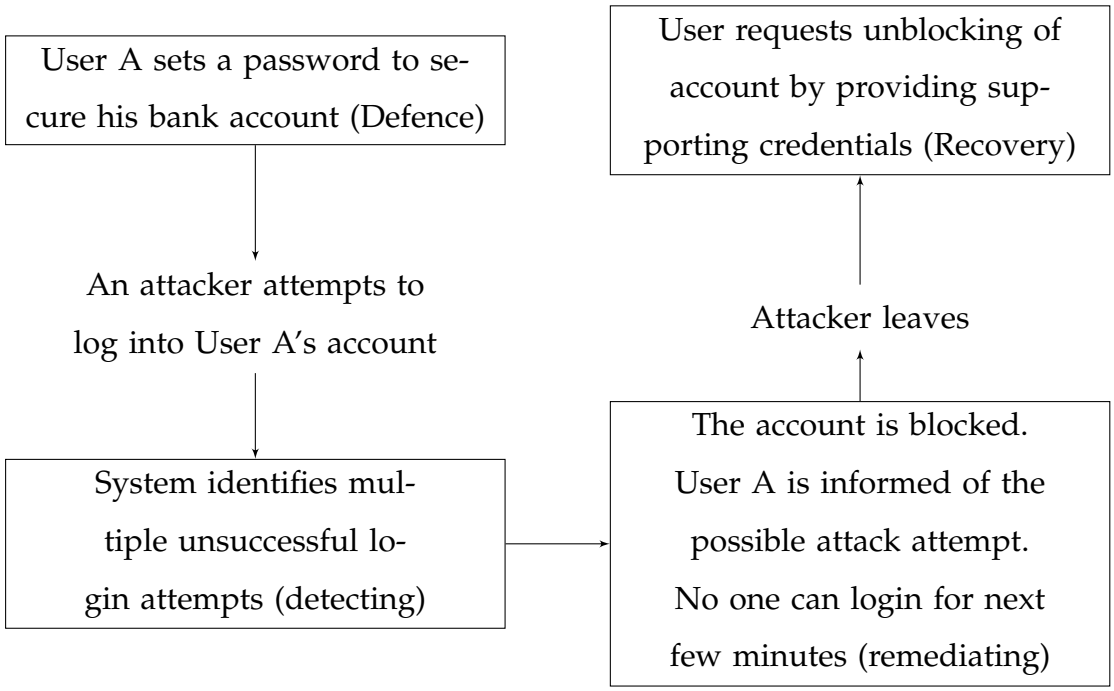


Figure 2.1: Illustrating the Resiliency Strategy

In the context of WSN, the service provided by the network is either in terms of raw data gathered by sensor nodes and/or data aggregated and processed by cluster heads or base station. A WSN is required to be resilient to attacks hampering the service either by disrupting the functioning at sensor node/aggregators or the communication within the sensor network. Due to the insecure communication on a wireless medium and resource constraints of sensor nodes in terms of computational, communication, memory and energy, the resilience to various attacks on the security of a WSN, such as known-key attack, sink hole attack etc. is very crucial. Moreover, WSNs are mostly deployed in unattended hostile environments such as battle-field surveillance, the nodes are exposed to physical attacks as well. Therefore, making a WSN resilient to physical attacks such as node capture attack is an essential requirement as well. A WSN is considered to be resilient if it is capable of providing and maintaining an acceptable level of security service even when some nodes are compromised. Also, it is believed that a distributed network provides better resilience as compared to a centralized network

when the network size is large [29]. A WSN should be capable to survive and successfully deal with internal attacks in case of some portion of compromised valid nodes. [97]. We can broadly classify the resiliency in terms of the security at various levels in a WSN as shown in Table 2.1.

Level	Resilience Against
Application Level	Key compromise, node collusion, message replay, impersonation, denial of Service through heavy inflow
Network Level	Sink hole, worm hole, selective forwarding, denial of Service through jamming
Physical Level	Node capture, Physical Tampering

Table 2.1: Resiliency at various levels in a WSN

At application level, if a key is accessible to an attacker, it can mount attacks such as known-key attack, and attack to forward/backward secrecy that prevents achieving confidentiality and integrity of the communication. Even without the knowledge of key, adversary may replay previous messages and interfere into the ongoing communication. Some malicious nodes may collude and misbehave to disrupt the normal network operation. An attacker may simply cause an excess inflow of messages to few selected nodes in the network to drain their batteries rendering them unavailable.

Routing, a network level function, is an essential function in a WSN. An adversary mounting the attacks such as sink hole, worm hole or selective forwarding aims to hamper the routing functionality and hence affecting average delivery ratio, average degree of nodes in order to detect abnormalities of neighborhood and and/or average path length. The routing resiliency defines keeping these routing metrics in order.

One of the most crucial aspects of WSN security is to ensure that the network is

resilient to physical attacks such as node capture. The resiliency to node capture attacks can be defined from three different perspectives [73].

- the probability that a link is compromised when an adversary captures a node
- number of nodes whose security credentials are compromised when an adversary captures a node, or
- number of sensor nodes required to be captured to compromise whole WSN

In this thesis, we have addressed the resiliency of the WSN at different levels. The proposed pair-wise key establishment protocol provides forward secrecy and resists known key, impersonation and replay attacks. The protocol also achieves node-to-node authentication, data confidentiality and mutual key control and is resilient to sink-hole, worm hole attacks as well as to node capture attack. The proposed healing enabled group-key distribution protocols also ensure resilience against replay and impersonation attack. In order to address the resilience at physical level, our proposed node capture detection protocol ensures that the victim of node capture attack is detected with very high probability. Moreover, capture of a node does not compromise the security credentials of any other node in the network. The node revocation and key update protocol ensures that the victim is revoked from participating in further network operations so the resiliency to further internal attacks is ensured. The revocation process ensure resiliency against node collusion, replay and impersonation attacks.

2.1.2 Key Management in WSN

Due to the wireless communication medium and deployment in difficult terrains, it is important to ensure that, within a WSN, the accurate data is generated and/or forwarded by the authenticated source and reach the intended recipient in time, without illegal alteration of the data in transit. In order to secure the data inside the node and while in transit, numerous approaches for securing communication between nodes in a WSN through a cryptographic key have been proposed and

followed. The encryption security largely depends on the underlying key management scheme. Although conventional network security solutions can not be used for WSN security due to resource constraints and application specific deployment, WSN security does rely on the principles of cryptography wherein the data being transmitted or stored in a system needs to be protected against unauthorized disclosure or modification [98].

To develop a secure application, the fundamental requirement is a key management protocol that is used for setting up and distributing cryptographic keys to the nodes in a network. The basic approaches used in key management in a WSN can be broadly classified in different categories as depicted in Figure 2.2.

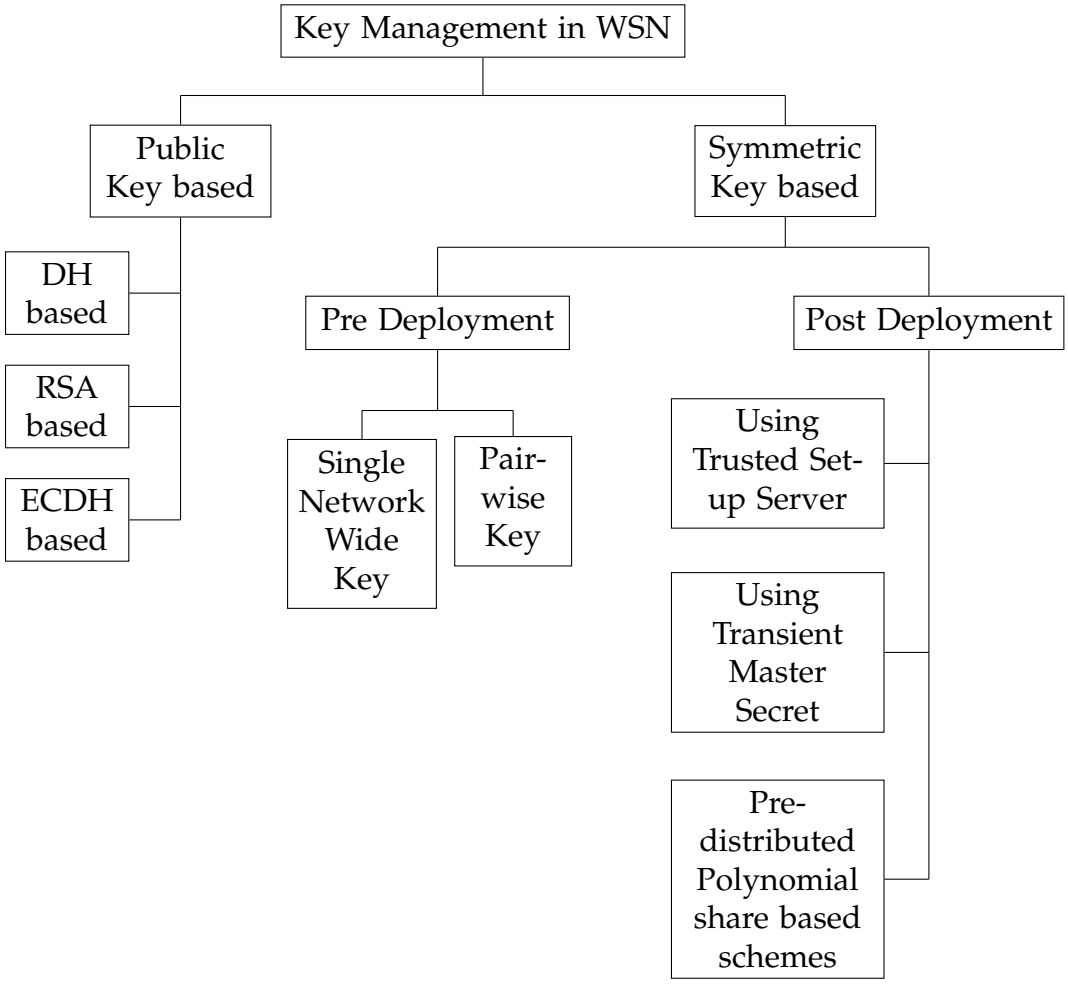


Figure 2.2: Key Management in WSN

PKC mechanisms that have been primarily used for WSN security are RSA [107]

and Elliptic Curve Cryptography [65]. A RSA based protocol *TinyPK* is proposed by Watro *et al.* in [129] for authentication and key agreement between an external user and a node in a WSN. The protocol uses challenge-response mechanism to authenticate the external user to a sensor node and to establish a session key between them in a secure manner. Since in RSA crypto system, the decryption and/or signature key generation (private key operations) are costly, TinyPK proposes the private key operations to be performed by the external user. The sensor nodes are required to perform the comparatively less costly public key operations i.e. signature verification and/or decryption. The challenge to a sensor node consists of two components: one is public key E_{Pub} of the external user E signed by the private key CA_{Pri} of the certification authority (CA), second component has a *nonce* and the checksum of E_{Pub} signed with the private key E_{Pri} of E. When a sensor node SN receives this challenge, it first verifies the public key E_{Pub} using the public key CA_{Pub} of CA. Then, the node SN verifies the signature on *nonce* and the checksum of E_{Pub} using E_{Pub}. Next, SN computes the check sum of E_{Pub} and compares it against the received checksum. The received *nonce* is compared with the previous nonce to ensure that the former is greater. After ensuring the authenticity of the received messages, the node SN prepares the response message. It picks the session key *TinySecKey* and encrypts *nonce* and *TinySecKey* with E_{Pub} and sends this response to the external user E. The external user E decrypts the response message using its own private key E_{Pri} and retrieves the *nonce* and *TinySecKey*. User E verifies the *nonce* and accepts *TinySecKey* as the session key shared with SN. The TinyPK protocol also provides key establishment between two sensor networks using Diffie-Hellman key exchange [38]. The key exchange is initiated by a node SN_1 that generates a random number $R_1 \in F_p^*$ (p is a large prime) and calculates $g^{R_1} \bmod p$. Here g is a generator for the group F_p^*. Node SN_1 sends this quantity $g^{R_1} \bmod p$ to another node SN_2. The second node SN_2 computes $g^{R_2} \bmod p$ using its own generated random number $R_2 \in F_p^*$ and sends $g^{R_2} \bmod p$ in parallel to node SN_1. Node SN_1 computes $(g^{R_2} \bmod p)^{R_1} \bmod p$ and SN_2 computes $(g^{R_1} \bmod p)^{R_2} \bmod p$. The common secret between SN_1 and SN_2 becomes $K = (g^{R_2} \bmod p)^{R_1} = (g^{R_1} \bmod p)^{R_2} = (g^{R_1 * R_2} \bmod p)$. Later, Kumar

and Das [68] found the "masquerade as sensor node to an unknowing external party" attack in TinyPK, wherein, if an adversary gets hold of E_{Pub} can generate its own *TinySecKey* and share it with E to communicate with E as a valid sensor node. Authentication of a sensor node is ensured in TinyPK by a node using a credential that consists of static Diffie-Hellman key pair along with a text string (node identity, date of manufacturing and so on) processed by the private key of CA. However, in [68], an alternate solution is provided to ensure authenticated key establishment with less overhead on the resource constrained sensor nodes. The protocol considers the presence of base station between an external user and a sensor node within the WSN. The external user sends a request to the base station to establish a session key with a sensor node. Base station first verifies the authenticity of the external user and then forwards this request to a sensor node encrypted with the secret shared between the base station and the sensor node. The sensor node can then directly communicate the session key to the external user by performing a symmetric decryption and two XOR operations. Although the protocol in [68] gives improvement over TinyPK protocol, it involves base station for each session key establishment that may result in traffic congestion at base station. Moreover, in real-life applications deployed in hostile environments, the continuous presence of base station may not be feasible.

Though, RSA public key operations are less costlier as compared to RSA private key operations, the researchers looked into the possibility of using Elliptic Curve Cryptography (ECC) [65] to construct security protocols for resource constrained WSN. ECC has an advantage over RSA because shorter keys with ECC are as strong as long key for RSA (160-bit ECC key similar security as 1024-bit RSA key). ECC uses an elliptic curve defined over a finite field and works on a group of points on that curve. An elliptic curve E_{F_p} defined over a field F_p of characteristic > 3 is the set of solutions $(x, y) \in {F_p}^2$ to the equation $y^2 = x^3 + \text{ax} + \text{b}$, $\{a, b\} \in F_p$ (where the cubic on the right has no multiple roots). It is the set of such solutions together with a "point at infinity" (O). In ECC, the main cryptographic operation is the scalar point multiplication that calculates $Q = k.P$, where P is a point on an elliptic curve, k is an integer and Q is another elliptic curve point that is obtained

when point P is multiplied with integer k. The scalar multiplication in ECC is actually the result of point additions, so, $k.P = P + P + P + \ldots + P$ (k times). The security of ECC depends on the difficulty of solving the Elliptic Curve Discrete Logarithm Problem (ECDLP). ECDLP states that given points P and $Q = k.P$ on a ECC, it is hard to find k. For each elliptic curve, there is a fixed base point G. In ECC, private key is taken as a large random integer k and the public key is the result of the multiplication of the private key k with the curve's base point G i.e. $k.G$.

Gupta *et al.* proposed authentication and confidentiality protocol, *Sizzle* [49], using elliptic curve Diffie-Hellman (ECDH) key exchange that is based on SSL. Sizzle offers scalable key management and provides end-to-end security. Sizzle is a small footprint implementation of an HTTPS stack that allows the security properties of SSL to be implemented in the embedded Internet. Sizzle does prove the feasibility of using public key cryptography in WSN, however, SSL-like handshake still incurs heavy overhead on resource constrained sensor nodes. In [84], two join protocols are proposed using ECC. Each node u_i is pre-deployed with the public key PK_S of sink and, its own public private key pair (PK_i, SK_i). The protocols use ECC and take G as a public generator point of the elliptic curve. The public key PK_i of a node is computed as $PK_i = SK_i * G$. With this pre-deployed information, each node u_i computes a key $K_{IS} = SK_i * PK_S$. Similarly, the sink computes $K_{SI} = SK_S * PK_i$. Here $K_{IS} = SK_i * PK_S = SK_i * (SK_S * G) = SK_S * SK_i * G = SK_S * (SK_i * G) = SK_S * PK_i = K_{SI}$ is the shared secret between the sink and the node u_i. The first protocol proposed in [84] is DJS (direct join to the sink) protocol that allows a sink and a node to join directly. In DJS, a new node u_{new}, having public-private key pair (PK_{new}, SK_{new}), computes the shared key K_{newS} with the sink. It selects a nonce N_{new} and sends a direct request $E_{K_{newS}}(N_{new}, ID_{new})$ to the sink. The sink, upon receiving this request, decrypts the message using the shared secret K_{Snew} (= K_{newS}), verifies the identity of node u_{new} and then randomly generates a new session key K'_{Snew}. The sink sends the join response to node u_{new} as $E_{PK_{new}}(N_{new}, ID_S, K'_{Snew})$. Since the join response is encrypted with the public key PK_{new} of node u_{new}, u_{new} decrypts this message with the help of its private key

SK_{new}. With nonce N_{new}, node ensures the authenticity and freshness of the key.
With the second protocol, indirect join to the sink (IJS) [84], a new node u_{new} can join the network with the help of an already authenticated neighbor node u_r. An indirect request is sent by node u_{new} to the sink S for establishing a session key with node u_r. The new node sends request $E_{K_{newS}}(N_{new}, ID_{new})$ to node u_r. The neighbor node u_r forwards this request to the sink S without any alteration. Node u_r sends the request via intermediate nodes that are trusted to forward the request towards the sink. Though, intermediate nodes are involved in the routing, only nodes u_{new} and the sink S can decrypt the messages encrypted with the shared key K_{newS} (= K_{Snew}). Likewise, the messages encrypted with K_{rS} (= K_{Sr}) are accessible only to node u_r and the sink S. Sink S responds back to u_r with $E_{K_{Sr}}(N_{new}, ID_S, PK_{new})$. Node u_r decrypts the message with the shared key K_{rS} (= K_{Sr}), picks the public key PK_{new} of node u_{new} and reconstructs the response as $E_{PK_{new}}(N_{new}, ID_r, K_{rnew})$. Upon receiving this response from u_r, node u_{new} decrypts the message with its private key SK_{new}, verifies the key K_{rnew} and the nonce N_{new} and establish the link with node u_r. Apart from having the public key setup, in join protocols, to establish a session key with any neighbour node, the base station support is needed. Recently, Hayouni and Hamdi [50] presented an authentication and pair-wise key establishment scheme that is based on elliptic curve public key cryptography. The scheme works with ECDH key exchange procedure for establishing pair-wise keys agreement between the sensor node.

Although, PKC based schemes have been found feasible for key establishment, since sensor nodes have inherent resource constraints [113], in order to avoid the overheads associated with public key set up, most of the schemes in WSNs still prefer symmetric key primitives. A substantial progress has been made for symmetric key establishment amongst communicating nodes by providing schemes based on pre-loaded shared keys/mathematical constructions ([108]), which offer a viable alternative to the use of public key cryptography. With symmetric key pre-distribution, either the secret key itself or some keying material to later establish the key is distributed to all sensor nodes prior to deployment. The simplest way for establishing symmetric key in WSN is to distribute a common key to all

the nodes before deployment i.e. single network wide key pre-distribution. However, with this approach, compromise of one node results in the compromise of the entire network. Second approach is to assign a random set of keys to each node from a large key pool and the nodes can communicate with other nodes, if they have at least one key common in the key set. Another option is pair-wise private key sharing. The key can be shared prior to deployment or may be established after the deployment. The trivial approach to pre-distributed pair-wise key sharing is to assign one key to each node and to share it with all other nodes in the network. In a network of n nodes, this arrangement requires each node to store n-1 keys, which is not practical for large number of nodes in WSN. A more practical mechanism is to establishment pair-wise key once the nodes are deployed in the network. We discuss different post-deployment pair-wise key establishment protocols using Diffie-Hellman key exchange [38], with the help of trusted setup server [100], using transient master secret [143] and with pre-distributed polynomial shares followed by the proposed pair-wise key establishment and key update protocol in Chapter 3.

In most of the applications, WSNs work in a distributed manner wherein the sensor nodes in a WSN collaborate together to communicate the object or event related data to their respective cluster heads. In such scenarios, the secure communication within the WSN takes place using a secret shared between a group of nodes that may be provided during the deployment or during the network operations. Group-key distribution is one of the possible approaches wherein a node can extract the group shared secret from the keying material broadcasted by a group manager (usually the cluster head) [104].

Various group-key broadcast protocols have been proposed in the literature to secure the group communication within a set of sensor nodes in WSN [28]. Recently, Wang et al. [124] discussed a bilinear pairing based group authentication and key distribution scheme. The scheme allows any user to easily generate a group for communication without involving a group manager. The group members are able to perform authentication and can distribute group key without having prior knowledge of the number of members attending the group communication. The

content of communication remain secret even if a group member leaves the group communication or join in. However, each node needs to perform multiple bilinear pairing operations and scalar multiplications. A key freshness scheme is presented in [46] that is based on XOR and left shift operations that allows key refresh at all the group member nodes without transmitting inter-node message. To each group member, the group manager sends an encrypted magic word and the group key is extracted from that magic word. This scheme requires the group manager to transmit n encrypted messages for n nodes in the group at each membership change, resulting in high communication cost.

We studied various group-key broadcast schemes that use CRT based secret sharing and observed that the CRT based approach offers light-weight solution to group-key broadcast problem in wireless sensor network. Zheng *et al.* [141] and Zhou *et al.* [142] proposed group-key management schemes using CRT but did not consider the XOR-overflow problem. This weakness is addressed by Bhaskar and Pais [13] wherein they introduced a multiplicative factor for each individual node. In [13], however, the group manager distributes the node specific multiplicative factor individually to the nodes which contributes to increased network traffic. Moreover, all of these schemes suffer from man-in-the-middle attack, since the CRT congruence value is distributed in plain to the individual nodes, if the attacker captures and changes this value, the individual nodes will not be able to compute the group key correctly. Moreover, the nodes do not have any mechanism to ensure the authenticity of the CRT congruence value. Liu *et al.* [80] had proposed the authenticated group-key distribution using CRT set-up but the scheme requires the resource constrained nodes to solve the system of congruence using CRT and moreover, the communication overhead on nodes as well as on the group manager is high.

Even with the most efficient group key distribution mechanism, if a node misses out on one or more broadcasts, it may not be able to extract the group key used for a particular session. In a distributed WSN, with the sensor nodes having limited resources and the communication cost is much higher than the computation cost, sending an explicit request to the group manager to obtain the session key infor-

mation would be an overhead. Using self-healing an authorized node is allowed to recover the key of some previous session(s) using the broadcast message received in a subsequent session, without requesting for the keying material explicitly from the group manager. Furthermore, when a node is not able to recover the key with self-healing because it has lost the current session's broadcast or when it has lost more than one broadcast and does not want to wait for the subsequent broadcasts, mutual-healing comes in picture. With mutual-healing, a node may request its neighboring nodes to provide the missing keying material. We discuss the state-of-the-art in self-healing and mutual healing enabled group key distribution and present our proposed healing enabled group key distribution protocols in Chapter 4.

2.1.3 Node Capture Detection and Revocation

Despite robust key management to secure the communication within a WSN, the sensor nodes deployed in unattended hostile terrains are vulnerable to physical attacks including node capture attack. Node capture attack is the ability of an attacker to access (and eventually change) the internal state of a sensor node [12]. An attacker gains full control over a sensor node through a direct physical access and then easily extract cryptographic primitives. The attacker can also apply a reverse engineering process on the captured node to obtain unlimited access to the information stored on its memory chip resulting in substantial damage to the entire WSN [63]. The node capture attack in wireless sensor networks (WSNs) can be decomposed into three stages: physical capture of node, redeployment of compromised node, and rejoin the network for various insider attacks [39]. An attacker getting hold of a node physically may reprogram and redeploy the node.

The possibility of node capture attack is classified based on severity and duration by Becher et al [10]. The simple attacks start from manipulating the radio communications, to influence sensor readings and reading out RAM or flash memory in whole or in part. In more severe attacks, the adversary may gain complete read/write access to the micro-controller. The attack that vary based on the duration are termed as short, medium and long attacks. In short attacks, adversary

takes less than 5 mins and simply create plug-in connections and transfer few data. Medium attacks take about 30 mins in which some mechanical work such as soldering can be carried out. Long attacks are possible only in specialized labs which may take hours or days depending on the intended damage. A node capture attack is aimed to be performed with an optimal cost in terms of time and efforts of an adversary.

The node capture attack is modelled using different schemes to analyze the feasibility and impact of the attack. In the proposal given by Tague *et al.* in [115], a model for node capture attack is formalized wherein it is considered that an attacker can collect the network related information by eavesdropping on the wireless medium and passively learn about the protocol states and network operations. The attacker can also have active participation in the network activities by injecting malicious packets into the network. After collecting sufficient details of network information, the physical capture of nodes can be carried out in planned manner to optimize the attack performance. An event based attack decomposition model is used to develop relevant performance metrics for node capture attack. Another formulation by Tague *et al.* is presented in [116] in which the node capture attack is modelled as a nonlinear integer programming minimization problem. The model uses greedy heuristics for approximation of minimum cost attack. The capture of each individual node is based on the information gathered from the already captured nodes to increase the vulnerability. The greedy node capture approximation using vulnerability evaluation (GNAVE) proposed in [116] provides increased attack efficiency with more compromised traffic using fewer captured nodes. However, the model does not discuss the time required in the execution to compromise the whole network. An epidemiological model is proposed by De *et al.* in [36] that identifies the probability of the compromise of the entire network or the sizes of the parts of the affected network. The model proposes to start the capture from the most vulnerable node, and it assumes that through wireless communication, the neighboring nodes can be compromised. The capture process is modelled to be propagated using the epidemic theory. Bonaci *et al.* modelled the physical capture of sensor nodes in a WSN using a control-theoretic frame-

work in [18]. The dynamic model maps the problem of network security into a control theory problem and efficiently specifies the behavior of the network under attack. The optimal control theory is used to achieve the control parameter in terms of minimal revocation rate that ensures network stability under attack. However, the model does not fully consider the effects of node capture attack. In [93], Mishra and Turuk presents a probabilistic model of the process of information gathering by an attacker as a birth and death process. The model shows that the time required to gather information depends on the strength and configuration of an adversary. The model can also compute the expected chunk of information that an attacker can possess at any given time. Chi Lin *et al.* in [75] model the node capture attack using matrix approach by proposing a matrix algorithm. The matrix algorithm takes minimum resources and produces maximum destructiveness. This model pays less attention to the relationship between attack efficiency and the cost of attack and, works only for random key pre-distribution schemes. Most of the node capture attack models discussed above consider the theoretical aspect and models the node capture attack from adversarial aspect.
While modelling the capture attack, usual assumption is that in physical attacks, attacker can access a node for an extended period of time without getting noticed. However, in normal WSN operation, nodes continuously interact with their neighboring nodes, an unusual absence resulting from physical capture of a node can be noticed with periodic monitoring. Various protocols have been proposed, such as [87] and [32], for detection of node capture by continuous monitoring. However, with the malicious neighbor collusion, the non-captured malicious nodes intentionally report an absent node to be present in the network, and because the absence is not noticed within the defined threshold period, the adversary may reprogram and redeploy the captured node in the network. In chapter 5, we review the existing approaches to detect node capture attack in a WSN and then discuss a secure and efficient protocol for node capture detection using program integrity verification.

The node that becomes a victim of node capture attack works as an insider attacker and poses a threat to the entire network. As a reprogrammed and rede-

ployed node, the victim can disrupt the routing in the network, can communicate with other nodes as a valid member of a group. Therefore, it is mandatory to revoke the node from the network as soon as it is found to be a victim of node capture attack. A Node may be revoked by revoking its secret keys so that it can no longer participate in any future network activities. Node revocation using public key setup as well as with symmetric key based approaches are proposed in the literature. Recently, in [84], the revocation protocol is proposed that allows the renewal of keys and the revocation of compromised nodes. The protocol [84] uses public key setup and encrypt the communication using the current session key to share the renewed public keys, or the network key. In such case, if the current session key is compromised then the renewed keys will be accessible to an attacker. Also, each node needs to store the public key of the sink as well as its own public private key pair and a network key. This incurs more storage overhead. The scalar multiplication is involved in computing initial key and, for each renewal, symmetric encryption and decryption needs to be performed by the node that results in additional computational overhead. Chuang *et al.* [31] proposed a node revocation scheme that utilizes PKC, certificate revocation list (CRL) and a one-way hash chain. In [31], each node is assigned a certificate signed by the base station. The certificate contains a node's unique identity that makes a node authorized to access data authentication and node revocation services. A certificate also has a unique identity. On detection of a compromised node u_j, a node u_i broadcasts a compromised revocation vote (CRV), along with the certificate ID of the compromised node, to all the neighbor nodes. When a neighbor node u_k receives a CRV message, it validates the authenticity of the vote using the hash value that is attached in the CRV. If the vote is found authenticated, the revocation vote count against node u_j is increased by node u_k. The node u_k cuts the link with the node u_j, if the revocation votes against u_j exceeds a pre-defined threshold t. The scheme in [31] demands high storage cost at node that needs to store its own public-private key pair, the public keys of its r neighbors and, r hash values corresponding to r neighbors which are used for revocation vote verification. The scheme also requires strict time synchronization to maintain the validity of

the node's certificates. Various symmetric key based revocation schemes also exist that are categorized as centralized, distributed and hybrid approaches of key revocation [83] [45]. We discuss the existing key revocation schemes and then present proposed node revocation and key update protocol in Chapter 6.

2.2 Primitives Used in the Proposed Protocols

In the subsequent chapters, we have presented security protocols that use various security primitives. We give an overview of those primitives in this section.

2.2.1 Pseudo Random Function

Pseudo random function (PRF) [11] is a family of functions for which the input-output behavior of a random instance of the family is "computationally indistinguishable" from that of a random function. PRF is a central tool in the design of symmetric key cryptography protocols. PRF can be used for symmetric encryption as well as for generating message authentication codes. A PRF is a deterministic function f: $\{0,1\}^n \rightarrow \{0,1\}^n$ that takes two inputs x, $k \in \{0,1\}^n$ and is computable in polynomial time. Here, k is the function index, $f(x,k) = f_k(x)$ and serves as the hidden random seed. Essentially, a true random function is a look-up table with random entries. A function $x \rightarrow f_k(x)$ is considered a good PRF, if it looks like a random function. If we take a black box that computes the function f:$\{0,1\}^n \rightarrow \{0,1\}^n$ and if there is a way to decide if f is a true random function, or a $f_k(x)$ with random k, then f is not a good PRF.

Formally, let F: $\{0,1\}^*$ x $\{0,1\}^* \rightarrow \{0,1\}^*$ be an efficient length preserving, keyed function. F is said to be **pseudo-random function** [59], if for all probabilistic polynomial time distinguisher D, there exists a negligible function $\varepsilon(n)$, such that:

$$|Pr[D^{F_k(.)}(n) = 1] - (Pr[D^{f(.)}(n) = 1]| \leq \varepsilon(n)$$

where k and f both are chosen uniformly at random and, f is chosen from the set of functions mapping n-bit strings to n-bit strings.

2.2.2 Bivariate Polynomials

A bivariate t-degree polynomial [78] $f(x,y) = \sum_{i,j=0}^{t} a_{ij}x^i y^j$ over a finite field F_q , where q is a prime number that is large enough to accommodate a cryptographic key. A symmetric bivariate polynomial has the property of $f(x,y) = f(y,x)$.

2.2.3 ID based Public Key Infrastructure with Bilinear Pairings

Identity based public key cryptography has widely been used for efficient key management as an alternative to certificate-based public-key infrastructure (PKI). In last two decades, bi-linear pairings are used to construct identity based crypto primitives. A trusted Key Generation Center (KGC) generates two cyclic groups G_1 (additive) and G_2 (multiplicative) defined over q (a large prime number) and, a mapping $e : G_1 \star G_1 \rightarrow G_2$.

The mapping e is ***Bi-linear mapping*** [66] satisfying the below mentioned conditions:

1. Bi-linearity: $\forall$ P, $Q \in G_1$ and $\forall$ a,$b \in Z_q^*$, $e(aP, bQ) = e(P,Q)^{ab}$
2. Non-Degeneracy: $\exists$ P $\in G_1$ and $Q \in G_1$ such that $e(P, Q) \neq 1$
3. Computability: An efficient algorithm exists to compute $e(P, Q)$ for any P, Q $\in G_1$

Bilinear mapping is called bilinear pairing as it associates pairs of elements from $G_1 \star G_1$ with elements in G_2. The groups G_1 and G_2 are isomorphic to each other as their group order is same and they are cyclic. Typically, group G_1 is an elliptic curve and group G_2 is a finite field.

For setting up the ID based PKI infrastructure:

- KGC selects a generator $P \in G_1$ randomly and defines two cryptographic hash functions:

  H_1: $\{0,1\}^* \rightarrow G_1$ and,

  H_2: $G_2 \rightarrow \{0,1\}^l$ (l is the bitlength of plain text)

- KGC randomly selects an integer $s \in Z_q^*$ and sets its own public key $P_{pub} = sP$

- KGC keeps s as secret master key known only to itself and publishes system parameters $params = \{G_1, G_2, q, P, P_{pub}, H_1, H_2\}$
- Each user privately shares its identity information ID to KGC
- For each user, KGC computes public key as $Q_{ID} = H_1(ID)$ and corresponding private key as $S_{ID} = sQ_{ID}$ and, secretly sends the public-private keys to the individual user.

Two legitimate parties can communicate securely with the help of identity based encryption scheme using bi-linear pairing such as Boneh-Franklin scheme [19]. A user B encrypts a plain text message $m \in \{0,1\}^l$ for another user A as follows:

(i) computes the public key of user A as $Q_A = H_1(ID_A)$

(ii) randomly selects an integer $r \in [1, n\text{-}1]$ and calculates $R = rP$

(iii) computes $c = m \oplus H_2(e(Q_A, P_{Pub})^r)$

User B now transmits the ciphertext (R, c) to user A who can use her private key S_{ID_A} to obtain the plain text m as $m = c \oplus H_2(e(S_A, R)$.

User A can retrieve m, since $e(S_A, R) = e(s.Q_A, r.P) = e(Q_A, s.P)^r = e(Q_A, P_{Pub})^r$.

Computational Assumptions.

Elliptic Curve Discrete Logarithm Problem (ECDLP). For a given elliptic curve E defined over F_p, a point $P \in E(F_p)$, the elliptic curve discrete log problem is to find $s \in Z$ such that $Q = s.P \in E(F_p)$.

Bi-linear Diffie-Hellman Problem (BDHP). Given P, aP, bP, cP for some $a, b, c \in Z_q^*$, there is no polynomial time algorithm to solve $e(P, P)^{abc} \in G_2$.

The security of bi-linear pairing based cryptosystem is based on the Bi-linear Diffie-Hellman (BDH) assumption. In the above mentioned communication between users A and B, if an attacker wants to retrieve m from (R, c), she needs to compute $e(Q_A, P_{Pub})^r$ using the publicly available parameters (P, Q_A, P_{Pub}, R) that is equivalent to solving BDHP.

Implementation Issues. As such the implementation of any efficient cryptographic schemes on wireless sensor networks is a difficult task wherein platform specific features must be analysed to get best results.

The identity based cryptosystems have been implemented either using Weil-pairing

or Tate-Pairing [21]. Most implementations consider Tate-pairing as it is more efficient as compared to Weil-pairing. For example, Weil-pairing takes 1.59ms and Tate-pairing takes 0.78ms (almost half of the time compared to Weil-pairing) in an experiment carried out on Intel i5 3.20GHz processor with 4GB RAM on an elliptic curve $E_{F_{103}}$: $y^2 = x^3 + x + 18$ embedded degree $k = 6$ and $n = 19$ (order of the point $P \in E_{F_{103}}$).
In the literature [22], some implementation issues related to PBC have been highlighted as below:

- For most of the people involved in implementation, it is difficult to understand the pairing computation
- the PBC shows key escrow problem, the KGC has the master key that it uses to generate the private keys of all the users
- For the security protocols based on elliptic curve cryptography (ECC), it is assumed to be infeasible to find the discrete logarithm of a random elliptic curve element with respect to a publicly known base point (ECDLP). The main advantage of using ECC is a smaller key size. An elliptic curve group provides same level of security as provided by an RSA-based system with a large modulus and larger key. For example, a 160-bit elliptic curve public key provides the 80-bit security level similar to 1024-bit RSA public key. The security of PBC is defined by the intractability of Elliptic Curve Discrete Logarithm Problem (ECDLP) in the group $E(F_q)$. Although, the ECC works with the elements that are defined over its base field F_q, the pairing based cryptography (PBC) that uses the elliptic curve $E(F_q)$ works with the elements and functions that are defined over the extension field F_{q^d} resulting in heavy computations in PBC.

2.2.4 Chinese Remainder Theorem

The Chinese Remainder Theorem (CRT) can be used to share a secret amongst a set of users using threshold secret sharing [90]. The CRT [43] is defined as:
Given two sets of integers $\{a_i | i = 1, 2, \ldots, k\}$ and $\{b_i | i = 1, 2, \ldots, k\}$, which satisfy

the conditions as follows:

i) a_i, a_j are co-prime when $i \neq j$; (i = 1, 2, ... k and j = 1, 2, ... k)

ii) $0 \leq b_i < a_i$; (i = 1, 2, ... k)

iii) For an integer X -

$0 \leq X < \prod_{i=1}^{k} a_i$; ($i$ = 1, 2, ... k)

iv)
$$\begin{cases} X \equiv b_1 \ mod \ a_1 \\ X \equiv b_2 \ mod \ a_2 \\ \vdots \\ X \equiv b_k \ mod \ a_k \end{cases}$$

X has one and only one solution as:

$X = b_1 A_1 A_1^{-1} + b_2 A_2 A_2^{-1} + \ldots + b_k A_k A_k^{-1}$ **(mod** N**)**,

where $N = a_1 a_2 a_3 \ldots a_k$, $A_i = N/a_i$ and A_i^{-1} is the multiplicative inverse of A_i modulo a_i.

We will be using this concept of CRT based secret sharing in the thesis in the chapters followed.

- In section 4.5, we have given CRT based Symmetric Key Mutual Healing Protocol, wherein CRT based secret sharing, as mentioned in section 2.2.4, is used to construct the broadcast message containing the secret session key as per Algorithm 4.1.

- In section 6.5, we have proposed Node Revocation and Key Update Protocol (NRKU) that uses CRT based secret sharing for distributing the updated session key to non-revoked members of the group. The key distribution process given in Algorithm 6.1 (section 6.5.2) provides the details of the same.

2.3 Trusted Platform Module

In the proposed system model (Refer Figure 1.9), the cluster heads are assumed to be equipped with Trusted platform module (TPM). In this section, we give an overview of the TPM with emphasis on the sealing functionality that is used in

the protocol for node capture detection. Trusted Platform Module(TPM) [122] is a dedicated microprocessor and an international standard for secure co-processor. It is designed for hardware security wherein the cryptographic keys are integrated into devices. Endorsement Key (EK) is unique to each TPM and is used to identify the TPM. It is typically a pair of public-private keys and the private key is embedded into the TPM and never leaves it. The private key is never needed to be known outside the TPM that prevents everyone (even the user) from knowing the private key value. TPM architecture has unique feature in terms of the Secure Platform Configuration Registers (PCRs). PCRs store the integrity metrics used to measure the integrity of any code prior to its execution. The integrity measurements are extended from boot loader to operating system and then to the application code.

TPM provides a unique capability of *Sealing* any data block with TPM's platform configuration. The sealed data block can only be unsealed, when during the unsealing, the TPM has the same platform configuration. This feature can be used to secretly store the data in TPM and also verify the integrity of the device which is embedded with TPM. For this purpose, TPM provides *TPM_Seal*() and *TPM_Unseal*() commands. Figure 2.3 shows the sealing and unsealing using TPM.

The data within the TPM is encrypted with the public key of the TPM. This encrypted data is bound with the current platform configuration (stored in PCRs) of the same TPM. To retrieve this data block, the private key of the TPM is needed and at the time of retrieval, TPM has to be in the same configuration in which the data was sealed. *TPM_Unseal*() command takes the TPM's private key and the current platform configuration to unseal the sealed data block. If the current configuration matches the initial configuration, it will unseal the data block with the help of the private key. A simple abstraction of *TPM_Seal*() and *TPM_Unseal*() can be:

PSeal(Platform_Configuration_at_sealing_time, public_key_of_TPM, data_to_be_sealed)

PUnSeal(Platform_Configuration_at_unsealing_time, private_key_of_TPM, sealed_data)

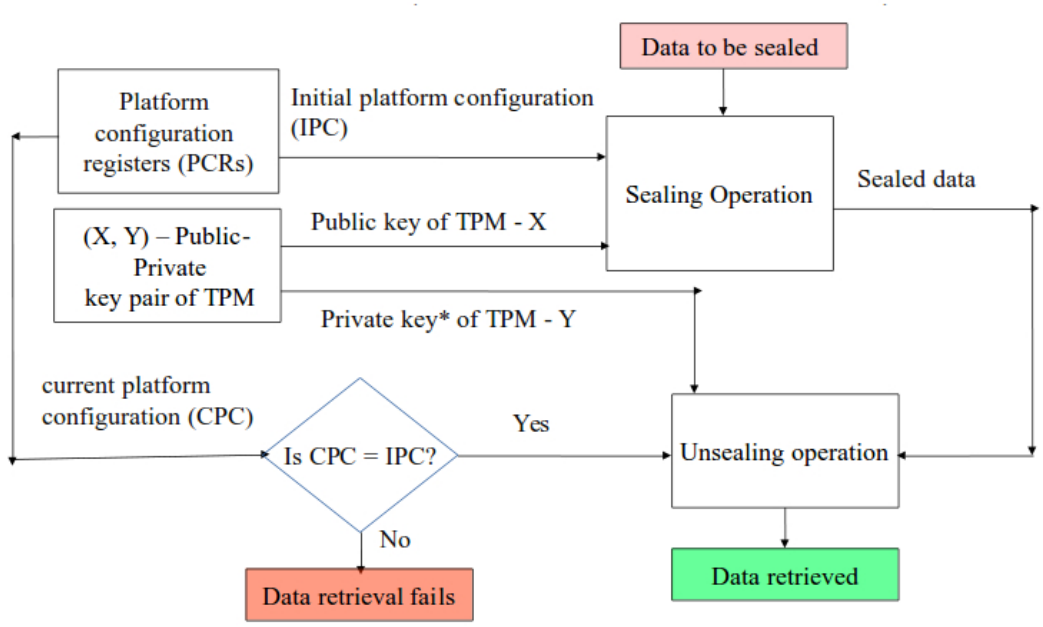


Figure 2.3: Trusted Platform Module (TPM) Sealing and Unsealing

2.4 Conclusion

In this chapter, we presented an overview of security in WSNs with a brief explanation of symmetric and asymmetric cryptography and their use in WSN security. We discussed key management in detail with classification of key management techniques with example protocols. We then presented the background study carried out in the context of node capture detection and node revocation. At the end, we introduced the cryptographic primitives and trusted platform module that are used in the proposed protocols in the subsequent chapters.

CHAPTER 3

Pair-Wise Key Establishment and Key Update

Pairwise key establishment is one of the essential security services that enables a pair of sensor nodes to securely communicate with each other using cryptographic techniques. Due to the resource constraints on sensors, it is not preferable to use traditional key establishment approaches using public key cryptography and key distribution center. With the basic probabilistic and q-composite key predistribution schemes, the fraction of affected pairwise keys quickly increases with the increase in the number of compromised nodes. In the random pairwise keys scheme, the network size is strictly limited by the desired probability of allowable neighbor nodes that a sensor can communicate with and sharing of a pairwise key between two nodes. The polynomial share based key establishment is deterministic and allows any two nodes to share a pair-wise key directly or indirectly. However, most of the existing polynomial share based schemes use static arrangement of nodes. In a dynamic WSN, during the entire network life-time, a node is mobile and may wish to communicate with any neighbor that it comes in contact with. Moreover, the existing polynomial share based schemes consider just one-time key establishment between a pair of nodes, therefore, compromise of a pair-wise key results in the compromise of the link between that pair of nodes for all forthcoming sessions as well. In this chapter, we present an overview of the existing pair-wise key establishment schemes in WSN using trusted server and without the help of trusted server. We then discuss, polynomial share based key establishment schemes wherein the pair-wise key is established using pre-distributed polynomial shares. We then present the proposed pair-wise session key update protocol for a dynamic WSN. The proposed protocol allows deterministically es-

tablishing a master secret between a pair of nodes. The master secret is then used to update the pair-wise key for each session with the help of random inputs from both the nodes. The proposed protocol resists replay, impersonation, known-key, worm and sink hole attacks and ensures forward secrecy, key freshness, and mutual key control. Due to the underlying multiple polynomial based key used as master secret, we also achieve high resilience to node capture attack. In our proposed solution, we consider the pair-wise key establishment between base station and cluster head pair and between a pair of cluster heads.

3.1 Introduction

Various key management schemes have been proposed to establish pair-wise key between a pair of nodes. For pair-wise symmetric key setup, a convenient method is using a public-key cryptography based protocol to bootstrap secure connections. However, for resource constrained sensor nodes, it is preferable to avoid public-key cryptography that is computationally expensive. WSNs also lack the infrastructure support and therefore traditional key establishment protocols suitable for infrastructure supported wireless networks such as Kerberos [91] also can not be used in wireless sensor networks. Researchers have proposed using symmetric-key algorithms with base station as a trusted agent for key setup. With trusted server setup, the server authenticates and helps the nodes to agree to a key for secure communication. For example, in [100], Perrig *et al.* proposed SNEP protocol that can be used to establish a shared session key between two nodes A and B with the help of base station as the trusted server. With this trust setup, Each node shares an encryption key and an authentication key with base station. A node initiates the key-agreement protocol by sending its identity and a nonce to another node. The receiver node chooses a nonce and sends identities and nonces of both the nodes along with authentication code computed using its encryption key shared with base station. Base station individually sends the pair-wise key between both the nodes to respective nodes. For this, base station encrypts the new key with the encryption key it shares with the node and computes the authentica-

tion code of another node's identity, nonce and encrypted pair-wise key using the authentication key. The node-to-node key agreement between two nodes A and B takes place as follows:

$A \Rightarrow B$: N_A, A

$B \Rightarrow S$: N_A, N_B, A, B, $MAC_{K_{BS}}(N_A||N_B||A||B)$

$S \Rightarrow A$: $E_{K_{AS}}(SK_{AB})$, $MAC_{K'_{AS}}(N_A||B||E_{K_{AS}}(SK_{AB}))$

$S \Rightarrow B$: $E_{K_{BS}}(SK_{AB})$, $MAC_{K'_{BS}}(N_B||A||E_{K_{BS}}(SK_{AB}))$

Here K_{AS} and K'_AS are encryption and authentication keys respectively, shared between base station and node A. Similarly, K_{BS} and K'_BS are keys between base station and node B. N_A and N_B are the nonces selected by nodes A and B respectively and SK_{AB} is the pair-wise key agreed between the two nodes with the help of base station. Although, we do assume the existence of base station as a central trusted authority in a WSN, dependence on base station for each pair-wise key agreement results in communication overhead. Moreover, in most of the real-life WSN applications, the continuous presence of trusted infrastructure is not feasible. Therefore, the pair-wise key establishment that does not require the continuous presence of the base station is preferred.

There are pair-wise key establishment protocols that use transitory master key, wherein each node is pre-deployed with a master key. Each node computes its individual secret using this master key and then the master key is deleted. A pair of nodes can then use their individual secrets to establish a pair-wise key. One such scheme based on transitory master key called LEAP (Localized Encryption and Authentication Protocol) is proposed by Zhu *et al.* in [143]. In LEAP, each node is pre-configured with master secret K_I. A node u computes its individual secret as $K_u = f(K_I, u)$ ($f()$ is a secure one-way function). Two nodes u and v can establish a pair-wise secret as $K_{uv} = f(K_u, v) = f(K_v, u)$. In LEAP, however, if the master secret is leaked, then an adversary w can establish the pair-wise key with any node x in the network as $K_{wx} = f(K_w, x) = f(f(K_I, w), x)$. To overcome this security issue, they extended this scheme [143] by setting the life time of a master key. The total network life time is divided into time slots T_1, T_2, ... T_m and a master key is considered to be valid only within a time slot. Therefore, an

adversary getting hold of the master secret in a time slot T_i will be able to establish pair-wise key with only those nodes that have joined the network in the time slot T_i. With this solution, however, the pair-wise node-to-node connectivity is limited to a specific time slot and deciding on the interval of a time slot is another challenge. In [17], polynomial share based key establishment was proposed that gives promising solution to the issue of symmetric pair-wise key establishment in WSNs. We extensively studied the polynomial share based pair-wise key establishment schemes that allows a pair of nodes to establish a secret, without taking help of the base station, using pre-distributed polynomial shares. In the subsequent section, we discuss the details of some of the existing schemes using polynomial share based pair-wise key establishment.

3.2 Polynomial Share Based Pair-Wise Key Establishment

For the post deployment pair-wise key establishment, many polynomial based key schemes have been proposed in last two decades. The core element of this family of schemes is the concept of bivariate t-degree polynomials $f(x, y)$ over a finite field F_q. Here, q is a large prime number that can accommodate a cryptographic key. In the most basic scheme [17], each sensor node is assigned a share from a polynomial, and to establish a pairwise key, each node requires to compute the key from its own polynomial share using the unique identity of the other node. For example, let the two nodes be u_i and u_j, having unique identities Id_i and Id_i respectively. A polynomial share assigned to node u_i is $f(Id_i, y)$ and that to node u_j is $f(Id_j, y)$. Node u_i computes $f(Id_i, id_j)$ and node u_j computes $f(Id_j, Id_i)$. Since the polynomial $f(x, y)$ is symmetric, $f(Id_i, Id_j) = f(Id_j, Id_i)$ and serves as the pair-wise key between nodes u_i and u_j. This scheme provides excellent connectivity and support for mobile nodes, as every node can create a pair-wise key with any other node in the network. Moreover, this scheme does not hinder the extensibility of the network, i.e. it is possible to add new nodes after the original deployment. However, this basic scheme has a major drawback: the actual

resilience of the network (i.e. ability to cope with stolen credentials and rogue nodes) is low, as an adversary only needs to subvert t nodes to control the whole network. Therefore, various authors have tried to improve the basic scheme over the years, but at the cost of sacrificing network connectivity or increasing memory consumption.

Over the years, number of polynomial share based schemes have been proposed. In [78], Liu and Ning proposed a general framework that combined the concept of polynomial-based key pre-distribution with the key pool idea used in [42]. The framework has three step process to establish a pair-wise key between two nodes:

- **Setup phase.** The setup server randomly generates a pool $P = \{f_1, f_2, \ldots, f_m\}$ of bivariate t-degree polynomials over the finite field F_q. Set up server assigns each node u_i a unique identity Id_i, picks a subset of polynomials P_i from the polynomial pool P and assigns the polynomial shares of these polynomials to node u_i.

- **Direct key establishment.** Two sensor nodes u_i and u_j having the shares from sets P_i and P_j and having a share from a common polynomial f_l, can establish the pair-wise key directly using the polynomial-based key pre-distribution as given in [17]. The polynomial share discovery can be carried out using either *pre-distribution* or *real time discovery*.

  - **Pre-distribution.** Setup server pre-distributes certain information to the nodes, so that given the identity of another node, a sensor node can determine whether it can establish a pair-wise key with that node. For example, each node may be preloaded with the unique identities of those nodes with whom it can directly setup a pair-wise key. However, for nodes, that join the network on the fly, setup server has to inform some existing nodes about the addition of these new nodes. Moreover, with this approach, an attacker may also know the distribution of the polynomials and thus can precisely target at certain nodes in order to learn polynomial shares of a particular bivariate polynomial.

  - **Real time discovery.** To avoid the issues with pre-distribution for poly-

nomial share discovery, nodes begin the real time discovery by exchanging the identities of polynomials of which they both have shares to identify the common polynomials. The nodes may choose to protect the identities of the polynomials by challenging the other party to solve puzzles instead of disclosing the identities of the polynomials directly. For example, node u_i may broadcast an encryption list α, E_{K_l} (α), l = 1, 2, ..., $|P_i|$, where K_l is a potential pair-wise key the other node may have. If another node u_j correctly decrypts any one of these, it can establish a pair-wise key with node u_i. This approach, however, introduces additional communication overhead.

– **Path key establishment.** When direct key establishment fails, two nodes can establish a pair-wise key with the help of other nodes. To establish a pair-wise key with node u_j, a node u_i needs to find a path (key-path) between itself and node u_j such that any two adjacent nodes in the path can establish a pair-wise key directly. Once the path is found, node u_i can initiate a request to establish a pair-wise key with node u_j through the intermediate nodes along the path. The main issue in this phase is the path discovery problem, which specifies how to find a path between two sensor nodes. This issue can also be addressed using predistribution or real-time discovery techniques.

Two actual instantiations of this general framework are proposed by the authors of [78]. The first one considers a random subset assignment, where the polynomials are randomly selected from the pool of polynomials and the shares are assigned to nodes. Here, the probability of direct key establishment is low and there may be situations wherein two nodes do not have any polynomial share in common and therefore, will not be able to establish a pair-wise, even though geographically they might be close neighbors. This issue is addressed by proposing another instance of key distribution from the pool as grid based key predistribution, in which nodes are arranged in m x m matrix with each node assigned an intersection point at the grid. The setup server randomly generates 2*m bivariate polynomials. Each row and each column in the grid is assigned one

polynomial from the pool of these 2*m polynomials. Each node gets two polynomial shares, one share corresponding to its row and the other share corresponding to its column. Two nodes can establish a pair-wise key directly, if they both share a common polynomial, i.e., they are either in the same column or in the same row of the grid. If two nodes do not have any row or column in common, they go through path discovery to establish a pair-wise key (Refer Figure 3.1). The

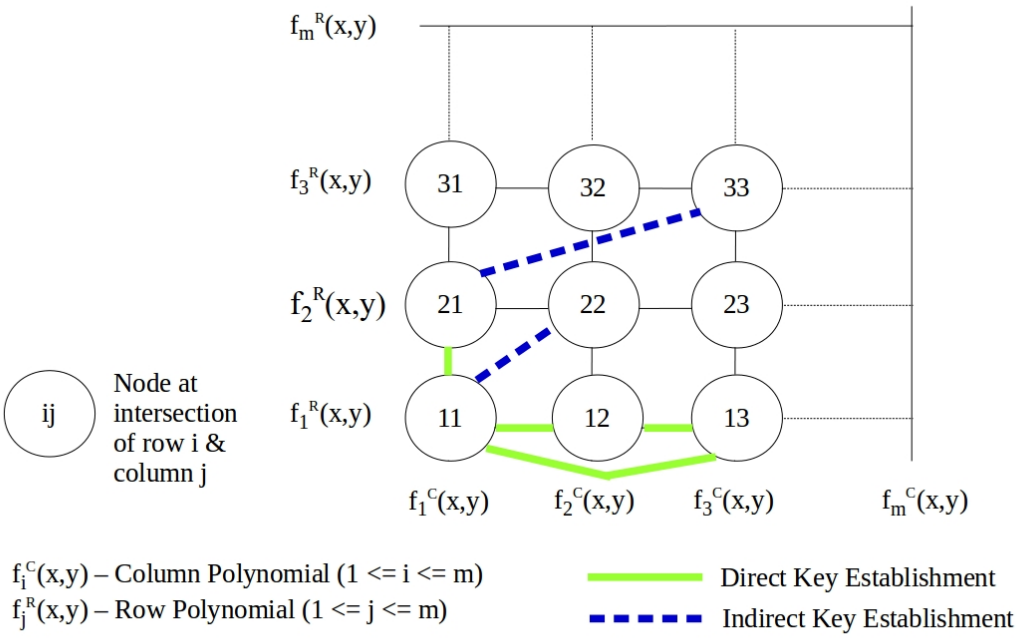


Figure 3.1: Two Dimensional Grid based Polynomial Key Distribution

grid-based idea was extended by Kim *et al.* [64] to increase the direct key-sharing probability in the grid-based basic scheme. Instead of a 2-dimensional grid, N nodes are arranged in k-dimensional grid with $k * m$ number of t-degree bivariate polynomials with $N = m^{1/k}$. A node identity contains k components and if two nodes have any one component in common, they can establish a pair-wise key directly. In case the two nodes are not able to establish a direct key, they can still connect to other nodes with the help of intermediate nodes. Delgosha and Fekri [37] extended the Kim *et al.* scheme [64] by using the multivariate polynomials instead of the bivariate polynomial. In this scheme, a k-tuple unique identity is associated with each node. The base station generates a virtual k-dimensional

space that has k axis lines. A k-variate polynomial will be associated with each axis line. Each intersection of the k axis lines in this k-dimensional space is assigned to a node. Each node is given shares from k polynomials corresponding to k axis lines intersecting at a point assigned to that node. Each of these polynomials is evaluated at (k-1) variables determined by the k-tuple node identity. Thus, k univariate polynomial shares are stored in the node memory. For the direct key establishment, two neighboring nodes with node identities at the Hamming distance of one from each other evaluate the keys using their (k-1) common univariate polynomial shares. A symmetric combination of these common keys is the final link key between the two nodes.

In 2009, Liao *et al.* [74] proposed a robust grid-based key pre-distribution scheme which uses algebraic tame automorphism, which is a variant of the grid-based scheme given by Liu and Ning in [78]. In [74], N nodes in the network are deployed in a 2-dimensional field. Each node will be having a field co-ordinate, say(xc,ym). xc-coordinate denotes the cell number of the node and ym-coordinate can be considered as marking within the cell. A node is assigned shares from two symmetric tame-maps based on its field coordinates. Two nodes can establish a direct pair-wise key if they belong to the same cell(inter-cell) or have same marking in their respective cells(intra-cell). In other words, if the xc-coordinate or ym-coordinate of the nodes are matching, they are considered to be direct neighbors. For example, nodes with coordinates (1, 1) and (1, 2) are direct neighbors and so are the nodes (1, 1) and (2, 1).

Recently, Guo *et al.* [47] proposed a permutation-based multi-polynomial scheme for the pair-wise key establishment. The scheme assigns the shares of m unique symmetric bivariate polynomials $f_k(x, y)$, $1 \leq k \leq m$, to the nodes in a random order. A node u_i with its unique identity, Id_i, gets the polynomial shares $f_k(Id_i, y)$. Any two nodes u_i and u_j with unique identities Id_i and Id_j respectively and, in the communication range of each other can establish a direct pair-wise key as follows:

Node u_i computes: $K_{ij} = f_{k_1}(Id_i, Id_j) \oplus f_{k_2}(Id_i, Id_j) \oplus \ldots \oplus f_{k_m}(Id_i, Id_j)$

Node u_j computes: $K_{ji} = f_{k_1}(Id_j, Id_i) \oplus f_{k_2}(Id_j, Id_i) \oplus \ldots \oplus f_{k_m}(Id_j, Id_i)$

Here, subscript range k_1, k_2,....k_m is a unique permutation of integers 1, 2, ..., m for each node. The unique secret key between nodes u_i and u_j will be $K_{ij} = K_{ji}$, since $\oplus$ operation is commutative and $f_k(x, y)$ is a symmetric bivariate polynomial (for $k = 1, 2, \ldots, m$).

The scheme in [42] relies on probabilistic key sharing. A limited number of keys are randomly drawn from a finite pool of keys, so it is difficult to achieve high network connectivity, especially in the dynamic environment and large networks. Similarly, the probability of direct connectivity is low with the random distribution of polynomial shares proposed by Liu and Ning in [78]. With the two dimensional grid based approach [78], another instantiation of the Liu and Ning general framework, the probability of key establishment between a pair of nodes actually improves. However, in their proposal, at any time m nodes are sharing the same row/column polynomial. If that polynomial is disclosed, the direct keys shared by these m polynomials can easily be discovered, hence compromising the nodes is easier. Kim *et al.* [64] uses k-dimensional grid in place of two dimensional grid to enhance the probability of connectivity, however, their scheme is comparatively less resilient to node capture. Although, more alternative paths are available for key-establishment when direct paths are compromised. Liao *et al.* [74] use tame-maps and further enhances the connectivity by giving the possibility of inter-cell and intra-cell connectivity. However, all the above mentioned schemes are optimized for static arrangements of nodes, where nodes are not mobile after the deployment. Finally, Guo *et al.* scheme [47] provides a promising solution for ensuring connectivity even in mobile WSN with high resilience to node capture attack.

The polynomial share based schemes provide inherent authentication, as a node is assured that only a valid node in the network could possess the required polynomial share to establish a pair-wise key. However, the key once established remains static and the approach is thus vulnerable to known key attack and replay

attack. Moreover, the approach does not provide forward secrecy, as once a key is known, an attacker gets hold of all the previous communications encrypted using that single static key. Thus, we need a key update protocol that allows updating pair-wise key between two nodes after each session.

In [48], a compromise resilient pair-wise re-keying protocol is discussed to establish and update the pair-wise between a cluster head and any of its sensor nodes or between two cluster heads. In [48], for each session key, the sender sends a polynomial share that consists of shares from a symmetric bivariate polynomial and a perturbation polynomial. A node needs to compute three candidate keys and may have to perform at most three symmetric decryption to check which is the valid key for the session. The receiver has no way to validate the encrypted message as no message authentication is provided. Moreover, the protocol in [48] uses perturbation polynomial that was introduced by Zhang *et al.* in [137] claiming to increase the resilience threshold while maintaining efficiency by adding some noise to polynomial-based systems. However, Albrecht *et al.* in [7] have shown that the heuristic security arguments given for perturbation polynomial can be broken completely. The EDDK scheme (energy-efficient distributed deterministic key management scheme) proposed in [138] allows a pair of nodes to update the key wherein one node selects the new key and shares the new key with the pairing node by encrypting it with the old pair-wise key. However, if an attacker gets hold of the old key, it can get access to the new updated key as well.

Therefore, a secure and efficient pair-wise key establishment and key update protocol is needed that not only ensures high resilience to node capture attack, but also protects the network from known key and impersonation attacks and allows a pair of nodes to establish and update the pair-wise key with mutual authentication and mutual key control. We discuss the proposed key update protocol for dynamic WSN [4] in the next section.

3.3 Proposed Pair-Wise Key Establishment and Key Update Protocol

3.3.1 Goals and Assumptions

We present an authenticated pair-wise key establishment and key update protocol to ensure forward secrecy and resilience to replay, known key and node capture attacks in mobile WSNs ensuring mutual pair-wise key control.

The proposed protocol [4] assumes that WSN has a central trusted authority as the base station and a large number of nodes that are mobile in nature. The nodes in the network are allowed to change their locations during the network operation. Each node is pre-deployed with a finite set of polynomial shares calculated over its unique identity. Each node is capable of computing a univariate polynomial over a finite field. The nodes can establish a pair-wise key with any other node in the network and can directly communicate with a node in its communication range. Whenever a node changes its location, it indicates the move to all its current neighbors. Nodes at the new location again set up and maintain a secure communication channel within the new location. Node-to-node authentication is done only once, thus, during the course of the move, if a node happens to meet an old neighbor, they need not re-authenticate each other; they already have negotiated their pair-wise key. Such key can be refreshed using the session key update protocol.

We assume that the adversary is capable of eavesdropping on the communication channel to listen to the on-going communication. The pair-wise master secret computed using these polynomials shares is not communicated at any time during the protocol execution and, therefore, not available to an adversary through the communication channel. Moreover, it is assumed that the nonces (ephemeral secrets) used to generate the initial pair-wise key are not available to the adversary.

3.3.2 Set-up and Initialization

In the initialization phase, a unique identity is assigned to each node and, using the unique identity, shares from m distinct bivariate polynomials are given. Any bivariate polynomial scheme that provides high connectivity and high resilience to node capture attack can be used, such as Guo *et al.* scheme [47]. The base-station assigns a unique identity Id_i to each node $u_i \in U = \{u_1, u_2, \ldots, u_n\}$. The base station chooses a large prime q s.t. $n \leq q \leq 2^l$, where l is the number of bits needed for pair-wise master key. The base station then follows the Guo *et al.* scheme [47] and constructs m symmetric bivariate polynomials of the form $f(x, y) = \sum_{a,b=0}^{t} A_{ab} x^a y^b$ of degree t for both x and y. Then, it assigns to each node, shares from the m distinct polynomials $f_k(x, y)$ $(1 \leq k \leq m)$ computed over their respective unique identities. So, a node u_i gets m shares $f_{k_1}(Id_i, y)$, $f_{k_2}(Id_i, y)$, $\ldots$,$f_{k_m}(Id_i, y)$. Here, the subscript sequence $(k_1, k_2, \ldots, k_m)$ is some random permutation of $(1, 2, \ldots, m)$.

3.3.3 Node Discovery and Node Authentication

In the node discovery and authentication phase, a node broadcasts its identity and all the nodes in its communication range respond. The communicating nodes compute the master secret from the shares of polynomials. This master secret is used in this phase for node-to-node authentication. We use the Diffie-Hellman key exchange protocol [38] to derive the initial pair-wise key. Once the nodes are deployed, they begin to establish the network by discovering their neighbors. In order to confirm the neighboring node, the computation of pair-wise key is required by two nodes to mutually authenticate each other. The mutual authentication process is carried out only once between a pair of nodes. For subsequent communications, the session specific pair-wise key established between the nodes ensures the authenticity of the nodes. If a node u_i wants to discover the nodes in its current neighborhood, u_i executes the protocol for node discovery and mutual authentication. In this protocol, node u_i broadcasts its unique identity Id_i in plain to the neighboring nodes. A neighbor node, say u_j, in the communication range

of u_i that listens to u_i's broadcast can establish the pair-wise master secret with node u_i. Node u_j computes master secret between u_i and u_j as $K_{ji} = f_{k_1}(Id_j, Id_i) \oplus f_{k_2}(Id_j, Id_i) \oplus \ldots \oplus f_{k_m}(Id_j, Id_i)$ using its polynomial shares. Then it selects a random nonce $N_j \in Z_q^*$ and computes g^{N_j}, where g is the primitive root mod q for large prime q. Node u_j further computes PRF value of Id_j, and g^{N_j} using key K_{ji} i.e. $PRF_{K_{ji}}(Id_j, g^{N_j})$. It finally constructs the response message Id_j, g^{N_j}, $PRF_{K_{ji}}(Id_j, g^{N_j})$ and unicasts the same to node u_i. On receiving a response from node u_j, node u_i computes $K_{ij} = f_{k_1}(Id_i, Id_j) \oplus f_{k_2}(Id_i, Id_j) \oplus \ldots \oplus f_{k_m}(Id_i, Id_j)$ using its own polynomial shares. It then computes $PRF_{K_{ij}}(Id_j, g^{N_j})$. Since $K_{ij} = K_{ji}$ due to the symmetry of underlying bivariate polynomials, the computed PRF value should match the PRF value received from node u_j. If these PRF values match, then node u_i authenticates node u_j. Now, node u_i selects a random nonce $N_i \in Z_q^*$ and computes g^{N_i} and $PRF_{K_{ij}}(Id_i, g^{N_i}, g^{N_j})$. Node u_i sends the message Id_i, g^{N_i}, $PRF_{K_{ij}}(Id_i, g^{N_i}, g^{N_j})$ to u_j as a unicast. Node u_j verifies the received PRF value by computing the same at its end to ensure that the g^{N_j} was correctly received by u_i and the term g^{N_i} received from u_i is correct. On this verification, node u_j authenticates node u_i. Once the mutual authentication is confirmed, both the nodes compute initial pair-wise key between them at their respective ends. Node u_i has N_i and g^{N_j}, so it computes the first pair wise session key between u_i and u_j as $K_{ij}^0 = (g^{N_j})^{N_i} = g^{N_j * N_i} = g^{N_i * N_j}$. Similarly, node u_j having N_j and g^{N_i} computes $K_{ij}^0 = (g^{N_i})^{N_j} = g^{N_i * N_j}$. Nodes u_i and u_j add each other to their neighbor list along with the initial session key K_{ij}^0. From this point on, both nodes can use K_{ij}^0 to protect their communication channel.

The protocol is summarized in Figure 3.2:

1. $u_i \Rightarrow$ *: Id_i
2. $u_j \Rightarrow u_i$: $Id_j, g^{N_j}, PRF_{K_{ij}}\left(Id_j, g^{N_j}\right)$
3. $u_i \Rightarrow u_i$: $Id_i, g^{N_i}, PRF_{K_{ij}}\left(Id_i, g^{N_i}, g^{N_j}\right)$

Figure 3.2: Node Discovery and Authentication

3.3.4 Session Key Update

A pair of nodes use the pair-wise key from the previous session, secret nonce values and the master secret to generate a new pair-wise key using a cryptographically secured PRF.

Once two nodes u_i and u_j have mutually authenticated each other and have established an initial pair-wise key, they can choose to update the current session key at any time. In order to do so, both nodes first exchange the nonce values secretly. These newly selected nonce values, current session key and master secret K_{ij} between u_i and u_j are used to generate the next session key. To secretly transfer the nonce value, it is encrypted with the current session key. The PRF value of this newly selected nonce along with the identities of both the nodes is computed using the master secret K_{ij}. This not only ensures the secrecy of nonce en-route, but also the receiver can verify that the sender is the legitimate party with whom it shares the unique keys and the nonce value is not corrupted.

Suppose the nodes u_i and u_j are currently operating in session s. If nodes have some data to exchange, this protocol also allows nodes to exchange data (say M_i^s and M_j^s respectively by nodes u_i and u_j in session s) along with key update parameters during the process to save on communication and computation cost, although this is optional. In order to now update the session key, let node u_i initiates the key update protocol. Node u_i randomly selects a nonce $N_i^s \in Z_q^*$. It encrypts the selected nonce N_i^s and message M_i^s with the current session key K_{ij} as $E_{K_{ij}^s}(M_i^s, N_i^s)$. Node u_i then computes the PRF value for Id_i, Id_j, N_i^s and M_i^s using the master secret K_{ij} i.e. $PRF_{K_{ij}}(Id_i, Id_j, N_i^s, M_i^s)$. It sends the message Id_i, Id_j, $E_{K_{ij}^s}(M_i^s, N_i^s)$, $PRF_{K_{ij}}(Id_i, Id_j, N_i^s, M_i^s)$ to node u_j via unicast. On receiving this message from u_i, node u_j uses the current session key K_{ij}^s and obtains M_i^s and N_i^s by performing decryption $D_{K_{ij}^s}(E_{K_{ij}^s}(M_i^s, N_i^s))$. Now, node u_j computes PRF over Id_i, Id_j, N_i^s and M_i^s using master secret K_{ij}. If the computed PRF value matches with the PRF value received from u_i then node u_j accepts the nonce value N_i^s and the data M_i^s. Node u_j now selects a random nonce $N_j^s \in Z_q^*$, picks the current data, say M_j^s, that it wishes to share with node u_i and encrypts N_j^s and M_j^s using the cur-

rent session key K_{ij}^s as $E_{K_{ij}^s}(M_j^s, N_i^s)$. It further computes the PRF of Id_i, Id_j, N_i^s, N_j^s and M_j^s using the master secret K_{ij} as $PRF_{K_{ij}}(Id_i, Id_j, N_i^s, N_j^s, M_j^s)$ and responds to node u_i with message Id_j, Id_i, $E_{K_{ij}^s}(M_j^s, N_j^s)$, $PRF_{K_{ij}}(Id_i, Id_j, N_i^s, N_j^s, M_j^s)$. Node u_i, when receives the response from node u_j, decrypts M_j^s and N_j^s with $D_{K_{ij}^s}(E_{K_{ij}^s}(M_j^s, N_j^s))$. It computes $PRF_{K_{ij}}(Id_i, Id_j, N_i^s, N_j^s, M_j^s)$ and verifies the same with the received PRF value. On successful verification, u_i accepts the data M_j^s and nonce N_j^s from u_j. Both the nodes compute the next session key $K_{ij}^{s+1} = PRF_{K_{ij}^s}(N_i^s, N_j^s, K_{ij})$ at their respective ends. Node u_i now constructs the key confirmation message as Id_i, $E_{K_{ij}^{(s+1)}}(Id_j, N_j^s - 1)$ and unicasts this message to node u_j that verifies the same by performing decryption $D_{K_{ij}^{(s+1)}}(E_{K_{ij}^{(s+1)}}(Id_j^s, N_j^s\text{-}1))$ to ensure that node u_i has indeed computed the same key for next session. Both nodes delete the nonces used in this process and the previous session key. As a result of this protocol, a fresh pair-wise session key will be established using the previous pairwise key as K_{ij}^{s+1}. Since the nonce values from both the participating nodes are used, the new key generated will not be biased by any communicating node. In the first key update, $K_{ij}^s = K_{ij}^0$. The summary of the s^{th}-iteration of the protocol is given in Figure 3.3:

1. $u_i \Rightarrow u_j$: Id_i, Id_j, $\{M_i, N_i^s\}_{K_{ij}^s}$, $PRF_{K_{ij}}(Id_i, Id_j, N_i^s)$
2. $u_j \Rightarrow u_j$: Id_j, Id_i, $\{M_j, N_j^s\}_{K_{ij}^s}$, $PRF_{K_{ij}}(Id_i, Id_j, N_i^s, N_j^s)$
3. $K_{ij}^{s+1} = PRF_{K_{ij}^s}(N_i^s, N_j^s, K_{ij})$
4. $u_i \Rightarrow u_j$: Id_i, $\{Id_j, N_j^s - 1\}_{K_{ij}^{s+1}}$

Figure 3.3: Session Key Update

3.3.5 Security Features

The proposed protocol provides *forward secrecy* and resists *known key*, *impersonation* and *replay attacks*. The protocol achieves *node-to-node authentication, data confidentiality and mutual key control* in the mobile sensor network. As the nonces of both participating nodes are used in updating the pair-wise key between them, the resulting key can not be biased by any single node and therefore, the protocol

ensures ***mutual key control***. For node-to-node authentication, both the participating nodes compute the master secret at their respective ends, using their common polynomial shares. The key is updated using previous session's key, nonces and the master secret. The previous session key and the nonce values are destroyed once the new key is computed. The protocol also ensures ***forward secrecy*** where the adversary is not able to derive the previous sessions' keys even if it gets hold of the long-term master secret.

We carried out the formal analysis for proving the authentication and confidentiality features of the proposed protocol using ProVerif tool. The script and the execution results are given in *Appendix 1: ProVerif Tool*. Since ProVerif tool considers an attacker with Dolev-Yao [40] capabilities, we could not use ProVerif for analyzing all our security claims. To maintain the uniformity in the security analysis, we created the inference rules using the logic drawn form ProVerif analysis. We use these rules to prove the security claims in the form of theorem proving technique in our proposed protocol in this chapter as well as in all the subsequent chapters. The notations used in the proof of a theorem are as given below:

$Attacker(S)$	- Attacker has access to S (S may be some term/name/variable/channel)
L	- Public channel
$W \wedge V$	- Logical AND operation on W and V
$W \vee V$	- Logical OR operation on W and V

We give the attacker's knowledge at various states, before and during the execution of the protocol, to reach the conclusion. The attacker's knowledge at initial state is as follows:

Initial State

In the initial state, through network topology, the attacker is aware of the unique identities of nodes. The public functions and public channel L are accessible to the attacker. Moreover, the attacker does not have access to the polynomial shares held by the valid nodes. Therefore:

Attacker(Id_i)	... (3.1.1)
Attacker(Id_j)	... (3.1.2)
Attacker(PRF())	... (3.1.3)
Attacker(L)	... (3.1.4)
Not Attacker($f_l(Id_i, y)$)	... (3.1.5)
Not Attacker($f_l(Id_j, y)$) (for $0 \leq l \leq m$)	... (3.1.6)

3.3.5.1 Forward Secrecy

Forward secrecy ensures that even when the long term secret is revealed, the previous session keys remain inaccessible.

Theorem 3.1. *During a session s, given that an attacker gets hold of the long term master secret K_{ij}, the protocol ensures that the attacker does not get access to previous session keys K_{ij}^l ($0 \leq l < s$).*

Proof. When the session s is being executed, the nodes have the long term master secret K_{ij}, the current nonces N_i^s, N_j^s and the current session key K_{ij}^s. The nodes have deleted the previous session key and old nonces. Therefore:

Not Attacker(N_i^l)		... (3.2.1)
Not Attacker(N_j^l)		... (3.2.2)
Not Attacker(K_{ij}^l)	($0 \leq l < s$)	... (3.2.3)
Attacker(K_{ij})	(Attacker gets hold of long term master secret)	... (3.2.4)

Now, we suppose that the protocol provides no forward secrecy and the attacker gains access to a previous session key K_{ij}^{s-1} using long term master secret.

Attacker(K_{ij}^{s-1})
$\Rightarrow$ Attacker($PRF_{K_{ij}^{s-2}}(N_i^{s-2}, N_j^{s-2}, K_{ij})$)
[from step 3 of session key update protocol (Figure 3.3)]
$\Rightarrow$ Attacker(N_i^{s-2}) $\wedge$ Attacker(N_j^{s-2})
$\wedge$ Attacker(K_{ij}) $\wedge$ Attacker(PRF()) $\wedge$ Attacker(K_{ij}^{s-2})
$\Rightarrow$ FALSE $\wedge$ FALSE $\wedge$ TRUE $\wedge$ TRUE $\wedge$ FALSE
[using assertions 3.2.1, 3.2.2, 3.2.4, 3.1.3 and 3.2.3 respectively]
$\Rightarrow$ FALSE

This is a contradiction. Thus, the assumption that the attacker gets access to K_{ij}^{l} ($0 \leq l < s$) after knowing long term master secret K_{ij} in session s is false.
Here, the session key is computed using a one way PRF with the help of the previous session key, nonce values and master secret (which is the long term secret). Once the nodes compute the new session key, all the ephemeral parameters i.e. nonce values and previous session keys are destroyed. At any given time, if the adversary gets hold of the long term secret, due to one-way property of the PRF, it can not compute key of any previous session. □

3.3.5.2 Resistance to Impersonation Attack

In an *impersonation attack*, the attacker assumes the identity of a legitimate node in the network and sends the messages on behalf of the node being impersonated or receive messages directed to the node it fakes.

Theorem 3.2. *The protocol resists impersonation attack by not allowing a fake node to assume the identity of a valid node in the network.*

Proof. Suppose a fake node u_x pretends to be a valid node u_i. In order to impersonate u_i, the node u_x needs to carry out mutual authentication with any node in its neighborhood. Suppose the node u_x successfully authenticates itself as node u_i to a neighbor node u_j. Therefore:

Attacker(Auth)
$\Rightarrow$ AttackerC($PRF_{K_{ij}}(Id_i, g^{N_i}, g^{N_j})$)
[Attacker can compute $PRF_{K_{ij}}(Id_i, g^{N_i}, g^{N_j})$]
$\Rightarrow$ Attacker(PRF()) $\wedge$ Attacker(Id_i) $\wedge$ Attacker(g^{N_i}) $\wedge$
Attacker(g^{N_j}) $\wedge$ Attacker(K_{ij})
[from step 3 of node discovery and authentication protocol (Figure 3.3)]
$\Rightarrow$ TRUE $\wedge$ TRUE $\wedge$ Attacker(g^{N_i}) $\wedge$ Attacker(g^{N_j}) $\wedge$ Attacker(g^{N_j})
$\wedge$ Attacker(K_{ij}) [using assertions 3.1.3 and 3.1.1 respectively]
$\Rightarrow$ Attacker(g^{N_i}) $\wedge$ Attacker(g^{N_j}) $\wedge$ Attacker(K_{ij})
$\Rightarrow$ Attacker(L) $\wedge$ Attacker(K_{ij})
[from steps 2 and 3 of node discovery and authentication protocol

(Figure 3.2)]

$\Rightarrow$ TRUE $\wedge$ (Attacker($f_l(Id_i, y)$) $\vee$ Attacker($f_l(Id_j, y)$)

[using assertions 3.1.4 and by definition of K_{ij} respectively]

$\Rightarrow$ FALSE $\vee$ FALSE [using assertions 3.1.5 and 3.1.6]

$\Rightarrow$ FALSE

This contradicts to our assumption of a fake node being authenticated as a valid node. Since node identity is publicly known in the network, the fake node u_x broadcasts id_i, the identity of the node u_i. Node u_j, which listens to the broadcast, computes the master secret K_{ij} using the common polynomial shares and responds back with its own identity Id_j and the value g^{N_j}, N_j being the nonce selected by u_j. Node u_j also computes the checksum using PRF and sends it back to u_x (faking as u_i). The first two steps of the node authentication protocol (as given below) may be executed successfully:

1. $u_x \rightarrow$ *: Id_i
2. $u_j \rightarrow u_x$: $Id_j, g^{N_j}, PRF_{K_{ij}}(Id_j, g^{N_j})$

However, the third step in the protocol needs u_x to acknowledge back to u_j using the master secret K_{ij} as to compute PRF and after that validating the received checksum as:

3. $u_x \rightarrow u_j$: $Id_i, g^{N_i}, PRF_{K_{ij}}(Id_i, g^{N_i}, g^{N_j})$

Since u_x is pretending to be u_i and does not possess the polynomial shares required to construct K_{ij}, u_x will not be able to construct the acknowledgement message and the protocol will fail. Hence a fake node u_x cannot impersonate the valid node u_i during the initial node discovery and authentication process. When authentication fails, the fake node cannot participate in any further communication in the network. □

3.3.5.3 Resisting Known-key Attacks

The protocol also resists known key attacks when session keys are compromised [69]. In known-key impersonation attacks (KKI), an adversary uses the captured session keys to impersonate himself/herself as a valid entity of the network, with the aim of interfering with the session key negotiation process. Known-key pas-

sive attacks (KKP) are launched by adversary wherein he uses past session keys to compute the current session key. In known-key attacks without impersonation (KKA), an adversary (i.e. a malicious insider u_x) uses legitimate information (e.g. session keys, public information) to derive new or past session keys.

Theorem 3.3. *The protocol is secure against known key attacks.*

Proof. Let us assume that the attacker has captured a session key K_{ij}^s that is used between u_i and u_j for session s. Thus:

$$\text{Attacker}(K_{ij}^s) \quad \ldots (3.3.1)$$

Now, suppose the attacker has successfully launched known key attack and has computed the next session key K_{ij}^{s+1} as a valid node u_i or u_j, i.e.

$\Rightarrow$ Attacker(K_{ij}^{s+1})

$\Rightarrow$ Attacker(PRF()) $\wedge$ Attacker(K_{ij}^s) $\wedge$ Attacker(N_i^s) $\wedge$
Attacker(N_i^s) $\wedge$ Attacker(K_{ij})
[from step 3 of session key update protocol (Figure 3.3)]

$\Rightarrow$ TRUE $\wedge$ TRUE $\wedge$ TRUE $\wedge$ TRUE $\wedge$ Attacker(K_{ij})
[using assertion 3.1.3, 3.3.1 and,
from steps 1 and 2 of session key update protocol (Figure 3.3)]

$\Rightarrow$ Attacker(K_{ij})

$\Rightarrow$ (Attacker($f_l(Id_i, y)$) $\vee$ Attacker($f_l(Id_j, y)$) [by definition of K_{ij}]

$\Rightarrow$ FALSE $\vee$ FALSE [using assertions 3.1.5 and 3.1.6]

$\Rightarrow$ FALSE

The contradiction to our assumption proves that the known key attack cannot be launched since the protocol uses K_{ij} (the shared secret produced from the bivariate polynomial) as an authentication token. In other words, a session key by itself does not provide information about future session keys. □

3.3.5.4 Resistance to Replay Attacks

In the replay attack, an adversary intercepts the messages transmitted between a pair of nodes u_i and u_j in the network during the session s. After that, the

adversary replays the same message aiming to establish a new session with u_i or u_j.

Theorem 3.4. *The protocol prevents an adversary from launching replay attack.*

Proof. During the network operation, suppose an adversary listens to the communications that take place between nodes u_i and u_j.

Case 1: During node discovery and authentication phase:

Attacker(g^{N_j}) $\wedge$ Attacker($PRF_{K_{ij}}(Id_j, g^{N_j})$)	... (3.4.1)
[from step 2 of node discovery and authentication protocol (Figure 3.2)]	
Attacker(g^{N_i}) $\wedge$ Attacker($PRF_{K_{ij}}(Id_i, g^{N_i}, g^{N_j})$)	... (3.4.2)
[from step 3 of node discovery and authentication protocol (Figure 3.2)]	

Since the attacker has the knowledge of g^{N_j} and $PRF_{K_{ij}}(Id_j, g^{N_j})$ (as per assertions 3.4.1) required to replay the message, after some time, attacker replays the same message Id_j, g^{N_j}, $PRF_{K_{ij}}(Id_j, g^{N_j})$ (as in step 2 of node discovery and authentication protocol) to node u_i, pretending to be node u_j. However, as the discovery protocol is executed only once per node, node u_i would have already negotiated the initial pair-wise key K_{ij}^0 with node u_j, thus node u_i would consider the replayed message to be a duplicate and delayed message and will immediately discard it. Similarly, replay of message Id_i, g^{N_i}, $PRF_{K_{ij}}(Id_i, g^{N_i}, g^{N_j})$ with knowledge as per assertion 3.4.2 will be discarded by node u_j.

Case 2: Session key update phase:

Attacker($\{M_i, N_i^s\}_{K_{ij}^s}$) $\wedge$ $PRF_{K_{ij}}(Id_i, Id_j, N_i^s)$	... (3.4.3)
[from step 1 of session key update protocol (Figure 3.3)]	
Attacker($\{M_j, N_j^s\}_{K_{ij}^s}$) $\wedge$ $PRF_{K_{ij}}(Id_i, Id_j, N_i^s, N_j^s)$	... (3.4.4)
[from step 2 of session key update protocol (Figure 3.3)]	

Again, when the adversary pretends to be node u_i and replays the message Id_i, Id_j, $\{M_i, N_i^s\}_{K_{ij}^s}$, $PRF_{K_{ij}}(Id_i, Id_j, N_i^s)$ to node u_j (using knowledge as per assertion 3.4.3), u_j will not accept the message, as it has already received this message from u_i and has updated the session key to K_{ij}^{s+1} using newly shared nonce values. Since the delayed message replayed by the adversary is still encrypted with the previous session key K_{ij}^s, u_j will not be able to decrypt it correctly and discard the

replayed message. The message Id_i, Id_j, $\{M_j, N_j^s\}_{K_{ij}^s}$, $PRF_{K_{ij}}(Id_i, Id_j, N_i^s, N_j^s)$ (using knowledge as per assertion 3.4.4) replayed by adversary pretending to be node u_j will be discarded by node u_i in a similar manner. □

3.3.5.5 Resilience to Node Capture

The resilience to node capture attack of our protocol depends on the resilience of the underlying bivariate polynomial scheme. In our specific instantiation, we integrate the protocol by Guo *et al.* [47], which uses m polynomials - each of degree t. This protocol might seem to be only t-collusion resistant, but due to the permutation of the m polynomials, the actual probability of breaking all polynomials in one attempt (once t+1 nodes are subverted) is:

$(\frac{1}{m!})^{t+1}$ (cf. [47])

We analyzed the resilience to node capture with varying degree of polynomials and taking 5 polynomials. We consider the network size of 1000 nodes. The figure below shows that when the master key is generated using permutations of multiple bi-variate polynomial shares, the network is resilient to node capture attack even with the polynomial degree 10. With degree 10 polynomials, the attacker needs to capture 1200 nodes (beyond the network size).

From Figure 3.4, it is evident that the resilience to node capture is significantly higher when we use permuted multiple bivariate polynomial scheme for establishing the master secret, as compared to grid based schemes. With an underlying polynomial of degree 10 and number of polynomials as 5, while the 2-dimensional grid based scheme will collapse with capture of only 14 nodes, our proposed scheme would survive even with capture of more than 1000 nodes. As the degree of polynomial increases, the resilience to node capture increases exponentially with our proposed scheme as compared to 2-dimensional grid based scheme. In fact, this formula can be used by a network designer to discover the most optimal configurations for his/her network. For example, the following (m, t) combinations provide a security level of 128 bits (i.e. the effort of breaking this scheme t+1 nodes subverted by the adversary is the same as breaking a 128-bit symmetric key): $\{(16,2),\ (13,3),\ (10,5),\ (7,10),\ (5,18),\ (4,27)\}$. Networks with low prob-

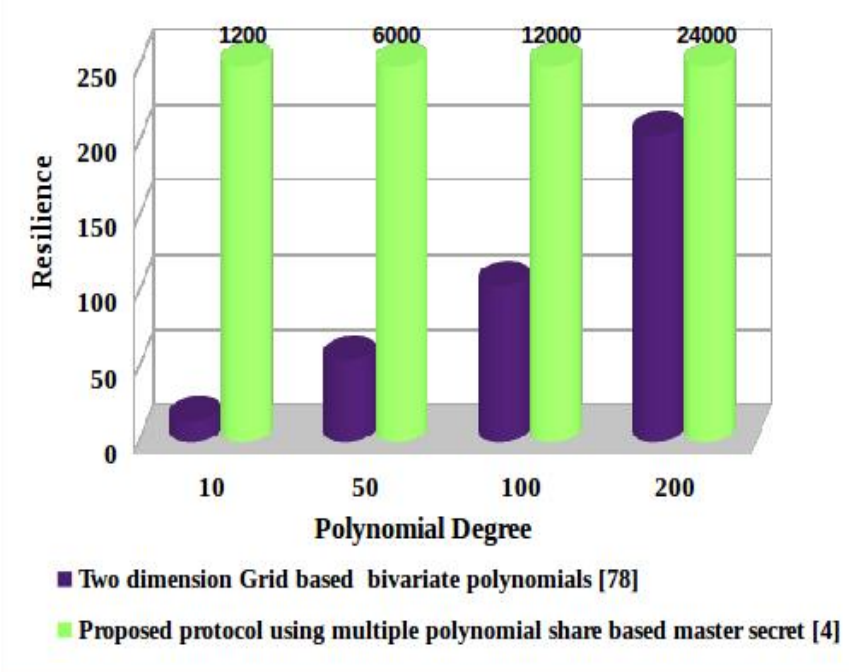


Figure 3.4: Resilience to Node Capture with Key Management

ability of node capture can use a smaller t value (less memory overhead), while networks in unattended and/or hostile environments can use a bigger t value (more memory overhead). Note that it is possible to substitute this underlying bivariate polynomial scheme with future ones, if their features are better.

Nevertheless, in order to provide a higher defense against node capture (regardless of the underlying bivariate polynomial scheme), we can also extend our protocol with the concept of the couple based protocol for early detection of node compromise [76]. To form a couple, when a node starts finding the neighbors, the first node it interacts with become its spouse. If the node is already coupled with some other node, the next node is searched for. Based on the common polynomial shares, the nodes compute the secret key. Once a couple (say node u_i and node u_j) is formed, the nodes start monitoring each other. Periodically, say after every T units of time, node u_i sends the following normal operation signal to node u_j: $u_i \rightarrow u_j$: $E_{K_{ij}^s}(Id_i, id_j, seq)$, where seq indicates the sequence number which is set to 0 for every new session. If node u_j receives this signal, it assumes the normal functioning of node u_i and vice-versa. However, if for three consecutive time periods, u_j neither receives this normal operation signal nor the message that u_i is moving

to some other location, u_i is suspected to be captured and other neighbors are informed of the same. When u_i moves to some other location, its spouse u_j looks for another partner u_x and makes couple with node u_x. We discuss the intricacies and the detection of node capture attack in detail in Chapter 5.

3.3.5.6 Resilience to Worm hole and Sink hole Attacks

After the initial deployment, the nodes have shared the keys and can communicate through encrypted messages using the pair-wise secret key. Since the network is mobile, we expect the nodes to move around. When a node u_i moves to a new location, it again broadcasts its identity to find neighbors at the new location. The adversary can not participate in the communication as it does not have the unique polynomial shares to compute pair-wise key and also the key is being updated after each node-pair communication. Therefore, wormhole or sinkhole attacks can not be mounted.

We show the security strength of the multiple polynomial share based key as compared to other polynomial based schemes in the Table 3.1.

Feature ⇒ / Schemes ⇓	Authentication	Forward Secrecy	Resilience to		
			Known key attack	Replay attack	Node Collusion (up to)
Single bivariate polynomial [17]	Yes	No	No	No	t
Random subset assignment with bivariate polynomial [78]	Yes	No	No	No	$((t+1)^*(s/s')-1)^*p$
2-dimensional grid with bivariate polynomial [78]	Yes	No	No	No	m - 1 + t
k-dimensional grid with bivariate polynomial [64]	Yes	No	No	No	(m/k) - 1 + t
Permutation with multiple bivariate polynomial [47]	Yes	No	No	No	$m! * t$
Proposed pair-wise key establishment and key update protocol [4]	Yes	Yes	Yes	Yes	$m! * t$

Table 3.1: Security Features of Polynomial based Key Management Schemes

3.3.6 Comparing Performance with Existing Protocols

The protocol performance would vary depending on the underlying scheme for generating long term pair-wise master secret. Table 3.2 gives the performance comparison of the proposed protocol with various polynomial based schemes.

Feature ⇒ / Scheme ⇓		Communication Cost (in bits)	Computation Cost	Storage Cost (in bits)
Single bivariate polynomial [17]		$\log q$	T_y	$(t{+}1)\log q + \log q$
Random subset assignment with bivariate polynomials [78]		$(s'{+}2)\log q$	T_y	$s' * (t{+}1)\log q{+}\log q$
2-dimensional grid with bivariate polynomials [78]		$2\log q$	T_y	$2(t{+}1)\log q{+} 2((t{+}1)\ log_2\ m)$
k-dimensional grid with bivariate polynomials [64]		$k\log q$	T_y	$k(t{+}1)\log q + k((t{+}1)\ log_2\ m)$
Permutation of multiple bivariate polynomials [47]		$\log q$	mT_y	$m(t{+}1)\log q + \log q$
Proposed pair-wise key establishment and key update protocol [4]	Initial key establishment	$4\log q$	$mT_y + 2T_x + 2T_p$	$m(t{+}1)\log q + \log q$
	Session key update	$6\log q$	$3T_p + 3T_s$	$\log q$

Table 3.2: Performance Comparison

We chose the multiple bivariate polynomial based approach due to its security strength. In Table 3.2, T_y and T_{dy} are timings to evaluate a single univariate polynomial and a polynomial with d-1 variables, respectively, T_x and T_p are the timings to compute exponent and PRF, respectively. Time T_s is to perform symmetric encryption/decryption. Size of the random key subset assigned to each node [78] is denoted by s' and m is number of distinct polynomials [47]). With the permuted multiple bivariate polynomial based master secret, the various overheads of the key update protocol are as follows:

Communication cost. In the node discovery and authentication, nodes need to find out the current neighbors. A node u_i broadcasts its identity. This needs log q bits. Now, a node u_j that responds to u_i with a message consisting of Id_j, g^{N_j},

PRF_j which needs 3 log q bits. The node u_i responds back to confirm the authentication with a message Id_i, g^{N_i}, PRF_i that also takes 3 log q bits. Therefore, for establishing one initial key, a node needs 4 log q bits. A node may be confirming to maximum d neighbors' responses to establish a pair-wise key with each. Thus, each node communicates maximum ($4d$ log q) bits during node discovery and authentication protocol to establish direct pair-wise key with d neighbors. Each time a node moves from its location, it needs to communicate the movement to its current neighbors and requires to again look for new neighbors repeating the same process. Thus it may require communicating further maximum (d + ($4d$ log q) bits.

Moreover, for a session key update, as initiating node, a node u_i communicates 6 log q bits and as a responding node u_j communicates 4 log q bits.

Storage cost Each node stores exactly m shares from m different t-degree polynomials. The node stores one master secret and one current session key for a pair-wise secret communication. Thus, the total storage requirement is $m(t+1)$ log q + log q.

Computation cost Initially, to discover and authenticate a node and establish the initial session key, a node u_i needs to compute m univariate polynomials (for pair-wise key), two PRF operations and two exponent operations. To update the session key, as initiating node, node u_i performs two encryptions, 3 PRF operations and one decryption. Similarly, as responding node, node u_j computes one encryption, 3 PRF operations and two decryption operations.

3.3.7 Experimental Results

For experimental purpose, we used ATmega328 processor using Arduino Duemilanove controller board and ArduinoISP programmer [8]. The protocols are programmed using Arduino IDE using 'C' language programming to evaluate the cost of computation of various operations in terms of time.

We evaluate the computation cost of our proposed protocol as given in Figure 3.5.

It shows that the one time initial cost of node discovery and authentication proto-

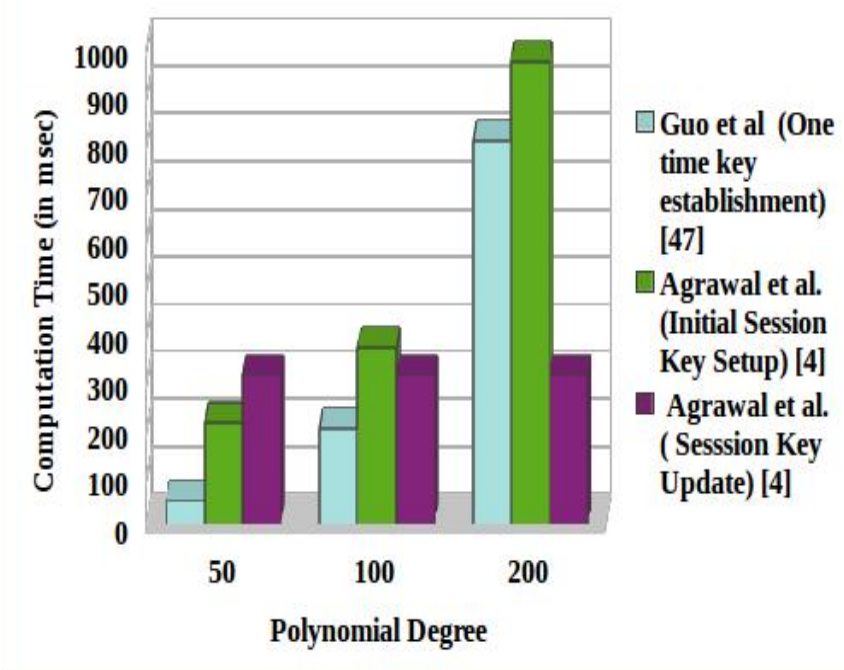


Figure 3.5: Computation Cost of the Key Update Protocol

col depends on the underlying scheme used for establishing the master secret and increases as we increase the degree of polynomials for increased resilience to node capture. However, the session key update protocol takes a constant time irrespective of the underlying polynomial scheme or degree of polynomials. Moreover, the cost of session key update is significantly less than the one time cost of node discovery and authentication.

To simulate the network performance, we have used *Castalia* simulator [20] specifically designed for low-power embedded devices such as wireless sensor networks. Castalia can be used for testing the distributed algorithms and/or protocols in realistic wireless channel and radio models. We have considered a uniformly distributed clustered network set up wherein some nodes are designated as cluster heads and the remaining nodes act as normal sensing nodes. The communication takes place at one-hop distance determined by the radio transmission range. The simulation parameters used are given in Table 3.3. The radio model used is CC2420 that is 2.4 GHz IEEE 802.15.4 compliant RF transceiver designed for low power and low voltage wireless applications that provides an effective

data rate of 250 kbps. The simulation is carried out using TMAC protocol that reduces the idle listening time. Periodically, each node wakes up and communicates with its neighbors, and then goes to sleep again until the next frame. In

Parameter	Value
Simulation Time	100s
Node Transmission output power	0 dBm
Cluster Head Transmission output power	5 dBm
Clear Channel Assessment (CCA) Threshold	-95 dBm
Field size	100 X 100 m^2
Radio Model	CC2420
Radio Range	25 meters
Node Deployment	Uniform
MAC protocol	TMAC

Table 3.3: Simulation Parameters

Figure 3.6, we present results of simulation for node energy consumption with different number of nodes in the network for one time initial key establishment and session key update. The simulation result shows the energy requirement for session key update is significantly low.

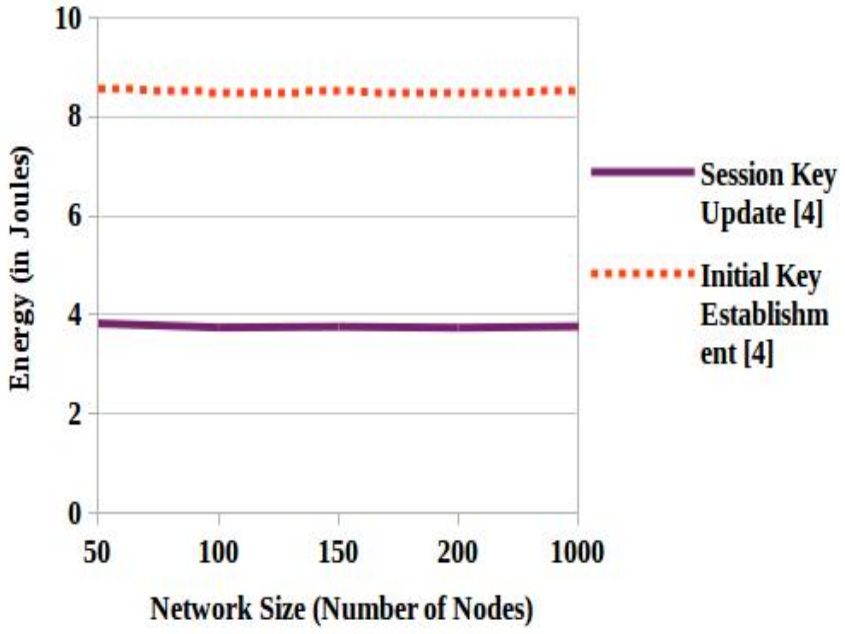


Figure 3.6: Node Energy Consumption

The proposed protocol updates the pair-wise key after each session using the master secret, previous session key and the nonces of both participating nodes at a nominal overhead of extra storage of a key and 3 communications per session which trades off for the additional security features such as forward secrecy and resilience to known key attack. All other protocols mentioned only use one pair-wise key during the whole lifetime of the nodes. In our scheme, we use the previous session key and the master secret to derive new session key. Since polynomial is t-degree, at least t+1 shares are needed to re-construct the polynomial to derive pair-wise key of some other pair of nodes. The actual resilience of the scheme once t+1 shares are extracted by an adversary is actually higher. Moreover, the proposed scheme can update the pair-wise key used in the communication channel. In other schemes, the key remains same for each session between two nodes in any pair. So, if the pair-wise key is compromised, all the past and previous communications would be revealed.

3.4 Conclusion

In this chapter, we discussed an authenticated pair-wise key establishment and update protocol for WSN in dynamic environments. A multi-polynomial based scheme is used to establish a master secret key between a pair of nodes. With the help of this master secret, the pair of nodes can generate a fresh pair-wise key between themselves. The protocol does not involve base-station or any additional communication overhead for node-to-node authentication; in the node discovery phase itself the nodes are authenticated and initial key establishment takes place. The protocol is secure against impersonation, replay, worm-hole and sink-hole attacks. Known key attacks are prevented as the protocol implements a key update mechanism that allows two nodes to negotiate a new pair-wise key any time during the communication. The protocol provides data confidentiality through a dynamic secret key and ensures forward secrecy and mutual key control. Even if the underlying multi-polynomial scheme provides a very good resilience, this protocol can make use of any present and future polynomial scheme that imple-

ments better features.

The computation cost of session key update is constant irrespective of the degree of polynomials used for setting up the master secret between a pair of nodes. When compared with the one time initial session key establishment with high degree of polynomial (≥ 200), it is observed that the session key update cost is significantly low. Irrespective of the network size, the energy consumption of a node for session key update is less than half the energy consumed during initial key establishment.

The pair-wise keys are established at the upper level in our solution framework between base station and a cluster head or between a pair of cluster head. Since sensor nodes work in collaboration to perform an application specific task, at sensor node level, a shared group key is proposed for a set of nodes managed by a cluster head. In the next chapter, we present a self-healing and mutual-healing enabled group key distribution protocol.

CHAPTER 4

Self-Healing and Mutual-Healing enabled Group Key Distribution

For secure group communication in a WSN, a central authority may distribute some keying material containing secret to a subset of nodes. However, a possibility exists wherein a node may miss out on one or more broadcasts. Self-healing is incorporated to get such missing information. Using the recent broadcast keying material, a node can extract the key used in some previous session with self-healing, if that node holds the membership in that session for which it is attempting to extract the key. Furthermore, if the node misses out the recent broadcast, it can seek help of its neighboring nodes to obtain the missing key through mutual healing. In this chapter, we discuss the proposed low cost self-healing and mutual-healing enabled group key distribution using bilinear pairing and then present a Chinese remainder theorem based group key distribution that provides self-healing and mutual-healing along with the required security features at significantly low cost overhead.

4.1 Introduction

Group-key broadcast is a commonly used approach to share a common secret to a group of nodes by a central authority [104]. However, the group key broadcast does not serve the purpose in case a node does not receive the broadcast message. Re-broadcasting of the messages is not a cost-effective solution in a WSN environment with resource constrained sensor nodes. This issue got researchers'

attention and the self-healing was proposed as a way to help a node obtain the missed broadcast without involving the distribution authority.

In self-healing, a node who had missed some broadcast can retrieve the key shared in that broadcast using a subsequent broadcast. Thus, the node does not require to send an explicit request to the distributing authority for getting the missed broadcast. For a resource constrained node, saving on the communication cost for an explicit request to distributing authority is significant. However, there is a possibility of a node missing multiple consecutive broadcasts and can not afford to wait for future broadcasts. Also, the node may require to have the current broadcast in the current session itself without requesting the same from the distributing authority. In such cases, mutual healing comes to rescue. With mutual healing, a node may request its immediate neighbors to share the missing broadcast. A neighbor node, having received the requested broadcast, may respond after ensuring the mutual authentication with the requesting node. Suppose, we consider a scenario of battle-field surveillance, where the nodes are deployed to observe the enemy intrusion across the border. The nodes are left unattended after deployment and a central authority is a mobile entity that periodically collects the data from the sensor nodes [102] and therefore, not always available to collect the data. The nodes may need to keep the measured or observed data for some length of time. In this time period, an attacker may compromise a node and read its data. The data collected before and after the compromise may be protected, if the key distribution mechanism provides forward and backward secrecy and some secure and effective way of handling membership changes and node addition and revocation is in place. A healing enabled group-key distribution is therefore needed.

4.2 Self Healing

Staddon *et al.* introduced the idea of self-healing group key distribution in [114] that addresses secure group communication in unreliable networks deployed for applications such as military surveillance. The group key distribution scheme in

[114] uses two dimensional polynomial based secret sharing that enables group members to recover lost session keys if a group member receives both the broadcasts sandwiching the missing broadcast. An enhanced and simpler version of [114] was proposed by Blundo *et al.* [16]. Both these protocols are based on exponential arithmetic and demand expensive maintenance costs and high computation costs. In [79], Liu *et al.* proposed an improvement over [114] with respect to communication complexity and memory storage. Liu *et al.* in [79] deals with coalition of more evicted group members as well.

Yuan *et al.* [134] used a security model that allows participation and revocation of at most t nodes and, resilience to collusion of at most $2t$ nodes, t being the degree of underlying polynomial. The protocol [134] incurs less communication and storage overhead compared to the protocols in [114] and [79]. Hong and Kang protocol [53] incurs lesser communication overhead and gives t-collusion resistance capability by allowing a user to recover keys of all the sessions (for which the user holds membership) from a single broadcast message. In [133], Yuan *et al.* proposed self healing with limited group membership wherein a fixed number of (t-1) users are allowed. By maintaining t users throughout the communication, the bandwidth is saved to get enhanced quality of service. During the lifetime of the secure group communication, the protocol in [133] allows revoking any number of members thus eliminating the limitations of revoking at most t members. The protocol would require about half the broadcast message size but twice the memory space as compared to [53].

To overcome some shortcomings in the existing schemes and yet keeping their advantages, Zou et al [145] introduced the concept of access polynomial. An access polynomial is defined as $A_j(x) = (x - VID) \prod_{i=1}^{d}(x - ID_i)$+1 using the unique identity ID_i (1 $\leq$ i $\leq$ d) of each of the d active users. The term $(x - VID)$ ($VID \neq ID_i$) is virtual and randomly selected for each access polynomial. The aim of $(x - VID)$ is to differentiate access polynomials even when they are computed for same active users. A user can evaluate an access polynomial $A_j(x)$ using its identity as $A_j(ID_i)$. For an active user, $A_j(ID_i)$ results in 1, while for a non-active user, it results in a random value. In the access polynomial based self-healing protocol

[145], the group manager (GM) randomly selects t-degree polynomial $p(x)$ and secretly distributes the unique identity $ID_i \in F_q$ and the polynomial share $p(ID_i)$ to each user $u_i \in U$. The GM chooses the secret K_j for session j and a random polynomial $S_j(x)$ of degree t. It splits the key K_j such that $K_j = S_j(ID_i) + T_j(ID_i)$ for each active user u_i. GM then broadcasts two polynomials $P_j(x) = A_j(x) * S_j(x) - p(x)$ and $Q_j(x) = T_j(x) - p(x)$, where $A_j(x)$ is the access polynomial computed for session j. Upon receiving the polynomials $P_j(x)$ and $Q_j(x)$, an active user u_i can retrieve the session key as:

$$K_j = ((P_j(ID_i) - p(ID_i))/A_j(ID_i)) + (Q_j(ID_i) - p_j(ID_i)) = S_j(ID_i) + T_j(ID_i).$$

Since, $A_j(ID_i) = 1$ for active users and some random value for non-active users, only active users would be able to get correct value of $S_j(ID_i)$ and therefore, retrieve the valid session key. Subsequently, Tian *et al.* [119] used access polynomial for self-healing to reduce computation and communication overhead. In [119], rather than splitting the key into two polynomial shares, a single polynomial $P_j(x)) = A_j(x) * K_j + p_j(x)$ is prepared by GM for a session j. The GM broadcasts this polynomial to all users. An active user can retrieve the valid key by computing $((P_j(ID_i) - p(ID_i))/A_j(ID_i))$, since $A_j(ID_i) = 1$ for active users. Wang [125] observed that in [119] forward secrecy is not provided and revocation is also limited to the degree t of the polynomial used. Dutta [41] also proposed access polynomial based protocol but it does not perform selective key distribution and thus gives the same effect as a simple broadcast of session key.

A bilinear pairing [66] based self-healing key distribution protocol is proposed by Tian *et al.* in [121] that allows a user to verify the integrity and correctness of the cipher text before carrying out key recovery operations. In [121], a subset of users form a communication group for each session and the group manager(GM) broadcasts keying material in each session. Only an authorized session group user is allowed to extract the session key of current and any previous session. The bilinear pairing based self-healing protocol in [121] assumes the existence of a key generation center (KGC) that takes two cyclic groups G_1 (additive) and G_2 (multiplicative) defined over q and a bilinear mapping e: $G_1 \times G_1 \rightarrow G_2$. The KGC randomly selects a generator $P \in G_1$ and defines two cryptographic hash func-

tions H_1: $\{0,1\}^* \rightarrow G_1$ and H_2: $G_2 \rightarrow \{0,1\}^*$. KGC also selects its private key as a random number $s \in Z_q^*$ and sets its own public key as $P_{Pub} = sP$. For each node u_i in the network, its public key is set as $Q_{ID_i} = H_1(ID_i)$ using its unique identity ID_i. KGC computes $S_{ID_i} = s.Q_{ID_i}$ as private key for the node u_i. The public-private key pair is given to each node through some secure channel. For a communication group of size $|CG_j|$ for a session j, a group manager (GM) constructs a broadcast message. GM first defines a $|CG_{j-1}| \times |CG_j|$ matrix $\{a_2, a_3, \ldots, a_{|CG_j|}\}$ with each vector a_i (i = 2, 3, ..., $|CG_{j-1}|$) of size $|CG_j|$. GM randomly chooses $r_j \in Z_q^*$ and formulates the broadcast message B_j in the following manner:

$U_1 = r_j$.P

$U_i = r_j.Q_{v_i}$ ($2 \leq \mathrm{i} \leq |G_j|$)

$V_j = K_j \oplus H_2(e(P_{Pub}, r_j.Q_{v_1}))$

$z_j = (U_i\ ((1 \leq \mathrm{i} \leq |G_j|), V_j)$

$B_j = (z_1, z_2, \ldots, z_j)$

GM broadcasts this message B_j to all the nodes in the set G_j. An authorized node u_x recovers the session key $K_j = V_j \oplus H_2(e(U_1, x_1.S_x).e(P_{Pub}, \sum_{i=2}^{|G_j|} x_i.U_i)$ from the broadcast message.

A node has to obtain the vector $(x_1, x_2, \ldots, x_{|G_j|})$ by solving a system of linear equations. With the bilinear pairing based self-healing in [121], any collusion of non-authorized users does not reveal a session key. The protocol also provides a constant storage overhead to each user and is not affected by whatever number of other users are revoked unless the private key of any user is disclosed/ compromised. However, the generation of broadcast message and key recovery process are computationally heavy. A member node requires to perform a matrix inversion operation to obtain the vector $(x_1, x_2, \ldots, x_{|G_j|})$ and need to perform $|CG_j|$ scalar multiplications in order to recover the key. Each node stores two vectors of size $|CG_j|$ (the group size in session j) and the public keys of all the nodes in its group, resulting in high storage overhead.

To summarize, although polynomial based algorithms [41] [134] [133] are efficient and simple, some information about pre-distributed user data is disclosed and such data can not be re-used. This weakness is not shown in exponent based

algorithms [114] [16], however, as compared to polynomial based protocols, exponent based algorithms are computationally heavy and do not provide backward secrecy. The most efficient one-way hash chain based protocols fail to provide collusion-resistance property. All the three security features namely, forward secrecy, backward secrecy and collision resistance required in self-healing protocols, are provided by bilinear pairing based protocol [121] that demands high computation cost due to bilinear pairing calculations. In [105], self-healing group-key distribution with extended revocation capability is proposed by Rams and Pacyna. The protocol in [105] requires the nodes to evaluate polynomial for recovering the update polynomial broadcasted by GM and perform exponential computation for session key computation. To reduce communication overhead in the self-healing protocol given by Tian *et al.* [121], recently, Hassan *et al.* [96] proposed hash chain based approach. However, with [96], whenever a new user joins in a session j, the seed value is changed. An existing valid user that had lost the session key broadcast for session j-1 does not have any means to retrieve that key from the broadcast received in session j. Mutual-healing is inevitable in such case.

4.3 Mutual Healing

The concept of mutual healing in wireless sensor networks was discussed by Tian *et al.* in 2011 [120]. They proposed a group-key broadcast protocol using bilinear pairing that allows a node to recover a missed key by taking help of a neighbor node in the same session. Mutual healing is assumed only between one-hop neighbors. Using some range-based secure localization process such as [71], for each node u_x, a location based secret LK_x is set up with the help of mobile robots. This secret LK_x is used to establish a pair-wise key between the two nodes involved in the mutual healing process. When a node u_i misses a broadcast message in some session t, it locally broadcasts an authentication request message with its own identity, location and the session number t of the requested broadcast message B_t, all in plain:

$u_i \rightarrow *$: ID_i, l_i, t

A neighboring node u_j that receives such request, verifies the claimed location l_i to be within its one-hop communication range and if so, calculates a shared key $K_{ji} = e(LK_j, \mathrm{h}(ID_i \| l_i))$ and sends a uni-cast response to u_i's request:

$u_j \rightarrow u_i$: ID_j, l_j, $E_{K_{ij}}(B_t)$

On receiving response, u_i performs location verification of neighbor u_j and then u_i calculates the key $K_{ij} = e(LK_i, \mathrm{h}(ID_j \| l_j))$ shared with u_j and extracts the broadcast message (B_t). During the mutual healing in [120], for request and response communication, the node location is communicated in plain, which may give way to known-location attacks. As there is no way to ensure the integrity of the location provided by the requesting node, an adversary may update the location or session number shared by the requesting node, resulting in location verification failure. While the protocol in [120] suggests sending the broadcast message in plain during normal course of operation, it computes a shared key using bilinear pairing for encrypting the broadcast message only to authenticate the response from the neighbor node. This is a computation overhead both for requesting and responding nodes.

4.4 Proposed Bilinear Pairing based Healing Protocol

4.4.1 Goals and Assumptions

We present a group key distribution protocol that provides self-healing and mutual-healing capability along with authentication and resistance to impersonation and replay attacks with significantly low overhead as compared to the existing protocols.

We assume that nodes may leave a group or join a group or may change the group. Due to this, for each session, the group manager may have a different set of authorized nodes in its group. The presence of an active adversary is assumed, who can intercept messages on the communication channel and is capable of injecting, updating or deleting the messages in the traffic.

4.4.2 Session Key Management

The proposed protocol consists of three phases. We first give the system set up and then present the step-wise description of the proposed protocol.

4.4.2.1 System Set-up

All the group managers and nodes in the networks are assigned unique identities prior to deployment by the base station. The base station generates two groups G_1, G_2 and a bilinear pairing e: G_1 X $G_1 \rightarrow G_2$ and, arbitrarily selects generator $P \in G_1$. It defines two cryptographic hash functions H_1: $\{0,1\}^* \rightarrow G_1$ and H_2: $G_2 \rightarrow \{0,1\}^n$. For public-private key setup, a group manager (GM) selects a random number $s \in Z_q^*$, which is known only to GM and serves as its private key. GM sets its own public key as $P_{Pub} = s.P$ (denotes scalar multiplication of integer s with the elliptic curve point P). The system parameters consist of $params$ = (G_1, G_2, q, P, P_{Pub}, H_1, H_2). Each node in the network submits its identity to GM, using which, GM computes node's public key $Q_{ID} = H_1(ID)$ and $S_{ID} = s.Q_{ID}$ as private key for the node and sends it to the node securely. The public keys are used by GM in the construction of broadcast message (as explained in subsequent paragraph). GM selects independent session keys K_1, K_2, ..., K_m for m sessions using a uniform distribution.

4.4.2.2 Group Key Broadcast

For a session j, GM randomly chooses session key $K_j \in Z_q{}^*$ that is used for communication within the group CG_j of size $d = |CG_j|$ for session j. It adds up the public key values Q_{ID_i} for all nodes $u_i \in CG_j$ to get a term Q. GM chooses a random number $r_j \in Z_q^*$. The chosen session key K_j, random number r_j, the term Q and the public key of the GM P_{Pub}, all these values are used to construct the broadcast message B_j. GM generates the signature $Sign_j$ on B_j using Zhang's short signature scheme [135]. The broadcast message B_j and the signature $Sign_j$ are then broadcasted. The step wise process of group key broadcast is as given below:

1. $Q = Q_1 + Q_2 + \ldots + Q_d$
2. $U_0 = r_j.P_{Pub}$
3. $U_i = r_j(Q - s.Q_i)$, for $i = 1$ to d, $u_i \in CG_j$
4. $V_j = K_j \oplus H_2(e(P_{Pub}, r_j.Q))$
5. $z_j = (U_0, U_i$ (for $i = 1$ to d, $u_i \in CG_j$), $V_j)$
6. $B_j = (z_1, z_2, \ldots, z_j)$
7. $Sign_j = \frac{1}{h(B_j)+s}.P$
8. GM $\Rightarrow$ *: B_j, $Sign_j$

4.4.2.3 Authentication and Key Extraction

Upon receiving the broadcast message B_j, a node u_x authenticates the broadcast message by verifying the short signature received as $Sign_j$. If the signature verification fails, the broadcast message is discarded. Otherwise, the node can use its private key S_x for extracting the required session key. The broadcast message is constructed in such a manner that an unauthorized user $u_y \notin CG_j$ can not extract the session key. The steps below give the authentication and key extraction process:

1. If $e(h(B_j)P + P_{Pub}, Sign_j) \neq e(P, P)$ then
   Discard B_j, Exit.
2. Compute the term $e(P_{Pub}, r_j.Q)$ as follows:
   $e(P_{Pub}, U_x).e(U_0, S_x)$
   $= e(P_{Pub}, U_x).e(r_j P_{Pub}, s.Q_x)$
   $= e(P_{Pub}, U_x).e(P_{Pub}, r_j.s.Q_x)$
   $= e(P_{Pub}, U_x + r_j.s.Q_x)$
   $= e(P_{Pub}, r_j.Q)$
3. Extracts session key $K_j = V_j \oplus H_2(e(P_{Pub}, r_j.Q))$.

In the authentication process, for a correct message B_j, the verification succeeds, because:

$e(h(B_j)P + P_{Pub}, Sign_j)$
$= e(h(B_j)P + s.P, (h(B_j)P + s)^{-1}P)$
$= e((h(B_j) + s).P, (h(B_j)P + s)^{-1}P)$

$$= e(P, P)^{(h(B_j)+s).(h(B_j)+s)^{-1}}$$
$$= e(P, P)$$

Here (Q_x, S_x) is the public-private key pair of node u_x and P_{Pub} is the public key of GM.

4.4.3 Healing

In the process of **self-healing**, if a node u_x misses the broadcast message B_i from session $i < j$ $(1 \leq j \leq m)$ and it was member of the communication group in session i, then it can use the broadcast message B_j from session $j > i$, pick up the component z_i from B_j (as B_j contains z_i, $1 \leq i \leq j$) and use the key extraction process (as explained above) to obtain the missing session key for i^{th} session.

For the purpose of **mutual healing**, it is assumed that the sensor nodes are stationary after deployment. The mobile robots compute the location l_x for each node u_x in the network, using a secure localization process such as in [71]. The nodes are communicated their respective locations by one of the mobile robots. Once the localization process is completed, the mobile robots leave the network. During mutual healing process, this location information will be used for neighbor authentication. The mutual healing process is carried out in two sub-phases that work as follows:

Mutual Healing Request: When a node u_x needs the broadcast message B_t, it broadcasts an authenticated request to the nodes in its local neighborhood:

$$u_x \rightarrow *: ID_x,\ E_{K_c}(l_x),\ t,\ c,\ PRF_{K_c}(ID_x \| l_x \| t \| c)$$

Here,

ID_x	-	ID of requesting node u_x
l_x	-	location of u_x
K_c	-	Group key in some previous session c, of which u_x was member
t	-	the session for which broadcast message is being requested
PRF()	-	Pseudo Random function

Mutual Healing Response: A node u_y, which receives this mutual healing request, first confirms that the message is from an authenticated node which is in its one-hop communication range. Node u_y verifies $|l_x - l_y| \leq R$ (Here, R is assumed to

be one-hop communication distance between two nodes in the sensor network, based on the communication range of the nodes). If the location verification fails, node u_y simply discards the request, else it uni-casts a reply to node u_x as follows:

$$u_y \rightarrow u_x\text{: } ID_y,\ E_{K_c}(l_y),\ z_t,\ N_y,\ PRF_{K_c}(ID_x \| ID_y \| l_x \| l_y \| z_t \| N_y \| t \| c)$$

We add a *Key Confirmation Message* for the responding node to ensure that the requesting node has correctly extracted the required key:

$$u_x \rightarrow u_y\text{: } ID_x,\ E_{K_t}(N_y + 1),\ PRF_{K_t}(ID_x \| ID_y \| (N_y + 1))$$

In this protocol, the responding node u_y sends the t^{th} component z_t of the broadcast message B_t meant for the requested session t, along with its own identity and encrypted location. The responding node does not need to send the entire broadcast message B_t, since the requesting node has specified which broadcast message it has missed, so it suffices to send the keying material specific to that session i.e. z_t. It also sends a nonce N_y which is used by the requesting node u_x to confirm the reception of the message and recovery of the required key K_t. The nodes must authenticate each other to ensure that the keying material is being requested/sent by a valid user. For authentication purpose, the protocol uses a keyed PRF such as HMAC [88] of all the transmitted values using a key shared by the group in some previous session c. Node u_x shares this previous session number c whose key is used for generating the PRF() for the request message. When a node u_y receives the request message, it checks if it holds the key K_c and the requested key K_t. If so, the node u_y computes the PRF() using the corresponding key K_c and is assured of the authenticity of the requesting node u_x. Similarly, u_x authenticates u_y by calculating the PRF() of the response message sent by node u_y. Here, we assume that the requesting and responding nodes share at least one session key from some previous session to complete this protocol. Once, requesting node u_x receives the broadcast message component z_t, it recovers the key for session t, using the key extraction process. Then, it sends (N_y + 1) encrypted by the newly recovered key K_t back to responding node u_y to confirm the receipt of the correct message and the required key.

Adding and Revoking Node. The protocol also allows adding and revoking of a node from the group. Suppose a node u_{new} willing to join session j requests GM

for including it in the communication group CG_j. Node u_{new} sends its identity, ID_{new}, and the session number j, the session from which it is interested to join the group. GM validates the identity of u_{new} by looking at the list of non-revoked nodes. If GM finds node u_{new} to be an authorized node, GM includes the new node's public key $Q_{ID_{new}}$ in the computation of Q and also computes U_i for this new node while preparing the broadcast message i.e.

$Q = Q_1 + Q_2 + \ldots + Q_{new} + \ldots + Q_d$

$U_0 = r_j.P_{Pub}$

$U_i = r_j.(Q - s.Q_i)$, for $i = 1$ to d

(including $i = new$)

$\vdots$

Similarly, when a node, say u_{rov}, is revoked from the j^{th} session, GM excludes its public key Q_{rov} in the computation of Q and, U_i value corresponding to this revoked node u_{rov} will not be included in the broadcast message i.e.

$Q = Q_1 + Q_2 + \ldots + Q_d$,

(excluding Q_{rov})

$U_0 = r_j.P_{Pub}$

$U_i = r_j.(Q - s.Q_i)$, for $i = 1$ to d

(excluding $i = rov$)

$\vdots$

Neither node u_{rov}'s public key is used in the computation of Q in the subsequent sessions, nor is U_i computed for $i = rov$. Therefore, even if u_{rov} receives the broadcast message B_t for a session $t \geq j$, it would not be able to extract the key K_t for session t.

The newly joined node can participate in the j^{th} session and subsequent sessions (if he is chosen as the valid member in the group in a particular session). A revoked user would not be able to participate in the j^{th} session and any subsequent sessions (unless it is joined again by GM)

4.4.4 Security Analysis

The security features of the proposed protocol are presented in the form of theorems with proofs by contradiction. We follow the same notations as in chapter 1, that is, $Attacker(T)$ implies adversary has access to a term/name/variable/channel T and, $T_1 \wedge T_2$ and $T_1 \vee T_2$ denotes logical AND and OR operations on T_1 and T_2, respectively. As, the protocol uses public key setup, the public parameters (G_1, G_2, q, P, P_{Pub}, H_1, H_2) are accessible to all the users. The hash function $h()$ is publicly available as well. The broadcast message B_j can be accessed on the public channel for each session j. The private key s of GM is secret that is known only to the GM. Therefore:

Attacker(P)	... (4.1.1)
Attacker(P_{Pub})	... (4.1.2)
Attacker(H_1)	... (4.1.3)
Attacker(H_2)	... (4.1.4)
Attacker($h()$)	... (4.1.5)
Attacker(B_j)	... (4.1.6)
Not Attacker(s)	... (4.1.7)

Theorem 4.1. *The protocol does not allow a group member $u_x \notin CG_j$, not authorized to participate in session j, to obtain the session key K_j through self-healing in any future session $> j$.*

Proof. We consider a user $u_x \notin CG_j$. Therefore:

Attacker($U_x \notin z_j$)	(from Group Key Broadcast process)	... (4.2.1)
Attacker(S_x)	(private key of the user u_x)	... (4.2.2)

Now, let us assume that the user $u_x \notin CG_j$ has obtained the session key K_j:

Attacker(K_j)

$\Rightarrow$ Attacker(V_j) $\wedge$ Attacker(H_2) $\wedge$ ($e(P_{Pub}, r_j.Q)$)

[from step 3 of the Authentication and Key Extraction (para 4.4.2.3)]

$\Rightarrow$ Attacker(z_j) $\wedge$ TRUE $\wedge$ ($e(P_{Pub}, r_j.Q)$)

[from step 5 of Group Key Broadcast (para 4.4.2.2)

and using assertions 4.1.4]

$\Rightarrow$ Attacker(B_j) $\wedge$ ($e(P_{Pub}, r_j.Q)$)

[from step 6 of Group Key Broadcast (para 4.4.2.2)]

$\Rightarrow$ TRUE $\wedge$ ($e(P_{Pub}, r_j.Q)$) [since B_j is available on public channel]

$\Rightarrow$ Attacker($e(P_{Pub}, U_x).e(U, S_x)$)

[from step 2 of Authentication and Key Extraction (para 4.4.2.3)]

$\Rightarrow$ Attacker($U_x \in z_j$)

$\Rightarrow$ FALSE [using assertion 4.2.1]

This contradicts to our assumption that user $u_x \notin CG_j$ has obtained the session key K_j. □

Theorem 4.2. *A group member does not accept a broadcast message from an unauthenticated source.*

Proof. In the proposed protocol, a group member accepts the authenticity of the broadcast message by verifying the signature received from the group manager along with the message.

Let us assume that an attacker has successfully been able to authenticate itself as an authorized group manager:

Attacker($Sign_j$)

$\Rightarrow$ Attacker($(h(B_j) + s)^{-1}$)

$\Rightarrow$ Attacker($(h(B_j) + s)$)

$\Rightarrow$ Attacker($(h(B_j))$ $\wedge$ Attacker(s)

$\Rightarrow$ Attacker($(h())$ $\wedge$ Attacker(B_j) $\wedge$ Attacker(s)

$\Rightarrow$ TRUE $\wedge$ TRUE $\wedge$ FALSE [using assertions 4.1.5, 4.1.6 and 4.1.7 respectively]

$\Rightarrow$ FALSE

An unauthorized GM, therefore, can not authenticate itself. □

Theorem 4.3. *Only a member $u_x \in CG_t \bigcup CG_c$ ($c < t$) can successfully participate in the mutual-healing protocol for session t.*

Proof. The public functions such as $PRF()$ are available to all the users. The messages exchanged for mutual-healing are available on the public channel, therefore:

Attacker(ID_x)	... (4.3.1)
Attacker(ID_y)	... (4.3.2)
Attacker(t)	... (4.3.3)
Attacker(c)	... (4.3.4)
Attacker(z_t)	... (4.3.5)
Attacker(N_y)	... (4.3.6)
Attacker($PRF()$)	... (4.3.7)

The mutual-healing in the proposed protocol involves two entities, a requesting node and a responding node.

As a requesting node, a node must be able to send the correct key confirmation message. Let us assume that a node $u_x \notin CG_t \bigcup CG_c$ ($c < t$) has successfully sent the key confirmation message to complete the mutual-healing protocol for session t:

Attacker(*Key Confirmation Message*)

$\Rightarrow$ Attacker(ID_x) $\wedge$ Attacker($E_{K_t}(N_y + 1)$) $\wedge$

Attacker($PRF_{K_t}(ID_x \| ID_y \| (N_y + 1))$))

[from Key Confirmation in Mutual Healing protocol (subsection 4.4.3)]

$\Rightarrow$ Attacker(ID_x) $\wedge$ Attacker(ID_y) $\wedge$ Attacker(N_y) $\wedge$

Attacker($PRF()$) $\wedge$ Attacker(K_t)

$\Rightarrow$ TRUE $\wedge$ TRUE $\wedge$ TRUE $\wedge$ TRUE $\wedge$ Attacker(K_t)

[using assertion 4.3.1, 4.3.2, 4.3.6 and 4.3.7 respectively]

$\Rightarrow$ Attacker(K_t)

$\Rightarrow$ FALSE [from Theorem 4.1 for $u_x \notin CG_t$]

A responding node must be able to send the valid mutual-healing response message. We assume that a node $u_y \notin CG_t \bigcup CG_c$ ($c < t$) is able to send a mutual-healing response. Therefore:

Attacker(*Mutual-healing Response*)

$\Rightarrow$ Attacker(ID_y) $\wedge$ Attacker($E_{K_c}(l_y)$) $\wedge$ Attacker(z_t) $\wedge$ Attacker(N_y) $\wedge$

Attacker($PRF_{K_c}(ID_x \| ID_y \| l_x \| l_y \| z_t \| N_y \| t \| c)$)

$\Rightarrow$ Attacker(ID_y) $\wedge$ Attacker(K_c) $\wedge$ Attacker(z_t) $\wedge$ Attacker(N_y)

$\wedge$ Attacker($PRF()$) $\wedge$ Attacker(ID_x) $\wedge$ Attacker(l_x) $\wedge$ Attacker(l_y)

$\wedge$ Attacker(t) $\wedge$ Attacker(c)

$\Rightarrow$ TRUE $\wedge$ Attacker(K_c) $\wedge$ TRUE $\wedge$ TRUE $\wedge$ TRUE $\wedge$ TRUE $\wedge$ Attacker(l_x)

$\wedge$ Attacker(l_y) $\wedge$ TRUE $\wedge$ TRUE

[using assertion 4.3.2, 4.3.5, 4.3.6, 4.3.7, 4.3.1, 4.3.3 and 4.3.4 respectively]

$\Rightarrow$ Attacker(K_c) $\wedge$ Attacker(l_x) $\wedge$ Attacker(l_y)

$\Rightarrow$ Attacker(K_c)

$\Rightarrow$ FALSE [from Theorem 4.1 for $u_x \notin CG_c$]

This contradicts to our assumption that a user u_x (and/or u_y) $\notin CG_t \bigcup CG_c$ could successfully complete the mutual-healing protocol. □

4.4.5 Performance Analysis

4.4.5.1 Computation cost

A node after receiving the broadcast message, takes the j^{th} component and for computing $e(P_{Pub}, r_t.\mathrm{Q})$, it needs to perform two bilinear pairing operations, one for $e(P_{Pub}, U_x)$ and other for $e(U_0, S_x)$. Then it computes the hash $H_2(e(P_{Pub}, r_t.Q))$ and finally extracts key K_j using an XOR operation. The hash and XOR computation cost is negligible in comparison to the bilinear pairing computation. So, if T_p is the cost of bilinear pairing computation, then self-healing and key-recovery operation takes only 2 * T_p.

In Tian *et al.*'s proposal [120], GM needs to define a $|G_{j-1}|$ X $|G_j|$ matrix and requires to compute $|G_{j-1}|$ additional ECC points using public keys of the members of current communication group, in order to construct the broadcast message. A node in the group needs to perform matrix inversion operation. Secondly, during the mutual healing process, the responding node encrypts the requested broadcast message with a key generated from the location based key using bilinear pairing operation. The requesting node needs to calculate the same key again using bilinear pairing operation.

In our proposal, we avoid the need of any matrix, additional ECC points computations by GM, matrix inversion operation by node and bilinear pairing operations

for authentication during mutual healing. For mutual healing, our protocol [3] requires one symmetric key encryption, and one PRF() computation, one symmetric key decryption, one simple comparison to check Euclidean distance and one PRF() to verify response and, one symmetric key encryption for key confirmation as a requesting node. In case node is responder, then it needs one symmetric key decryption, one simple comparison to check Euclidean distance and one PRF() to verify request, then for response one symmetric key encryption, and one PRF() computation. When compared with the protocol in [120], we find that computation cost is greatly reduced, as now a node does not require to solve system of linear equations and also it could avoid doing scalar multiplication with respect to all other nodes in the group in order to recover a key.

4.4.5.2 Communication cost

The group manager broadcasts one single message in each session. So, for m sessions, there will be total m broadcast messages. In each session j, $j(2d+4) \log q$ bits are transmitted as B_j, which is similar to Tian's protocol [120]. At node end, node receives a broadcast message in each session and in usual operation, it does not need to transmit anything for obtaining the group secret key. When a node needs to obtain the group key information through mutual healing, it needs to send out two messages, one broadcast message as mutual healing request and another key confirmation message to the responding node.

The overall bits communicated in the process is much less as compared to Tian's protocol, with an additional security feature of key confirmation. Since the requesting node is specifying the session number t for which it has missed the broadcast message, it implies that the requesting node had received all the previous messages, so the responding node need not to send the whole broadcast message, but can only send the t^{th} component i.e. z_t from the broadcast message B_t.

4.4.5.3 Storage cost

Each node stores its own public-private key pair and the group session keys for maximum m sessions. It also needs to temporarily store the broadcast message, which contains j ($1 \leq j \leq m$) entities. We are able to save on the cost of storing a $\|G_j|$ x $\|G_j|$ matrix and also a node does not need to store the public keys of all the nodes in the group to recover a session key, unlike in [120].

4.5 CRT based Symmetric Key Healing Protocol

The bilinear pairing based healing in group key distribution is computationally heavy on resource constrained sensor nodes due to bilinear pairing computation involved in key extraction process. The bilinear pairing approach also incurs more storage overhead on the nodes due to public key setup. Therefore, we propose a symmetric key based protocol that uses Chinese Remainder Theorem based secret sharing for group key distribution. The goals and assumptions remain the same as with the proposed bilinear pairing based healing protocol discussed in section 4.4, however, the objective here is to provide the similar security with notably reduced cost overhead. The detailed discussion of the proposed CRT based healing protocol follows.

4.5.1 System Model

The proposed protocol is defined with the help of following modules:

(a) (ID, S, HK, w) = **Setup**(U):

Input : U - Set of nodes in the network

Output : ID - Set of unique identities for nodes

S - Set of personal secrets for nodes

HK - Set of hash chains of auxiliary keys created by each GM

w - Self-healing window size

The *Setup* module sets up system for key distribution and healing for nodes in set U to assign identities and keys to all the nodes in the network.

(b) B_j = **GpKeyDistr**(K_j, CG_j, N_j, KA_j, S, w):

Input : K_j, CG_j, N_j, KA_j (for session j), S, w

K_j - Group shared secret for session j

CG_j - Set of nodes authorized for communication in session j

N_j - Nonce

S - Set of individual node secrets

w - Self healing window size

Output : B_j - Broadcast message in session j

Module *GpKeyDistr* is used to construct broadcast message for session j with group session key K_j, nonce N_j and auxiliary key KA_j using the personal secrets of nodes from set S for the nodes in communication group CG_j and, returns the set B_j of broadcast messages consisting of all z_l ($(j - w) \leq l \leq j$) for self-healing window of size w.

(c) K_j = **AuthKeyExtr**(B_j, s_i):

Input : B_j - Broadcast received in session j

s_i - Personal secret of node u_i

Output : K_j - Group key for session j

AuthKeyExtr module authenticates the broadcast message set B_j received by a node u_i from group manager in session j and extracts the session key K_j for node u_i using its personal secret s_i.

4.5.2 Session Key Management

The proposed protocol for healing enabled group key distribution using CRT based secret sharing has the following mandatory phases: *system set up*, *group key construction and distribution* and, *authentication and key extraction*. If a node misses broadcasts for one or more consecutive sessions, the *self-healing* is performed. When the current session broadcast is missed by a node and the node

requires to obtain the information in the current session itself then *mutual-healing* is executed. We describe each of the phases in the subsequent paragraphs.

4.5.2.1 System Setup

Prior to deployment, the base station assigns unique identity $ID_i \in_R Z$ to each node $u_i \in U$. A personal secret $s_i \in_R Z_q^*$ is also assigned to each node. A cluster head acts as a group manager (GM) managing a subset of nodes. The base station secretly gives the personal secret $s_i \in S$ for each node u_i to its respective GM. Each group manager selects a key seed $KA_m \in_R Z_q^*$. Using the seed KA_m, the GM computes hash chain of auxiliary keys as:

$$KA_0 = h(KA_1 = h(KA_2 = \ldots = h(KA_m)))$$

This hash chain is added to the set HK of hash chains. The auxiliary key KA_j associated with a communication session j is used for authentication purpose during mutual-healing. The GM also sets up the self-healing window size w ($\ll m$) and the initial session number as $j = 0$.

4.5.2.2 Group Key Message Construction and Distribution

Just before the start of a communication session j ($j \geq 0$), GM chooses the session key $K_j \in_R Z_q^*$. GM picks the auxiliary key KA_j corresponding to session j from the hash chain. For each node u_i, a unique value is computed using the secret K_j and the node's personal secret s_i. GM uses these node specific unique values to construct the broadcast message with the help of Chinese remainder theorem (CRT) based secret sharing for secretly distributing the session key. The PRF values over the elements being shared using the individual node secrets are also computed for broadcast authentication. The GM picks the broadcast messages constructed for last w (the self-healing window size) sessions that is $z_{j-w}, z_{j-w+1}, \ldots, z_{j-1}$ and broadcasts these messages along with the current broadcast message z_j.

Algorithm 4.1 gives the detailed steps for group-key message construction and distribution by GM.

Algorithm 4.1 Group Key Message Construction and Distribution

Input : K_j, CG_j, N_j, KA_j (for session j), S, w
K_j - Group shared secret for session j
CG_j - Set of nodes authorized for communication in session j
N_j - Nonce chosen for session j
S - Set of individual node secrets
w - Self healing window size

Output : B_j - Broadcast message in session j

1: **procedure** GROUPKEYDIST(K_j, CG_j, N_j, KA_j, S, w)
2: Select group key $K_j \in_R Z_q^*$ and a nonce $N_j \in_R Z_q^*$
3: Pick auxiliary key $KA_j \in_R Z_q^*$ from hash-chain
4: Compute $V_j = \mathrm{h}(K_j) \oplus KA_j$
5: **for** each node $u_i \in CG_j$ **do**
6: Compute $b_{ij} = s_i \oplus K_j$
7: Compute $r_{ij} = \lfloor (s_i \oplus K_j)/s_i \rfloor$
8: **end for**
9: Solve congruence $X_j \equiv b_{ij} \bmod s_i$ ($\forall\ u_i \in CG_l$)
10: Solve congruence $Y_j \equiv r_{ij} \bmod s_i$ ($\forall\ u_i \in CG_l$)
11: **for** each node $u_i \in CG_j$ **do**
12: Compute $c_{ij} = PRF_{s_i}(ID_i, X_j, Y_j, V_j, N_j)$
13: **end for**
14: Solve congruence $W_j \equiv c_{ij} \bmod s_i$ ($\forall\ u_i \in CG_j$)
15: Set up $z_j = (X_j, Y_j, V_j, W_j, N_j)$
16: Set up $B_j = (z_{j-w}, z_{j-w+1}, \ldots, z_{j-1}, z_j)$
17: **return** B_j
18: **end procedure**

4.5.2.3 Authentication and Key Extraction

When a node u_i listens to a broadcast, it checks the authenticity of the message. It computes the PRF using its own secret s_i and compares with the PRF value received to check the authentication. On successful authentication, it confirms the freshness of the message by comparing the received nonce with the nonce of the previous session and then extracts the new group session key. The detailed steps are as given in Algorithm 4.2.

Algorithm 4.2 Authentication and Key Extraction

Input : B_j - Broadcast received in session j
s_i - Personal secret of node u_i
Output : K_j - Group key for session j
fl_0 - Flag bit for authentication check
fl_1 - Flag bit for freshness check

1: **procedure** AUTHKEYEXT(B_j, s_i)
2: Picks $z_j = (X_j, Y_j, V_j, W_j, N_j)$ from B_j
3: Computes $c'_{ij} = PRF_{s_i}(ID_i, X_j, Y_j, V_j, N_j)$
4: Retrieves c_{ij} from W_j as $c_{ij} = W_j \bmod s_i$
5: **if** $c'_{ij} = c_{ij}$ **then**
6: $fl_0 = 1$ "Authentication Successful !!"
7: **else**
8: $fl_0 = 0$ "Authentication Failed !!" Discard z_j
9: **end if**
10: **if** $N_j > N_{j-1}$ (for $j > 0$) **then**
11: $fl_1 = 1$
12: Extracts $K_j = ((X_j \bmod s_i) + (Y_j \bmod s_i)*s_i) \oplus s_i$
13: **return** K_j
14: **else**
15: $fl_1 = 0$
16: "Replay Attack !!" Discard z_j **return**
17: **end if**
18: **end procedure**

4.5.3 Healing

When a node misses a session broadcast message, it will not be able to extract the group key for that particular session. With ***self-healing***, such node can extract the missed key from a future broadcast, if the node is an authorized group member for the specified session for which it missed the broadcast. A node missing one or more broadcasts waits for a future broadcast. The GM sets a self-healing window of size w, assuming that a node missing broadcasts should recover from within a limited number of future broadcasts. Whenever, a next broadcast is received, the node can pick all the missing broadcast components (maximum w) and extract the missing keys. As depicted in Figure 4.1, suppose at some point of time the

missing count $c = 0$, that is a node has been receiving all the broadcasts. If a node misses a broadcast in session j, it waits for the next broadcast, so missing count c in incremented by 1. As long as the node does not receive a broadcast, it keeps waiting and the missing count keeps increasing. Eventually, when the node receives a broadcast, it checks the missing count (if the missing count is more than 0, then only node would require self-healing). The total miss tm is set to the missing count c and then node starts recovering keys (for whatever missed sessions it needs).

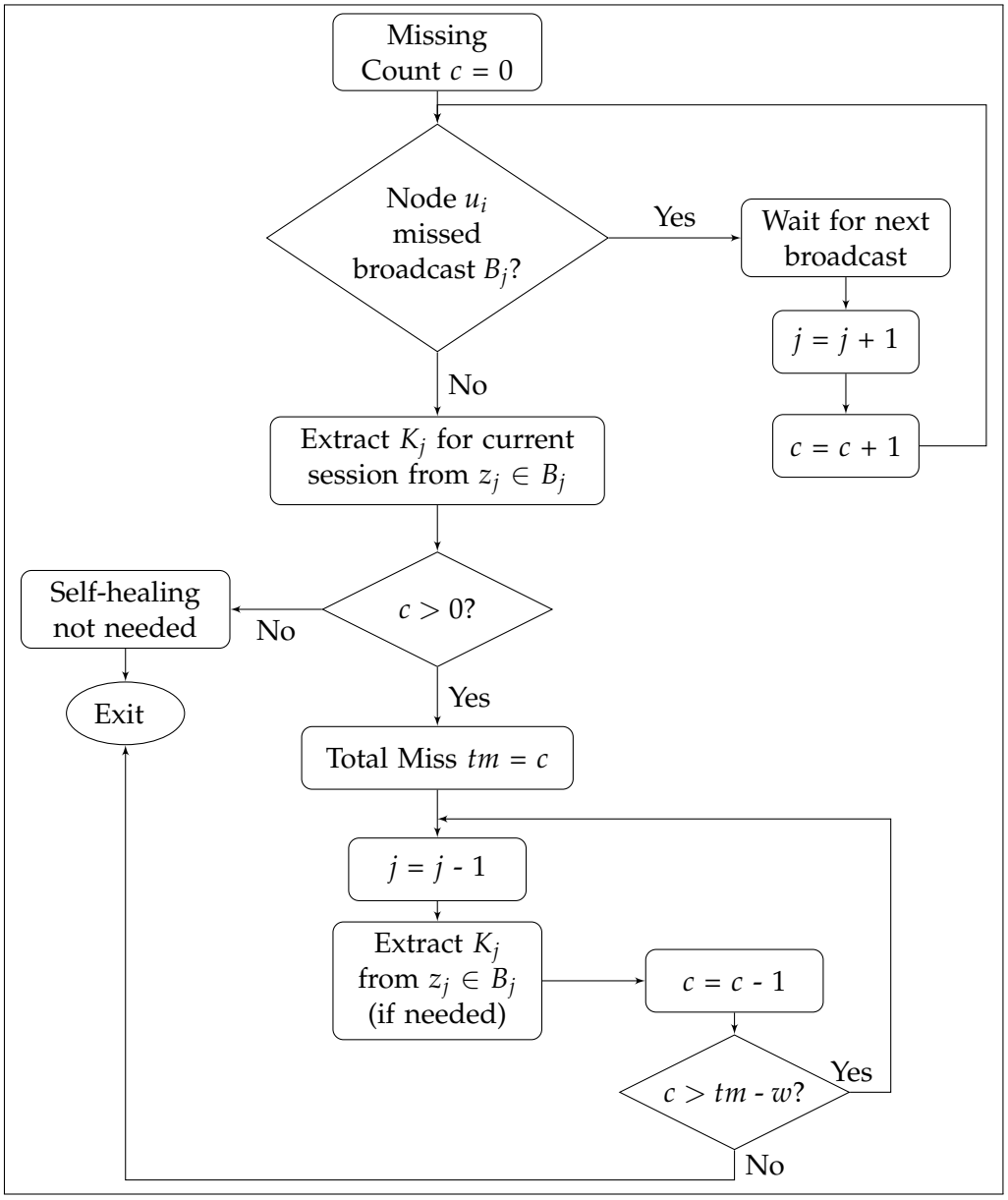


Figure 4.1: Self-Healing

Although, with self-healing, a node can easily recover the missing keys using future broadcasts, a node missing the current broadcast may not want to wait for the future broadcasts. In such scenarios, a node can use the ***mutual-healing*** protocol to obtain a missing broadcast by requesting its neighbor nodes. The mutual-healing protocol involves three steps. In the first step, a node u_x, that needs mutual-healing, broadcasts a *mutual-healing request*. A neighboring node u_y listening to this broadcast may opt to respond by unicasting the *mutual-healing response*. After ensuring that the correct session key is received, the requesting node sends the *key confirmation* message to the responder. The detailed steps of the mutual-healing protocol are given in Figure 4.2.

4.5.4 Security Analysis

We present the security strengths of the proposed protocol in the form of theorems and prove them using proof by contradiction. We use the following existing results in the proof of theorems.

Statement 4.1: For a one-way cryptographic hash-function with collision resistance $h(.)$ defined as $h : \{0,1\}^n \rightarrow \{0,1\}^l$, the probability of finding a $x^{'} \neq x$, such that $h(x) = h(x^{'})$ for a given x and $h(x)$, i.e. the probability of hash collision [101] is:

$$\mathcal{P}r\textbf{[(}h(x) \textbf{=} h(x^{'})\ \textbf{(}x^{'} \neq x\textbf{))]} \textbf{=} 1 \textbf{-} (1 - \tfrac{1}{2^l})^{n-1}$$

Statement 4.2: The probability $\mathcal{P}r$[forge] of accepting forged data as authentic with HMAC is $\frac{2^t}{2^\lambda}$, where λ is the length of the output of the HMAC in bits and 2^t is the number of failed verifications allowed [34].

The protocol uses HMAC SHA-256 [88] as PRF to authenticate the communication. Using the session key K_j and the node secrets s_i for all $u_i \in CG_j$, the GM in the protocol computes the congruence value X_j, supporting congruence Y_j and V_j to protect auxiliary key, for a session j. Then, the GM computes PRF values of these contents along with nonce N_j using individual secrets s_i of the nodes. The

Requesting Node u_x

1. Picks a nonce $N_x \in_R Z_q^*$

2. Mutual Healing Request (Broadcast):

$\xrightarrow{Id_x,\, t,\, N_x}$

(To all neighbor nodes)

Responding Node u_y

3. Picks a nonce $N_y \in_R Z_q^*$
4. Extracts KA_t from z_t as $KA_t = V_t \oplus \mathrm{h}(K_t)$
5. Computes $PRF_{KA_t}(Id_x \| Id_y \| N_x + 1 \| N_y)$

6. Mutual Healing Response:

$\xleftarrow{Id_y,\, z_t,\, N_y\ PRF_{KA_t}(Id_x \| Id_y \| N_x + 1 \| N_y)}$

7. Extracts K_t from z_t (using key extraction)
8. Computes $KA_t = V_t \oplus \mathrm{h}(K_t)$
9. If $\mathrm{h}(KA_t) \neq KA_{t-1}$ then 'Discard respond from u_y'
   EXIT
10. Computes $PRF_{KA_t}(Id_x \| Id_y \| N_x + 1 \| N_y)$
11. If Computed PRF $\neq$ Received PRF then
    'Message Corrupted'. EXIT
12. Computes $PRF_{K_t}(Id_x \| Id_y \| N_y + 1)$

13. Key Confirmation:

$\xrightarrow{Id_x,\, PRF_{K_t}(Id_x \| Id_y \| N_y + 1)}$

14. Computes $PRF_{K_t}(Id_x \| Id_y \| N_y + 1)$
15. If Computed PRF $\neq$ Received PRF then
    'Message Corrupted'; EXIT
16. 'Mutual Healing Success' !

Figure 4.2: Mutual-Healing

PRF value is distributed as a congruence value computed using individual secrets of all the participating nodes. Here, the individual node secret s_i is shared only between node u_i and the GM. The broadcast message contains $z_j = (X_j, Y_j, V_j, W_j,$

N_j) and the communication channel is public. Therefore:

Attacker($z_j = (X_j, Y_j, V_j, W_j, N_j)$)	... (4.4.1)
Not Attacker(s_i) $\forall\ u_i \in U$	... (4.4.2)

Theorem 4.4. *An adversary can evade the authentication process by forging the PRF with probability $\mathcal{P}r$[forge] = $\frac{1}{2^{(\lambda-t)}}$, where λ is the output size of the PRF used for authentication and 2^t is the number of failed verifications allowed.*

Proof. Let us assume that an adversary is able to evade the authentication process. This implies that the adversary has reached step 6 of Algorithm 4.2 and could successfully compute c'_{ij} (= c_{ij}). If we trace back the adversary steps:

Attacker(c_{ij}) for some $u_i \in CG_j$

$\Rightarrow$ Attacker($PRF_{s_i}(ID_i, X_j, Y_j, V_j, N_j)$)

$\Rightarrow$ (Attacker(s_i) $\wedge$ Attacker(ID_i, X_j, Y_j, V_j, N_j))

$\vee$ Attacker($PRF_{s'_i}(ID_i, X_j, Y_j, V_j, N_j)$ (= $PRF_{s_i}(ID_i, X_j, Y_j, V_j, N_j)$))

[for some key $s'_i \neq s_i$ to forge the PRF]

$\Rightarrow$ (FALSE $\wedge$ TRUE) $\vee$ TRUE(with Probability $\mathcal{P}r$[forge] = $\frac{1}{2^{(\lambda-t)}}$)

[using assertions 4.4.1, 4.4.2 and Definition 4.2]

$\Rightarrow$ TRUE(with Probability $\mathcal{P}r$[forge] = $\frac{1}{2^{(\lambda-t)}}$)

Therefore, the assumption that an adversary is able to evade the authentication process holds with the probability $\mathcal{P}r$[forge] = $\frac{1}{2^{(\lambda-t)}}$.

In the proposed protocol, since HMAC SHA-256 is used as PRF, the value of λ (from definition 4.2) is 256. Therefore, the adversary requires 2^{256} attempts to forge data as authentic with HMAC. Even if an adversary makes 2^{200} attempts which is practically infeasible in an operational WSN scenario, the probability $\mathcal{P}r$[forge] = $\frac{1}{2^{256-200}} = 1.39 * 10^{-17}$. □

Theorem 4.5. *The protocol is secure against replay and impersonation attacks*

Proof. In the proposed protocol, the nonce value N_j for each session j is selected in such a manner that $N_{j-1} < N_j$. When a node receives a group key broadcast message from the GM for session j, it checks for the freshness of the nonce value N_j. If nonce $N_j \leq N_{j-1}$, it is discarded. This prevents an attacker from replaying the broadcast message from any previous session. Furthermore, any attacker that

attempts to assume the identity of GM must possess the personal secrets of all the nodes in the cluster in order to construct the broadcast message. To impersonate a node u_x, an attacker must have its personal secret s_x to extract the key and participate in the network operations. We prove the security of the proposed protocol against replay and impersonation attacks by contradiction. We consider different possibilities for an attacker under this scenario as follows:

(i) Suppose the adversary chooses a nonce $N'_j > N_{j-1}$ and successfully carries out replay attack. This implies that:

Attacker(z'_j = (X_{j-1}, Y_{j-1}, V_{j-1}, W_{j-1}, N'_j))

$\Rightarrow$ Attacker($c_{ij} = PRF_{s_i}(ID_i, X_{j-1}, Y_{j-1}, V_{j-1}, N'_j)$) $\forall\ u_i \in CG_j$

$\Rightarrow$ Attacker(s_i)

$\Rightarrow$ FALSE [using assertions 4.4.2]

(ii) Let us assume that an adversary has impersonated the GM to broadcast B_j. Thus:

Attacker(z'_j = (X'_j, Y'_j, V'_j, W'_j, N'_j))

$\Rightarrow$ Attacker($X'_j \equiv b'_{ij} \bmod s_i$) $\wedge$ Attacker($Y'_j \equiv r'_{ij} \bmod s_i$) $\wedge$

Attacker($V'_j = K'_j \oplus KA'_j$) $\wedge$ Attacker($W'_j \equiv c'_{ij} \bmod s_i$) $\forall\ u_i \in CG_j$

$\Rightarrow$ Attacker(s_i)

$\Rightarrow$ FALSE [using assertions 4.4.2]

(iii) If we assume that the adversary has impersonated a node $u_i \in CG_j$, then:

Attacker(K_j)

$\Rightarrow$ Attacker((($X_j \bmod s_i$) + ($Y_j \bmod s_i$)*s_i) $\oplus$ s_i) [from step 12 of Algorithm 4.2]

$\Rightarrow$ Attacker(X_j) $\wedge$ Attacker(Y_j) $\wedge$ Attacker(s_i)

$\Rightarrow$ TRUE $\wedge$ TRUE $\wedge$ FALSE [using assertions 4.4.1 and 4.4.2]

$\Rightarrow$ FALSE

Therefore, the protocol is secured against replay and impersonation attacks. □

Theorem 4.6. *For a given session t, self-healing can be performed only by a group member authorized to participate in session t.*

Proof. The protocol ensures that self-healing for a session t can be done only by a node u_x that was a valid member in session t. This is due to the fact that when key information message z_t is constructed for a session t, the GM computes the

congruences X_t and Y_t that take into account the personal secrets of only those nodes that belong to the communication group CG_t. Therefore, an unauthorized member or an attacker can not perform self-healing even when the broadcast messages from previous sessions are available in current session. We prove this through contradiction, so let us assume that a node $u_x \notin CG_t$ receives broadcast message in a future session j ($t < j$) within self-healing window and $u_x \in CG_j$. Thus:

Attacker($u_x \notin CG_t$)	... (4.5.1)
Attacker(s_x)	... (4.5.2)

Now, suppose node u_x successfully obtains key K_t for session t through self-healing, therefore:

Attacker(K_t)

$\Rightarrow$ Attacker(((X_t mod s_x) + (Y_t mod s_x)*s_x) $\oplus$ s_x) [from step 12 of Algorithm 4.2]

$\Rightarrow$ Attacker($X_t \equiv (s_x \bmod X_t) \bmod s_x$) $\wedge$ Attacker($Y_t \equiv r_{xj} \bmod s_x$) $\wedge$ Attacker(s_x) [from steps 9 and 10 of Algorithm 4.1]

$\Rightarrow$ Attacker($u_x \in CG_t$) $\wedge$ Attacker(s_x)

$\Rightarrow$ FALSE $\wedge$ TRUE [using assertions 4.5.1 and 4.5.2]

$\Rightarrow$ FALSE

Thus, only an authorized member of a given previous session can use self-healing to extract key of that session. □

Theorem 4.7. *The probability of an unauthorized group member responding to mutual-healing request is* 1 - $(1 - \frac{1}{2^l})^{(n-1)}$ *(where n is input size and l the output size of hash respectively)* that is negligible for $l >= 256$.

Proof. Suppose an unauthorized member participates in mutual-healing and responds to a mutual-healing request with its own broadcast component z'_t and nonce N'_y and uses its own auxiliary key KA'_t. Assuming that protocol is successfully completed, the responding node gets the *key confirmation* message. Thus:

Attacker(*key confirmation*)

$\Rightarrow$ Attacker($h(KA'_t) = KA_{t-1}$) [from step 9 of Figure 4.2]

$\Rightarrow$ Attacker($h(KA'_t) = h(KA_t)$) [since $KA_{t-1} = h(KA_t)$]

$\Rightarrow$ TRUE (with probability 1 - $(1 - \frac{1}{2^l})^{(n-1)}$) [from definition 4.1]

In the proposed scenario, the secret size l = log q = 256 bits and the input size = $sizeof(KA_t)$ = 256 bits, resulting in probability of hash collision to 1 - $(1 - \frac{1}{2^{256}})^{(256-1)}$ = 1 - (1- 8.64 * 10^{-78})^{255}. □

The proposed mutual-healing protocol ensures that only a node who posses valid key being the authorized member of session t can respond to mutual-healing request. We see that when an attacker attempts to respond to a mutual-healing request, he chooses a session key K'_t, auxiliary key KA'_t and nonce N'_y and, constructs its own message z'_t. The requesting node u_x receives the response as Id_y, z'_t, N'_y, $PRF_{KA'_t}(Id_x \| Id_y \| N_x + 1 \| N'_y)$. Node u_x gets some key K''_t by following key extraction process (step 7 of Figure 4.2). Using this key it would get some auxiliary key KA''_t. When node u_x verifies auxiliary key by checking if h(KA''_t) = KA_{t-1} (as per step 9 of Figure 4.2), the validation fails and the response is discarded. For a valid response, the auxiliary key validation as well as the PRF verification (step 11 of Figure 4.2) must succeed so that node u_x can extract key K_t and send back the confirmation message. When the responding node u_y successfully validates the key confirmation message, it is assured that the correct key is received by the requesting node. Thus, the protocol also ensure *Key confirmation* in mutual-healing.

4.5.5 Performance Analysis

The proposed healing enabled group key distribution protocol incurs less communication, computation and storage overhead on resource-constrained sensor nodes.

4.5.5.1 Computation cost

For Self-healing, a node has to extract a key from a future broadcast message. The key extraction process demands a node to first compute a PRF of the received value that it uses to authenticate the message and the sender. Once the authentication is successful, the node extracts the key by performing one XOR operation, one multiplication and 2 mod operations. If a node participates in mutual-healing to obtain or provide the keying material, it acts as a requesting node or a responding node. To place a request, the node performs one XOR, one hash and, it requires

to perform one XOR and 2 PRF operations. When a node responds to a mutual-healing request, it needs to perform only one XOR and two PRF operations. As compared to the existing mutual-healing protocols, this is a significant reduction in computation overhead.

4.5.5.2 Communication cost

For self-healing, a node does not transmit any message, so communication cost only involves the cost of message reception. The node receives only $5w$ * log q bits in each session, for self-healing window size w and secret size log q bits. For mutual-healing, as a requesting node, for sending a request and subsequently a key confirmation message, node sends 8 * log q bits of data. In order to respond to a mutual-healing request, the node needs to send 5 * log q bits. It may seem some overhead as compared to the protocols [96], [141], [13] and [80] where the transmission cost overhead is null, but then, these protocols do not provide mutual-healing capability.

4.5.5.3 Storage cost

The storage overhead is low as a node keeps the current session key for self-healing. For mutual-healing purpose, the node stores an additional auxiliary key.

4.6 Comparison with Existing Schemes

4.6.1 Security Features

We compared the security features of our proposed protocols with the reference protocols. As shown in Table 4.1, both our proposed protocols provide the self-healing and mutual-healing capability unlike the CRT based group key distribution protocols proposed in [141] [13] and [80]. The CRT based protocol presented in [96] does provide self-healing, but it does not discuss mutual-healing. Our proposed protocols resist replay and impersonation attacks as opposed to the protocols discussed in [141] [13] and [96]. The protocols ensure authentication that is

not provided in [141] [13]. We also provide additional security in terms of key confirmation at a low cost overhead during mutual-healing as compared to the only existing mutual-healing protocol in [120]. Although, both our proposed protocols provide same security features, the CRT based healing proposal gives significant performance improvement as compared to the bilinear pairing based healing, as discussed in subsequent subsection.

Schemes ⇒ Attributes ⇓	Tian et al. [120]	Hassan et al. [96]	Zheng et al. [141]	Bhaskar et al. [13]	Liu et al. [80]	Proposed pairing based [3]	Proposed CRT based [1]
Resists Replay Attack	✓	✓	×	×	×	✓	✓
Resists Impersonation Attack	✓	✓	×	×	×	✓	✓
Authentication	✓	✓	✓	×	×	✓	✓
Self Healing	✓	×	✓	×	×	✓	✓
Mutual Healing	✓	×	✓	×	×	✓	✓
Key Confirmation*	×	NA	NA	NA	NA	✓	✓

* - in mutual-healing
NA - Not Applicable

Table 4.1: Comparison of Security Features

4.6.2 Performance

We present the analytical comparison of the performance of our proposals with some of the relevant schemes in Table 4.3. The notations used in the performance table are given in Table 4.2.

Notation	Description
T_{bp}	Time taken for Bilinear Pairing operation
T_s	Time taken for Symmetric key operations such as encryption/ decryption/ hash/ PRF
T_p	Time taken for Public key operations such as signature verification/ scalar multiplication
T_b	Time taken for basic operations such as XOR, Mod
T_a	Time taken for Solving system of algebraic equations
T_c	Time taken for CRT congruence solving
$\log q$	Size of secret in bits

Table 4.2: Notations used in Performance Comparison

Attributes ⇒ / Schemes ↓	Computation Overhead			Communication Overhead			Storage Overhead	
	Self Healing	Mutual Healing		Self Healing	Mutual Healing		Self Healing	Mutual Healing
		Request	Response		Request	Response		
Tian et al. [120]	$2T_{bp}$+$(d$+1$)T_p$ +T_s +T_a	T_{bp}+$2T_s$ +T_p	T_b+T_s	($2d$+3)j log q	3log q	(($2d$+3)j +2)log q	($2jd$+$2j$+5) log q+$2d$log q +16(d^2+d)	2log q
Proposed bilinear pairing based [3]	$2T_{bp}$+T_s +T_p	$6T_s$	$6T_s$	($2d$+3)j log q	8log q	(($2d$+3) +4)log q	($2jd$+$2j$+5) log q log q	2log q
Hassan et al.[96]	$2T_{bp}$+dT_p+ $2T_s$+T_a	NA		($2d$+1) log q	NA		(4d+2)log q+ 16(d^2+d)	NA
Zheng et al. [141]	$2T_b$			log q			2log q	
Bhaskar et al. [13]	T_s+$2T_b$			2 log q			3log q	
Liu et al.[80]	T_c+T_b+$2T_s$			(2d+6) log q			2d log q+3	
Proposed [1]	T_s+$3T_b$	$4T_s$+T_b	$3T_s$+T_b	5w log q	8log q	5log q	log q	log q

Table 4.3: Performance Comparison

With our proposed bilinear pairing based protocol, we reduce the computation overhead of scalar multiplications and solving of system of algebraic equations that was needed by Tian *et al.* protocol [120]. The storage overhead of keeping the public keys of all other group nodes is also avoided as opposed to the requirement in [120]. The CRT based protocol proposed for self-healing and mutual-healing further reduces the overheads considerably, as it is based on symmetric key cryptographic primitives to be used at node end. Although, the overhead seems to be higher as compared to the existing CRT based schemes discussed in [141] and [13], our proposed scheme trades-off for security features of self-healing, mutual-healing, resistance to replay and impersonation attacks and authentication. The protocol proposed in [80] has more communication and storage cost in comparison to our proposed CRT based protocol and does not provide healing capability.

4.7 Experimental Results

We carried out the simulation with Castalia Simulator considering 50 nodes within a cluster and self-healing window size of 5 for proposed CRT based protocol. The simulation results show that with the proposed healing protocol using bilinear

pairing, the cost for communication for a mutual-healing protocol is significantly reduced. The communication cost for self-healing as well as for mutual-healing response is notably reduced with our CRT based healing protocol (Refer Table 4.4). The comparison of storage overhead in terms of Kbits is shown in Figure 4.3.

Scheme ⇒ / Features ⇓	Tian *et al.* [120]	Proposed bilinear pairing based [3]	Proposed CRT based [1]
Self-Healing	2.060	2.060	0.025
Mutual Healing - Request	0.003	0.008	0.008
Mutual Healing - Response	2.062	0.107	0.005

Table 4.4: Communication Cost in Sec

The cost of storage with the existing healing protocol increases as the number of nodes and the number of sessions are increased. With the help of proposed bilinear pairing based healing, we are able to significantly reduce this increase as the number of nodes in the network grows and the number of sessions increase. The proposed CRT based healing protocol has very low and constant storage overhead irrespective of the growth in network size and the number of sessions.

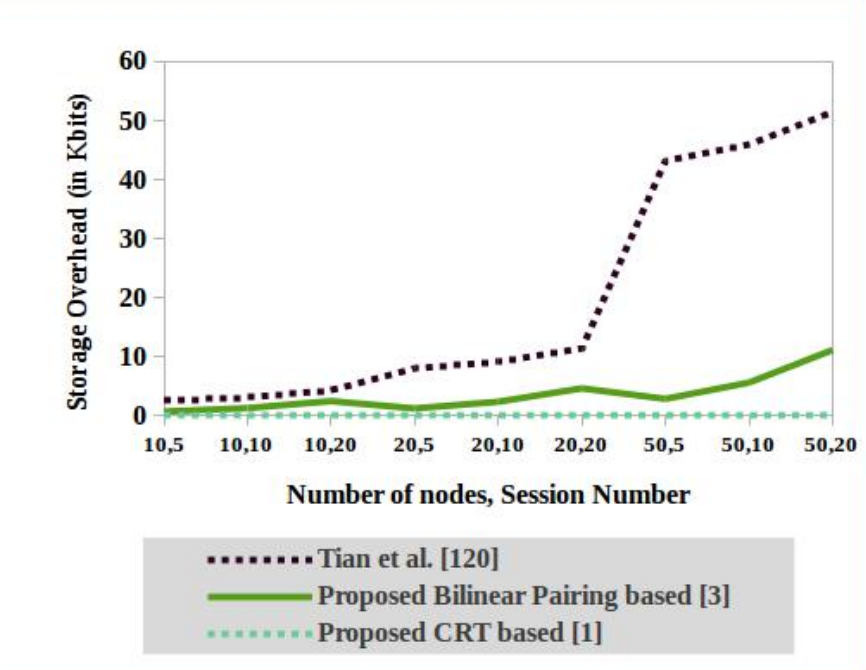


Figure 4.3: Storage Cost for Mutual-Healing

We also carried out the experiments on ATMega 328 processor using Arduino

Duemilanove controller board and ArduinoISP programmer to evaluate the computation cost for self-healing and mutual-healing (Refer Figures 4.4 and 4.5). As

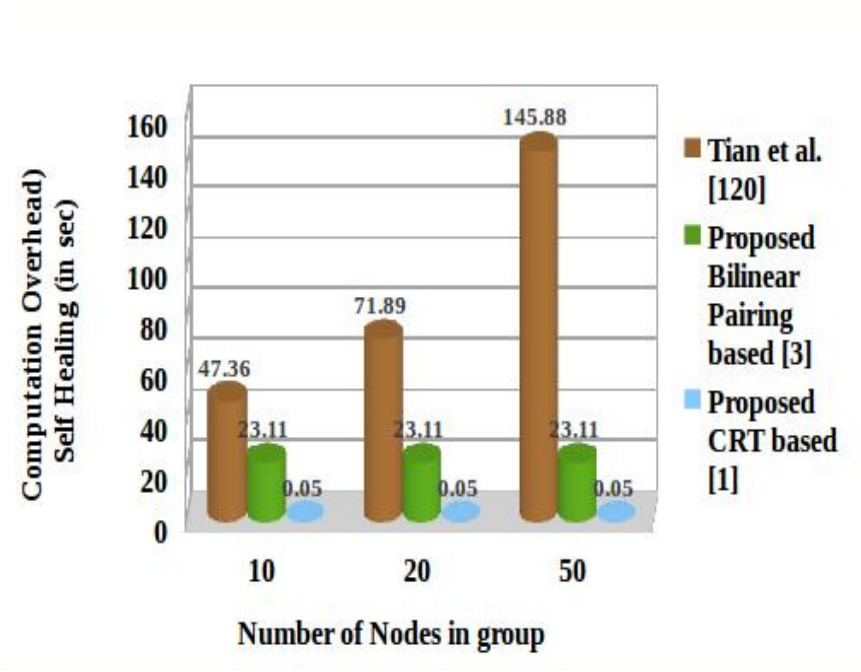


Figure 4.4: Computation Cost for Self-Healing

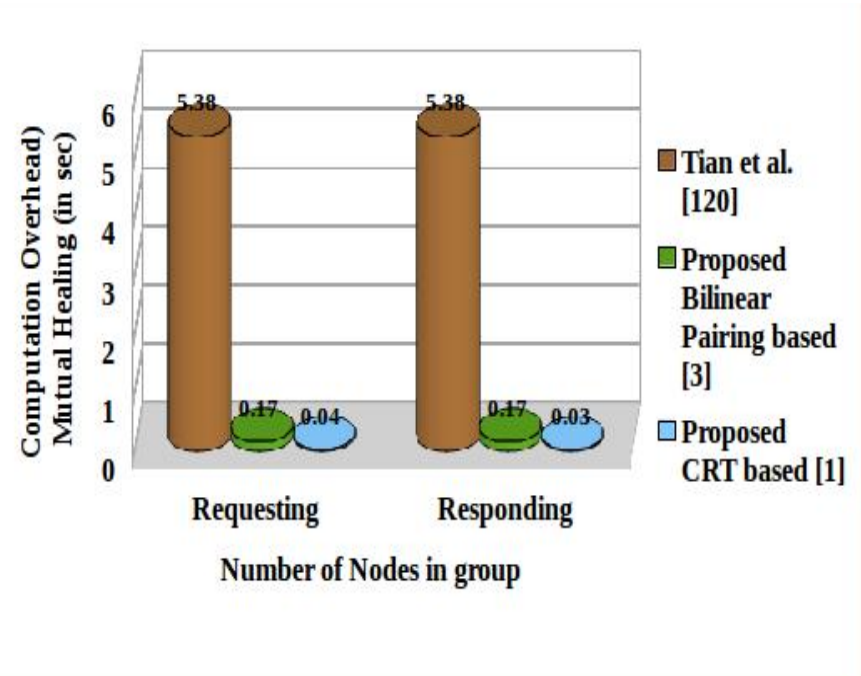


Figure 4.5: Computation Cost for Mutual-Healing

seen in the Figure 4.4, with our proposed protocols, the computation overhead

for self-healing remains constant irrespective of the increase in the group size. Specifically with the CRT based protocol, the cost of computation incurred in self-healing is notably low. Also, the computation cost for mutual-healing (Figure 4.5) drops down significantly with our proposed protocols as compared to the existing mutual-healing protocol proposed in [120].

4.8 Conclusion

In this chapter, we have discussed the self-healing and mutual-healing enabled group key distribution protocols for WSN. As the communication channel is wireless, the broadcast messages during group-key distribution may not reach all the nodes. In such scenarios, self-healing is used to obtain the missing information using a future broadcast. When the missed broadcast message is required in the current session itself, a node takes help of some neighbor node and use mutual-healing to get the required broadcast message. We first discussed a bilinear pairing based healing enabled group key broadcast protocol that gives performance improvement over the existing bilinear pairing based healing protocol. With the proposed bilinear pairing based healing protocol, the nodes are able to perform self-healing and mutual healing with reduced computation and storage overhead. The proposed mutual-healing protocol provides additional security features of secure localization and key confirmation as compared to the existing bilinear pairing based mutual-healing protocol. Although, the bilinear pairing based healing enabled group-key broadcast provides robust security, the pairing operations are costly for resource constrained sensor nodes. The pairing based approach involves public key setup, that incurs more storage overhead as well. Therefore, we proposed further improvement and presented a healing protocol that uses Chinese remainder theorem (CRT) based secret sharing for group key distribution. The proposed CRT based protocol provides authentication, resistance to impersonation and replay attacks along with secure self-healing and mutual healing. From the experimental and simulation results it is evident that the proposed bilinear pairing based healing protocol has significant performance boost over the

existing healing protocol, specifically during mutual-healing. With the proposed CRT based healing, the cost overhead in terms of communication, computation and storage is almost negligible when compared with the only existing pairing based mutual-healing protocol.

The resilience to node capture attack can be reduced with a robust key management scheme in place wherein the impact of node capture attack can be optimized. However, the threat of node capture attack remains and it is vital to have timely detection of a victim of node capture attack. We, therefore, present an efficient and secure protocol of node capture detection in the next chapter.

CHAPTER 5

Node Capture Attack

Wireless sensor networks (WSNs) deployed for applications, such as battle-field surveillance and forest fire detection, work in hostile environments. In such scenarios, the sensor nodes are left unattended after deployment and therefore, are vulnerable to various types of physical attacks including node capture attack. Node capture attack allows an adversary to physically capture one or more nodes, reprogram and redeploy the nodes in the network to benefit as an insider attacker. The secure key management can help in increasing the resilience to node capture attack, however, secure and efficient detection of node capture attack is vital to the security of a WSN. In this chapter, a program integrity verification protocol is discussed. The protocol compares the program memory content of the node before and after capture. This verification for a node capture suspect is carried out by its respective cluster head that is equipped with trusted platform module (TPM). The proposed TPM enabled program integrity verification (TPIV) protocol uses dynamically computed hash based key and pseudo random function for successfully detecting the node capture attack. With the TPIV protocol, the probabilities of a victim of node capture attack eluding the PIV and leaking the secret of any non-captured node are negligible. TPIV protocol detects the node capture attack even in the presence of a strong adversary capable of putting additional memory to elude the PIV. The performance improvement in terms of low communication, computation and storage overhead as compared to the already existing protocols for program integrity verification is evident from the analytical comparisons and experimental results.

5.1 Introduction

In most of the real-life applications such as battle-field surveillance and forest fire detection using wireless sensor networks (WSNs), a group of sensor nodes are densely and randomly deployed and, are left unattended in hazardous conditions. The nodes collaboratively monitor events such as movement of enemy troops in battle-field and communicate with each other over wireless channel. The sensor nodes may directly communicate the observations to a central trusted powerful entity, called base station, or through the intermediate more resourceful nodes, called cluster heads. The wireless nature of the communication channel and the unattended deployment in difficult terrains leave a WSN vulnerable to various attacks including node capture. The inherent resource constraints of nodes restrict using traditional security solutions directly for WSN. Although during the last years, researchers have successfully addressed various aspects of WSN security such as secure key management [4], secure location and so on, the problem of node capture attack is a major concern.

Researchers have proposed protocols for attestation and program integrity verification (PIV) [112] [99] [25] [117] for detecting a victim of node capture attack. The software attestation protocol in [112] requires optimal program code and accurate time synchronization between the verifier and a prover. Soft tamper proofing proposed in [99] is vulnerable to impersonation and replay attacks. In addition, both protocols need centralized verifier. In [25], the verifier compromise reveals the program code of all the nodes. Both [99] and [25] does not address memory addition attack. In the case of utilization of specialized hardware (e.g. Trusted Platform Module (TPM) [122]) at all the sensor nodes, such as in hardware based protocol [117], cost overhead occurs when large sensor networks are considered. Recently, Zhao discussed about node capture attack in his work [139], however, that work only analyzes the q-composite scheme for key pre-distribution, that mitigates the node capture vulnerability in the neighbor discovery phase. In [92], the node replication detection attack is proposed that adversary can carry out after capturing the node. However, neither [139] nor [92] discuss about detecting node

capture. Clearly, the need to efficiently and securely detect a captured node is an excellent motivation to address this issue.

In the subsequent sections, various approaches to detect node capture attacks are discussed and then the TPM enabled program integrity verification (TPIV) protocol to efficiently and securely detecting the node capture attack is presented.

5.2 Identifying Node Capture by Monitoring

Monitoring based node capture attack is a classic approach for detection of a victim of node capture attack and has been widely discussed in the WSN security literature. Wong *et al.*in [57] had given a protocol for identifying malicious nodes in wireless sensor networks through detection of malicious message transmissions. The protocol compares the signal strength of a message with the geographical location of the sender and if it is incompatible, the transmitted message is considered to be suspicious and the information about the identified malicious node is disseminated in the network. This protocol assumes homogeneous and symmetric network with uniquely identifiable nodes, but it also requires the arrangement to be static and nodes to know their own geographical position. Liu *et al.* in [77] used various buffers such as *Arrival Time buffers* and *Transmission Time buffers* to store the arrival and transmission time of each message transmitted/received. If arrival time is not within the specified limits, base station is alerted and based on the decision of the base station, node identity of the suspect is put in *Compromised list* by other nodes. The scheme incurs additional storage overhead for maintaining arrival time and transmission time buffers. Also, it needs to check for each packet the arrival time validity and it is designed for static WSN. In [86], Mathews *et al*. looked into an anomaly-based intrusion detection system to detect compromised nodes using the event-driven characteristics of sensor networks to verify whether a node is sending packets in fixed time intervals. When the base station gets an alert about the abnormal behavior of some node, it verifies it by checking the difference in packet transmission times of that suspected node. The algorithm assumes network of stationary homogeneous nodes without clock syn-

chronizations amongst themselves and with no scope of new additions. Zhang, Yu and Ning [136] proposed an application-independent framework in which the sensors monitor the activities of the other nearby nodes assuming the existence of application-specific detection mechanism. It focuses on static sensor networks.
Conti *et al.* in [33] proposed two specific protocols that use node's mobility to detect node capture attacks in a completely distributed way. In the first proposal, Simple Distributed Detection (SDD), the attack is detected using only the information local to the nodes. The second solution, the Cooperative Distributed Detection (CDD), exploits node collaboration to improve the detection performance. According to the SDD protocol, if node A has listened to a transmission originated by node B (meeting occurred) at a certain time t, and later it does not re-meet node B for some long enough time, then node A can infer that node B has probably been captured. In the cooperative version of this protocol, two or more nodes monitor a node's presence in the network. They may share the meeting time of the monitored node with each other to reduce the false positive rate. This work was, however, at conceptual level. Later, they presented detailed algorithms both for simple and cooperative distributed detection [87], that leverage mobility and cooperation (CMC Protocol). The attacker, in their scheme [87], is assumed to know the detection protocol. It is an event-based protocol. Each node A tracks a set of nodes T_A. When a tracked node $B \in T_A$ meets node A, both the nodes execute CMC meeting algorithm through which they update the meeting time and set a time-out for each other. In case node B does not meet node A again within a specified time-out period (say λ) node A executes CMC-timeout() routine, which raises an alarm against node B, claiming it to be absent from the network and flood the whole network requesting revocation of the node. Any node X, that receives a message from node A, runs *CMC-receives()* module and checks the message type. If it is a usual message, then it simply sets the meeting time with node A, else if it is an *Alarm* then it checks, if the alarm is against itself, it floods the whole network claiming its presence, else if the accused node, say B, is not revoked, then the receiving node X sets revocation time out for node B. On expiry of revocation timeout, the node B is revoked. Two nodes A and B, may also cooperatively mon-

itor a given node *C* and update the meeting time with node *C* virtually without real meet with node *C*, based on the inputs from each other.

We observed the possibility of insider attacks in this approach. The tracking node may act maliciously and it might raise a false alarm against a node. Although the presence proving mechanism is there, we cannot ensure if it is the node which is actually present or some other node giving proxy. Also the time out and revoke timeout is set based on local clocks of the nodes thus we cannot completely rely on the timings for deciding if the node is actually captured. The scheme does not deal with the nodes that are captured and redeployed in the network.

Li, Song and Alam in [72] defined a Data Transmission Quality (DTQ) function (a measure of node's communication quality) to differentiate the legitimate nodes and compromised nodes. For legitimate nodes, the value of DTQ function remains constant or changes smoothly. However, if the node behaves suspiciously, this value keeps decreasing. Finally, the group members' voting is used to decide whether a suspicious node is actually compromised or not.

A weighted-trust evaluation (WTE) based scheme to detect compromised or misbehaved nodes in wireless sensor networks was proposed by Atakli *et al.* in [9]. The scheme considers three levels in the network - at the lowest level the sensor nodes are there, then the middle level has forwarding nodes and the top level consists of access point(s) (APs) or base station(s). Each forwarding node heads a cluster of sensor nodes that send the sensed data to the forwarding node. A forwarding node assigns trust values to each of the nodes in its cluster. If a node sends meaningless/wrong information, it implies that a node has been compromised, so the trust level of the node is lowered by its forwarding node. Deciding on the trust value and recalculating the trust depends on the application and does not seem to be a trivial task. The proposal assumes that the compromised nodes would always behave abnormally.

A new approach to detect the attack at the first stage is proposed by Ding *et al.* [39]. Their detection is based upon the discovery of missing nodes and malfunctioning of nodes due to the physical capture. Each node is assigned a guardian node. A node needs to send a *Hello* message to the guardian node at the periodic

intervals with least possible radio power level. When guardian node does not receive a *Hello* message for three consecutive time intervals, it sends out two *Are you there* messages consecutively and if no response is received, the guardian node will broadcast a *Captured* message to neighbors with its maximum radio power. This *Captured* message will be forwarded by its neighbors to the entire network. Any node that receive this message will add the announced node to its list of known captured nodes and will not forward or aggregate the data packets from the captured node. If the keys are used in sensor network security scheme; the current keys are replaced with alternative keys. Furthermore, a captured node can also broadcast an *IMC (I Am Captured)* message to neighbors. For inactive nodes, the nodes that are put into hibernation or sleep, the frequency of sending out *Hello* message will be much lower. Lin [76] addressed the node compromise problem in the first stage of physical capture in a different way. A new couple-based scheme is presented in [76] to detect the node compromise attack in an early stage. According to this, once the sensors are deployed, they build couples in an ad-hoc pattern. Now, the nodes in a couple monitor each other to detect any node compromise attempt. It is assumed that each sensor node can detect being connected by a programming board when an adversary launches the physical node compromise attack. When a node detects itself to be connected to a programming board, it sends the signal to its spouse and the spouse informs the base station and its neighbors of the compromise. In case, the adversary shuts down the node after capture, then also the spouse will come to know, as it does not receive any signal for a defined time period and thus suspects the compromise. The scheme uses Elliptic Curve Cryptography (ECC) based keys and designed for static network wherein the nodes are stationary after deployment.

Couple-based monitoring can keep track of the absence of a node from the network for more than a specified threshold period. However, setting the optimal threshold to decide if the node is captured has practical difficulties in multi-hop wireless sensor networks. To decide on the threshold value for absent time to consider a node as captured, Ho in [52] suggested using the Sequential Probability Ratio Test (SPRT). SPRT is a dynamic threshold scheme for setting up the

threshold that changes dynamically according to the absence time duration measured for a node. This dynamic change in threshold helps in minimizing the false positive and negative rates of node capture detection. Furthermore, SPRT not only achieves the low rates of false positive/negative, it is capable of reaching the decision with a few samples. However, this scheme is designed for static sensor networks, wherein the nodes divide the time period T_x of monitoring nodes into n time slots and then checks the presence of the monitored node in each slot and uses SPRT to decide if the node is present/absent during the time period T_x.
If the monitoring nodes collude, they can give the proxy for the absent node. The adversary could have captured the node, and would be able to reprogram and redeploy the node in the network for launching various insider attacks. It is, therefore, necessary to periodically check the integrity of the sensor node program.

5.3 Software and Hardware Attestation

Seshadri *et al.* [112] created a SoftWare-based ATTestation technique (SWATT), a code attestation algorithm that is executed solely through software means. Their technique was designed with the intention of creating a method to externally verify the code running on embedded devices. A trusted verifier is the key component in achieving this goal of their algorithm. The malicious node will contain at least one line of code that is different from the expected code running on normal sensors. The verifier has a copy of the memory contents residing in the nodes. The verifier sends a challenge to the node, which the challenged node uses as the input to a pseudo-random generator to create random memory addresses. A checksum is performed in the device on each of these memory addresses. The verifier runs the same verification procedure locally to compute the expected value of the check-sum. This expected value is compared to the value returned by the node in question. This scheme requires the trusted verifier (in case of WSN, the base station) to continuously be in communication with the nodes. A node must have memory-content-verification procedure, which itself can be tempered with by the attacker. They also proposed a protocol called SCUBA [111] to recover the sen-

sor nodes after compromise, which uses two-authenticated channel between a node and the base station. Hash chain is generated by the node as a function of the indisputable code execution (ICE) check-sum, therefore, malicious node can not pre-compute the hash chain to save time and forge the ICE check-sum. In this scheme also, the expected value of the check-sum computed locally at the verifier end is compared with the value returned by the node in question. The node and the trusted verifier requires continuous communication and the verification procedure loaded in node memory itself can be corrupted, if the node is compromised. Software attestation based techniques such as [112] and [111] are not suitable for multi-hop WSNs, which require mutual authentication between the prover and the verification authority. Also, they depend upon the accurate time measurement as well as optimal coding of the sensor node program to decide whether the valid code is currently running on the node. In [30], Choi *et al.* proposed improvement over SWATT with proactive code verification protocol. Whenever a verification is requested, the protocol cleans the memory space of the target node where malicious code can reside. Each node shares a pre-deployed pair-wise key with the base station. After, node deployment, verification code V and firmware code C stay in SRAM or flash memory. To start the verification protocol, base station sends a request message to the target node along with an encrypted nonce. Target node decrypts the nonce and uses it as a seed to generate random numbers. These random numbers are fed into an algorithm that fills the empty space of the node. The node responds back with the hashed memory content encrypted with the pair-wise key between base station. The base station checks the received hash and if it is not as expected, the node is considered to be compromised. In [30], at node end, the fill memory algorithm including random number generations and multiple hash computations are involved. The node needs to perform a symmetric decryption after the request phase and encryption before response. The request-response communication does not provide message authentication. Though, the values are encrypted, the receiver side has no way to ensure if the encrypted value itself is tampered.

Researchers have also proposed hardware attestation protocols for detecting node

capture attack. These protocols use Trusted Platform Module (TPM). Kraus *et al.* in [67] proposed two hardware attestation protocols for hierarchical WSNs organized in clusters. In [67], the WSN has two types of nodes - sensor nodes with limited resources and more resourceful cluster heads equipped with trusted platform module (TPM). The trustworthiness of a cluster head can be validated by more than one sensors simultaneously in regular intervals using Periodic Broadcast Attestation Protocol (PBAP). One way hash chains adapted from μTESLA [100] are used for authentication. The protocol binds a one-way hash chain to the platform configuration of a cluster head using the sealing function of the TPM. The sensor nodes in the cluster can verify the values of hash chains, released periodically by the cluster head. Only if the platform configuration of the cluster head is not altered, the validation of hash chain value succeeds and trustworthiness of the cluster head is proved. The second protocol named as Individual Attestation Protocol (IAP) allows the validation of the trustworthiness of a cluster head by a sensor node or sink at any time. A sensor node or a sink sends the data along with its identity and a nonce to cluster head encrypted with a pair-wise key shared between cluster head and the node/sink. This key is sealed within the TPM of cluster head. If the cluster head is successful in unsealing the key, decrypt the nonce and data and respond back with the same nonce sent, the cluster head is considered to be trustworthy. The hardware attestation protocols in [67] can also be in the scenarios where many mobile TPM-enabled sinks are deployed in insecure locations and a network verifier needs to verify the trustworthiness of the data received from these sinks. The protocols in [67] can be used to prove whether the cluster is trustworthy or not, it can not prove the untrustworthiness of a cluster head. Moreover, both the protocols are used for validating the cluster head, the verification of the integrity of program stored in a sensor node is not addressed. In [117], Tan *et al.* suggested using TPM at all the sensor nodes using which a node can verify the integrity of any peer node. Hardware attestation with all nodes enabled with TPM proposed in [117] do consider the node program integrity verification. However, for large sensor networks, equipping all the nodes with TPM will result in increased network cost. The scheme needs a

node to interact with base station for each verification, which is a communication overhead. Yang *et al.* in [131] suggest using TPM enabled cluster heads for node compromise detection. However, the authors have only mentioned monitoring of nodes by cluster heads for compromise detection. In their scheme the sensor node needs to perform few symmetric encryption/decryption and a public key encryption to establish key with the cluster head. Khiabani *et al.*[60] proposed Unified Trust Model (UTM) in which a node's trustworthiness is calculated on the basis of its history, recommendation, context, and platform attestation. A node A can evaluate the trustworthiness of another node B by checking the history of past interactions H_{AB}, recommendations of other entities in the same environment as R_1 to R_n, communication context and platform configuration of TPM associated with node B. This scheme does not consider all nodes equipped with TPM, but incurs overhead on sensor nodes for computing all the trust parameters involved in the validation process. For resource constrained sensor node in a WSN, the endeavor is always to optimize the overhead on nodes, and therefore, a secure and efficient solution to deal with node capture detection is required.

5.4 Program Integrity Verification

Park and Shin [99] proposed a software solution tailored made for constrained sensor nodes that claims to prevent manipulation/ reverse engineering/ reprogramming of sensors not degrading normal sensor functions. The Software Tamper Proofing (STP) protocol in [99] is a Program Integrity Verification (PIV) protocol that works towards prevention of nodes rejoining the network after capture and redeployment. The protocol uses a trusted central authority, usually the base station, as Authentication Server (AS), cluster heads as PIV servers and the sensor nodes which need to verify the integrity of their programs through PIV servers (PIVS). The approach is based on Randomized Hash Function (RHF). The entire program on a sensor node is divided into program blocks. The digest of these blocks are kept with the PIVS. Server nodes have PIV codes (PIVC) to verify the integrity. Whenever a node newly joins the network or rejoins after a long ab-

sence, PIVS verifies the integrity of the nodes. The nodes can verify the authentication of the PIVS through the authentication server, before executing the PIVC with a specific PIVS. The STP protocol triggers infrequently and does not rely on self decryption or result checking and can be used with/without tamper-resistant hardware. However, the scheme requires base station to authorize the program integrity verification servers (PIVS) and also the communication between a PIVS and a node is done on public channel which may cause man-in-the-middle attack causing a valid node to fail the verification. The protocol considers the scenario after captured node rejoins the network, however, it does not discuss as to how the PIVS would decide if the node is redeployed. Chang and Shin [25] suggested using PIV with distributed authentication, wherein a set of PIVS authenticates the PIVS involved in a node's program integrity verification. The centralized authentication in STP is not appreciated as the structure is not consistent with the distributed structure of WSNs. Central entity may become a bottleneck for communication, reliability and security. The energy consumption of the nodes near base station will be more.

In the Distributed Authentication Protocol of PIVSs (DAPP) [25], all nodes and PIVSs share polynomial share based [17] pair-wise secret with each other. PIVSs can use this secret for the purpose of authenticating one another. Before deployment, the setup server randomly generates a symmetric bi-variate polynomial $f(x,y) = \sum_{i,j=0}^{t} a_{ij}x^i y^j$, and assigns the shares of these polynomials to all PIVSs and nodes in the network. Post-deployment, PIVS A with a share $f(A,y)$ and PIVS B with another share $f(B,y)$ can authenticate each other and establish a pair-wise key as $K_{AB} = f(A,B) = f(B,A)$.

Once the initialization and PIVS discovery is completed, sensor nodes joining the network require to get their program integrity verified. A sensor node can choose the nearest PIVS within its communication range and authenticates the chosen PIVS by asking the PIVS to present authentication tickets from other PIVSs. A sensor node E requests authentication from the PIVS A using its nonce N_E and message authentication code (MAC), $MAC_{K_{EA}}(N_E)$, that is computed on nonce with the help of the pair-wise key between the node and the verifying PIVS. The

verifying PIVS A in turn sends the request for authentication tickets to other PIVSs in the network. For requesting authenticating ticket to another PIVS B, the PIVS A sends the identity E and nonce N_E of the target node along with the authenticator $MAC_{K_{AB}}(E\|N_E)$ computed over E and N_E using the pair-wise key between the PIVS A and PIVS B. The responding PIVS B prepares the authentication ticket of the form (B, $MAC_{K_{BE}}(N_E)$), where K_{BE} is the pair-wise key between node E and PIVS B. PIVS B sends this authentication ticket along with node identity E and the authenticator $MAC_{K_{BA}}(E\|(B, MAC_{K_{BE}}(N_E)))$ to verifying PIVS A. PIVS A has to collect at least n_{auth} valid authentication tickets from other PIVSs. After collecting all required authentication tickets, the verifying PIVS A now prepares the response of authentication request from target node E. This response includes all the authentication tickets and the authenticator $MAC_{K_{AE}}(N_E + 1)$.

After validating the response and all the authentication tickets from n_{auth} authenticating PIVSs, the target node E accepts the PIVS A to execute the program integrity verification protocol with it. The node now checks for the latest version of the program integrity verification code (PIVC). The PIVS sends the new PIVC in response, if needed. Node writes the new PIVC into flash memory and requests the new hash key. Using the new hash key, the node computes the hash of its program code and requests PIVS for verification. The PIVS, on successful verification, registers the identity of validated node into its database and confirms the verification to the target node. Node can now start executing its application. The complete process of program integrity verification with DAPP is depicted in Figure 5.1. Although, the authors have discussed a parallel approach to monitor the compromise of PIVS and revocation of a compromised PIVS as a corrective measure, PIVS compromise is a threat to the overall integrity verification system. This scheme also does not talk about how to decide when the redeployment after capture of a node happens. The pair-wise key scheme [17] used between PIVS has security weaknesses, which are already discussed in Chapter 3. Both STP and DAPP protocols for program integrity verification can not handle the adversary that is capable of adding additional memory in the node.

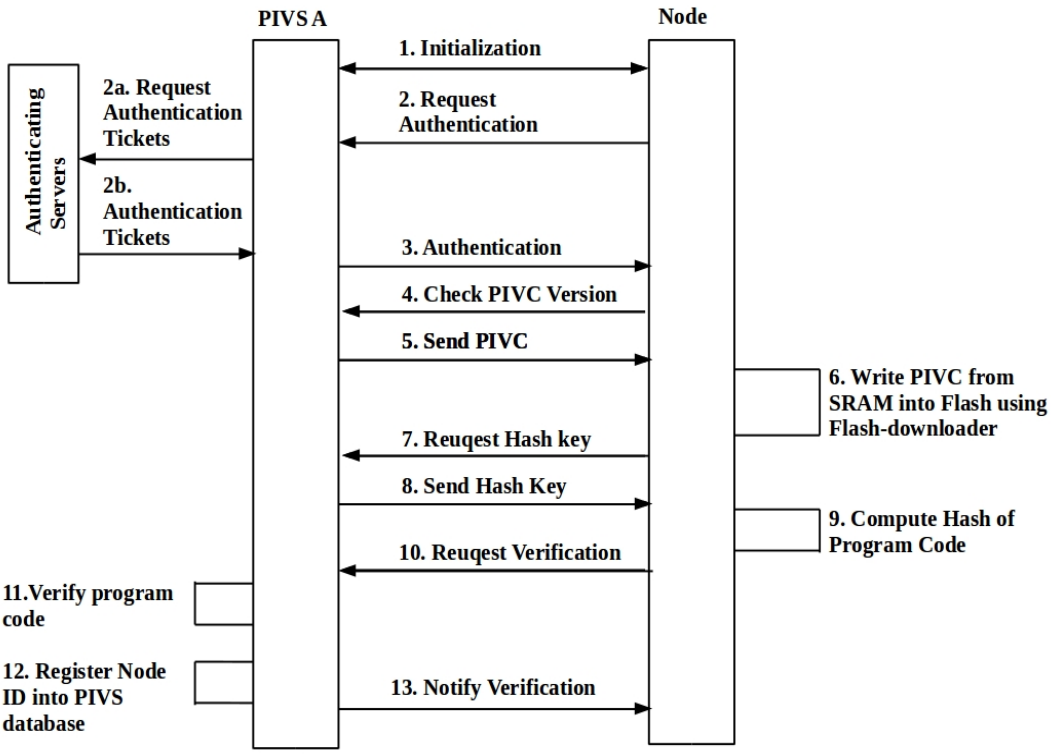


Figure 5.1: DAPP Protocol

In the subsequent section, we present the proposed trusted platform enabled program integrity verification (TPIV) protocol for detection of node capture attack.

5.5 Trusted Platform Module Enabled Program Integrity Verification (TPIV) Protocol for Node Capture Detection

In the proposed TPIV protocol, a cluster head equipped with TPM compares the program memory content of the node before and after capture to verify the integrity of the node program. A dynamically computed hash based key and pseudo random function are used for successfully detecting the node capture attack. TPIV protocol works even when additional memory is put in the captured node to elude the PIV.

5.5.1 Goals and Assumptions

The goal of our TPIV protocol is to detect node capture attack ensuring that:

- only an authorized verifier executes the PIV for detecting a node capture suspect;
- a victim node can not elude the PIV;
- a captured node does not reveal the secret of other non-captured nodes.

We assume that the adversary does not know the program memory contents of any non-captured node. The hash of program memory content of a node is the initial secret between a node and its cluster head. We also assume the sensor nodes having separate user/data and program memory.

Prior to deployment, base station keeps main application code, boot code and, the public functions in program memory. All TVSs and nodes are capable of computing a cryptographic Pseudo Random Function $PRF_k()$ using key $k \in Z_q^*$ for a large prime q and, a one way hash function $h()$. The PRF used is a keyed hash that is SHA-256 HMAC [88]. Unique random incompressible bit strings are filled in the free space of the program memory of each node. Any two nodes (say A and B) have random incompressible bit strings S_A and S_B ($S_A \neq S_B$, $\{S_A, S_B\} \in Z_p^*$ for some large prime $p \geq q$ (q is a large prime defining the size of the secret)) respectively in their free spaces. Despite the same application code, the overall program memory content of each node would be different due to the uniqueness of random bit strings. We assume that the application code always leaves some reasonable size of free space in the program memory and whenever the application code changes due to software update, free space is adjusted in accordance with the updated code size. In each TVS, the base station securely copies the free space contents of each node along with one common copy of the remaining program memory content. The node program contents stored at a TVS are sealed using the initial platform configuration and the public key of TPM embedded in that TVS.

Post deployment, the nodes associate themselves with a cluster head nearest in

their transmission range to form a cluster. An active adversary is assumed to be present in the network and we discuss his capabilities in the security analysis section.

5.5.2 TPIV Setup and Monitoring

In the proposed setup, the sensing nodes are vulnerable to node capture attack. The trusted base station interacts with the outside world on behalf of the WSN and keeps overall control of the network. Verification is performed by a small number of resourceful TPM enabled cluster heads referred as TVS (TPM enabled Verification Server). Each TVS manages a group of resource constrained normal sensor nodes. Base station assigns a unique identity ID_A to each node A and CH_V to each TVS V, prior to deployment. The TPM attached to each TVS V has a unique non-migratable public-private key pair (Pub_V, Pri_V) always residing in protected storage within TPM. At time t=0, TVS V switches on and PCRs of associated TPM stores the initial platform configuration (PC) PC_V^0. Using PC_V^0 and Pub_V, the program memory content P_A of node A is sealed within the TPM associated with V using $PSeal()$ function as:

$$\mathcal{P}_{Asealed} = PSeal(PC_V^0, Pub_V, P_A)$$

This sealed content $\mathcal{P}_{Asealed}$ can be unsealed later at time t using PC_V^t, the PC of V at time t and, Pri_V (provided the configuration $PC_V^t = PC_V^0$) as:

$$P_A = PUnseal(PC_V^t, Pri_V, \mathcal{P}_{Asealed})$$

If $PC_V^t \neq PC_V^0$, the unsealing fails and the sealed contents become inaccessible to TVS V.

A cluster head monitors the transmission of the nodes within its cluster. For each node A, the last transmission heard time t_A^{last} is recorded. At time $t_A^{new} > t_A^{last}$, when next transmission is heard from A, if the time interval between t_A^{new} and t_A^{last} is more than a set threshold time T (t_A^{new} - $t_A^{last} \geq T$), cluster head proceeds to perform PIV for the suspect node A.

5.5.3 Authentication and Code Verification

A TVS carries out the PIV for a node suspected to be a victim of node capture attack. Before the node presents itself for such verification, it ensures the validity of the verifying TVS. The two step protocol for authentication and code verification is given in Figure 5.2.

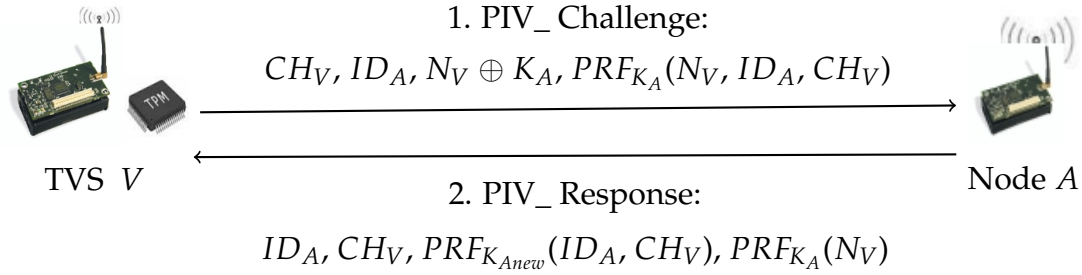


Figure 5.2: Authentication and Code Verification Protocol

A TVS V, suspecting a node A, initiates the authentication and code verification protocol. TVS V, having the current platform configuration same as initial platform configuration, retrieves the program memory content P_A by unsealing and computes the key shared with node A as $K_A = h(P_A)$. TVS V picks a nonce N_V randomly and sends the *PIV_Challenge* message to node A as CH_V, ID_A, $N_V \oplus K_A$, $PRF_{K_A}(N_V, ID_A, CH_V)$ (Figure 5.2: step 1).

On receiving *PIV_Challenge*, node A extracts the nonce N_V as $(N_V \oplus K_A) \oplus K_A$ and computes $PRF_{K_A}(N_V, ID_A, CH_V)$. If computed PRF value matches with received PRF value in *PIV_Challenge*, node A authenticates the challenger TVS V. Node A, picks N_V extracted from the challenge and computes new key K_{Anew} = h(P_{Acurr}, N_V, ID_A, CH_V) using its current program memory content P_{Acurr}. Node A sends the *PIV_Response* message to TVS V in response to PIV_challenge as $ID_A, CH_V, PRF_{K_{Anew}}(ID_A, CH_V), PRF_{K_A}(N_V)$ (Figure 5.2: step 2) . Since the verification depends on the computation of new key K_{Anew} as expected, node A sends $PRF_{K_A}(N_V)$ to assure the TVS of correct nonce received. When TVS V receives *PIV_Response*, it first verifies the $PRF_{K_A}(N_V)$ and then computes new key at its end as $K_V = h(P_A, N_V, ID_A, CH_V)$. TVS V verifies PRF value $PRF_{K_V}(ID_A, CH_V)$.

If the value of P_{Acurr} used to compute K_{Anew} is same as the original value P_A stored at TVS V i.e. program code for node A is unaltered, the PIV succeeds. TVS V considers node A to be authenticated and untempered. Otherwise, node A is treated as captured and revocation follows.

At the end, node A and TVS V delete nonce and keep the new secret $K_{A_{new}}$ (= K_V). When node A leaves the cluster monitored by TVS V, node A informs the move to TVS V who informs this move to all the cluster heads and to all the nodes within its own cluster. The new TVS W, whose cluster is joined by node A intimates this new join to old TVS V who secretly shares the key K_V with W to communicate with node A.

5.5.4 Security Strengths

The TPIV protocol detects a victim of node capture with high probability and ensures that only an authorized verifier can execute the program integrity verification of victim node. Secrecy of non-captured nodes is maintained by not allowing a captured node to reveal the secret of any non-captured node. The TPIV protocol also ensures security against replay and impersonation attacks. The security analysis is carried out against the adversary model as given below:

Adversary Model. The existence of an active adversary is considered that is capable of capturing a node physically and steal its secret information. By impersonating a valid node, the adversary can take part in the network operations. Adversary is also capable of reprogramming and redeploying the captured nodes in order to launch various insider attacks such as sybil attack and selective forwarding. The strong adversary is capable of putting additional memory in the node. We further assume that the adversary can not remain present in the network all the time due to its deployment in the hostile terrains.

We analyze the protocol against the security goals and give the security proofs against each security claim in the form of theorems. The definitions used in the proofs are as below:

Definition 5.1: If $h(.)$ is a one-way cryptographic hash-function with collusion re-

sistance defined as $h\{0,1\}^m \rightarrow h\{0,1\}^k$, then the probability of hash collision, i.e. finding a $x^{'} \neq x$, such that $h(x) = h(x^{'})$ for a given a x and $h(x)$, is:

$$Pr\textbf{[(}h(x) \textbf{= } h(x^{'})\textbf{, for } x^{'} \neq x\textbf{)]} \textbf{= 1 -} (1 - \tfrac{1}{2^k})^{m-1}$$

Definition 5.2: If p_{succ} is the probability of a node successfully receiving a message from another node within its communication range, the probability of a verifier successfully receiving the response from challenged node is given as:

$$Pr[TVS\ V \xleftarrow{PIV\_Response} Node\ A] = \begin{cases} p_{succ}, & \text{if TVS is authorized verifier} \\ 0, & \text{otherwise} \end{cases}$$

Here, $p_{succ} = \sum_{j=0}^{e^c} \binom{k}{j}.(1-e)^{k-j}.e^j$, on a channel with bit error rate e, for k out of maximum e_c number of bits encoded.

Definition 5.3: If s_1 is a binary string of length l, then the maximum probability $p_{s_1 s_2}$ of finding another string s_2 having hamming distance at least 1 from s_1 is $\frac{l}{2^l}$

Definition 5.4: **EludePIV(.)** is a boolean function that determines if a captured re-programmed node eludes the PIV:

Algorithm 5.1 Elude Program Integrity Verification

Input : P_A, P_{Anew}, CH_V, Id_A, N_V
- P_A - Original Program Memory Content of node A
- P_{Anew} - Program Memory Content of A after reprogramming
- CH_V - Unique identity of TVS V
- Id_A - Unique identity of node A
- N_V - Nonce received by node A from server V

Output : TRUE/FALSE

```
1: procedure ELUDEPIV(P_A, P_Anew, CH_V, Id_A, N_V)
2:     if h(P_A, N_V, CH_V, Id_A) = h(P_Anew, N_V, CH_V, Id_A) then
3:         return TRUE
4:     else
5:         return FALSE
6:     end if
7: end procedure
```

The following security strengths are identified through the security analysis:

5.5.4.1 High Probability of Node Capture Detection

Theorem 5.1. *The Pr[EludePIV(P_A, P_{Anew}, CH_V, Id_A, N_V)] = TRUE is (1 - $(1 - \frac{1}{2^k})^{m-1}$) for $m = |P_A||N_V||CH_V||Id_A|$ and $k = |h(P_A, N_V, CH_V, ID_A)|$.*

Proof. We prove that a victim of node capture attack can not elude the PIV. We trace through the time line for an attacker carrying out the node capture attack (Refer Figure 5.3).

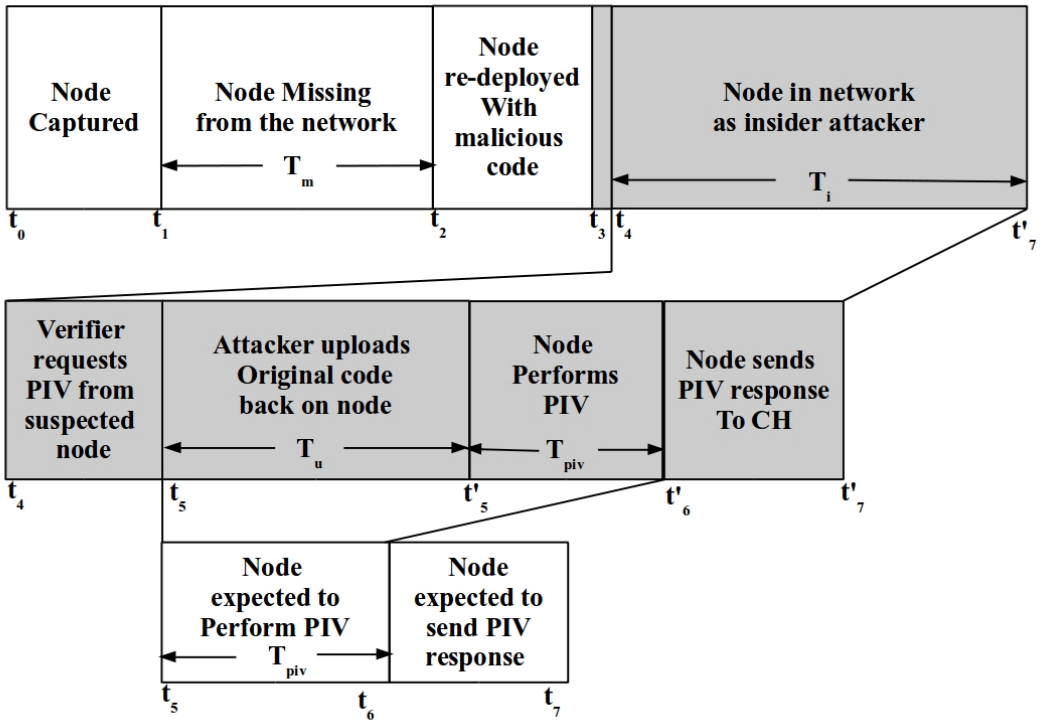


Figure 5.3: Attacker Time Line

After capturing node A, an adversary had access to secret K_A (i.e. $h(P_A)$). When the attacker reprograms this node, the program memory P_A changes to P_{Anew}. Since the free space in the memory is filled by an incompressible random bit string, an adversary can not insert code pieces while keeping the original program intact [5]. When the attacker puts additional memory and keeps the original program code there, at any point of time, the attacker has either P_A or P_{Anew} as program memory. Therefore:

Attacker(K_A)	... (5.1.1)
Attacker(N_V)	... (5.1.2)
(as Attacker($N_V \oplus K_A$) $\wedge$ Attacker(K_A), $N_V = (N_V \oplus K_A) \oplus K_A$)	
Attacker(P_A) $\wedge$ Not Attacker(P_{Anew})	... (5.1.3)
Attacker(P_{Anew}) $\wedge$ Not Attacker(P_A)	... (5.1.4)

To elude the verification, victim node must swap out the malicious code P_{Anew}, and swap in the original code P_A in the program memory before starting PIV protocol. As per attack time line (Figure 5.3), node A starts with uploading memory with P_A and starts the PIV execution at time t'_5 (= $t_5 + T_u$) instead of t_5 as expected by TVS. Therefore, it sends "PIV_Response message" at time t'_6 (= $t'_5 + T_{piv}$) and not at t_6 and TVS receives the delayed response at t'_7 ($> t_7$). Although the time synchronization in TPIV protocol is not crucial, the implementation (with ATmega328 processor using Arduino Duemilanove controller board and ArduinoISP programmer) of TPIV protocol at node end shows that to upload 9.4 KB of code associated only with PIV protocol takes about 11.27 seconds (T_u) which itself is significantly more than the protocol execution time (T_{piv}) measured as 5.86 seconds. Thus, the victim does not have enough time to use the original code to elude the PIV while having malicious code to carry out insider attacks otherwise. Another possibility is the victim node A eluding the PIV even after reprogramming i.e. holding P_{Anew} as its program memory and, A is able to send a valid PIV_Response to TVS V. Let us track back the protocol execution:

Attacker(valid PIV_Response)

$\Rightarrow$ Attacker($PRF_{K_{Anew}}(ID_A, CH_V)$) $\wedge$ Attacker($PRF_{K_A}(N_V)$)

$\Rightarrow$ Attacker(K_{Anew}) $\wedge$ Attacker(K_A) $\wedge$ Attacker(N_V)

$\Rightarrow$ Attacker(K_{Anew}) $\wedge$ TRUE $\wedge$ TRUE
[using assertions 5.1.1 and 5.1.2 respectively]

$\Rightarrow$ Attacker($h(P_{Anew}, N_V, ID_A, CH_V) = h(P_A, N_V, ID_A, CH_V)$) $\vee$
(Attacker(P_{Anew}) $\wedge$ Attacker(P_A))

$\Rightarrow$ TRUE (With probability (1 - $(1 - \frac{1}{2^k})^{m-1}$) $\vee$ FALSE

[using Definition 5.1 and assertions 5.1.3/5.1.4]

$\Rightarrow$ Pr[EludePIV($P_A, P_{Anew}, CH_V, Id_A, N_V$)] = TRUE is (1 - $(1 - \frac{1}{2^k})^{m-1}$)

[by Definition 5.4] □

5.5.4.2 Node Capture Detection by Authorized Verifier

Theorem 5.2. *The probability of an unauthorized verifier executing the PIV for a detecting a node capture attack is hash-collision probability (1 - $(1 - \frac{1}{2^k})^{m-1}$), where m and k are the input and output sizes of hash, respectively.*

Proof. Consider a TVS V compromised by an attacker. Being TPM enabled, any change in TVS V program changes its platform configuration. Now, TVS V sends the "PIV_challenge" to node A in order to execute PIV. Thus:

Not Attacker(PC_V^0)	... (5.2.1)
Attacker(N_V)	... (5.2.2)
(as attacker sending "PIV_challenge" message)	

If verifier is successful in executing the PIV for node A, the verifier must receive the "PIV_Response" from node as reply to "PIV_challenge". Let us trace back the protocol execution:

Attacker(PIV_Response)

$\Rightarrow$ Attacker($PRF_{K_{Anew}}(ID_A, CH_V)$) $\wedge$ Attacker($PRF_{K_A}(N_V)$)

$\Rightarrow$ Attacker(K_A) $\wedge$ Attacker(N_V)

$\Rightarrow$ Attacker(K_A) $\wedge$ TRUE [using assertion 2.2]

$\Rightarrow$ Attacker(P_A) (since $K_A = h(P_A)$) $\vee$ Attacker(h' (= $h(P_A)$)

$\Rightarrow$ Attacker(PC_V^t(=PC_V^0)) $\vee$ TRUE (With probability(1- $(1 - \frac{1}{2^k})^{m-1}$)

[using Definition 5.1]

$\Rightarrow$ FALSE [using assertion 5.2.1] $\vee$ TRUE (With probability (1 - $(1 - \frac{1}{2^k})^{m-1}$)

$\Rightarrow$ TRUE (With probability (1 - $(1 - \frac{1}{2^k})^{m-1}$)

Thus, even if an unauthorized verifier receives a message from a node with probability p_{succ}, the probability of the unauthorized verifier receiving "PIV_Response" from challenged node i.e. the probability of unauthorized verifier executing PIV

for a node is equivalent to the hash-collision probability.

The protocol does not allow an attacker to keep the hash of the program memory content pre-computed to be used for verification, as the nonce supplied by TVS is used for each verification round. For both Theorems 5.1 and 5.2, with our implementation with $m = 2^{18}$ bits (for program memory size of 32 KB) and $k = 256$ bits (secret size), the probability would be (1 - $(1 - \frac{1}{2^{256}})^{2^{18}-1}$) that is negligible. □

5.5.4.3 Secrecy of Non-captured Nodes

Theorem 5.3. *Given the capture of a node A, the probability of an attacker obtaining the secret of another node B is Pr[Attacker(K_B)/K_A] = $\frac{l}{2^l}$, where l is the size of the random bit string in the free space of a node.*

Proof. From the system model, we know that the program memory contents of any two nodes A and B are different because they have unique random bit strings S_A and S_B in their respective free spaces. Therefore, for two nodes A and B, the free space contents S_A and S_B (assuming the size of strings l bits each) differ by at least one bit. When the adversary captures node A, he knows the entire program memory content P_A and thus the secret $K_A = h(P_A) \in Z_q^*$ of node A.

Suppose an adversary has captured a node A and, the secret K_B of node B is available to the adversary:

Attacker(K_B)

$\Rightarrow$ Attacker($K_B = (K_A)$) $\vee$ Attacker($h(P_B)$)

$\Rightarrow$ Attacker($h(P_B) = (h(P_A))$) $\vee$ Attacker(P_B)

$\Rightarrow$ Attacker($P_A = P_B$)

$\Rightarrow$ Attacker($S_A = S_B$)

$\Rightarrow$ TRUE (with probability $\frac{l}{2^l}$) [by Definition 5.3]

For free space bit string size of 256 bits, the probability Pr[Attacker(K_B)/K_A] = $\frac{256}{2^{256}}$ = $2.21 * 10^{-75}$. □

5.5.4.4 Comparing TPIV with Existing Schemes

Table 5.1 compares the security features of proposed TPIV protocol with some of the existing proposals. We observed that unlike the SWATT [112] and STP [99] protocols, the TPIV protocol does not require accurate time synchronization between verifier and prover. While the integrity of challenge and response messages is not protected in STP and TRAP [117], the SWATT, STP and TRAP protocols require base station to be involved in each verification. The proposed TPIV protocol provides mutual authentication as opposed to STP. The DAPP protocol provides only *k*-collusion resistant (for *k*-degree polynomial used for key) against captured node revealing the secret of any non-captured node, while TPIV resists the key leakage of non-captured node with very high probability. The TPIV protocol is also capable of detecting the node capture attack with very high probability even when additional memory is put in the captured node.

Features ⇓ Scheme ⇒	SWATT [112]	STP [99]	DAPP [25]	TRAP [117]	TPIV [5]
no accurate time synchronization	×	×	✓	✓	✓
challenge-response message integrity protected	✓	×	✓	×	✓
base station only involved prior to deployment	×	×	✓	×	✓
mutual authentication between verifier-prover	✓	×	✓	✓	✓
captured node does not reveal valid node secrets	✓	✓	*	✓	✓
protected against node memory addition	✓	×	×	✓	✓

**k*- collusion resistant for k-degree polynomial used for key

Table 5.1: Comparing Security Features of Node Capture Detection Protocols

5.5.5 Efficiency and Experimental Results

5.5.5.1 Analytical Comparison

Table 5.2 captures the performance of the proposed TPIV protocol in terms of storage, computation and communication overhead as compared with existing protocols.

Features ⇒ Protocol ⇓	Prover or Verifier	Storage overhead (bits)	Computation overhead	Communication overhead (bits)	
				Transmit	Receive
DAPP [25]	Verifier	(k+s) log q	(N_{auth} + s +(s + 1) T_p	(3 N_{auth} + 8) T_s + 7) log q	(4 N_{auth} + 8) log q
	Prover	(k+N_{auth} +2)log q	(N_{auth}+7) T_s + (N_{auth} + 1) T_p	10 log q	(2 N_{auth} + 8) log q
TRAP [117]	Verifier	3 log q	T_t+ 3 T_s + T_p	3 log q	3 log q
	Prover	log q	2 T_t + 2 T_s	2 log q	log q
TPIV [5]	Verifier	log q	T_t + 4 T_s	4 log q	4 log q
	Prover	log q	5 T_s	4 log q	4 log q

Table 5.2: Performance Comparison of Node Capture Detection Protocols

For computation overhead, we only consider the time incurred in three main categories of operations namely, public key, symmetric key and heavy TPM operations such as TPM_Unseal and TPM_Sign, ignoring the light weight operations such as XOR and TPM_Read. In the performance comparison Table 5.2, timings required to perform a public key operation, symmetric key operation and TPM Unseal/Sign operations are depicted as T_p, T_s and T_t respectively. Size of secret log q is given by a large prime number q. For DAPP protocol [25], k is the degree of polynomial used for generating key, s is total PIVSs in network and N_{auth} is number of authentication tickets needed by a PIVS. With this protocol, we achieved an overall performance and security improvement with the TPM enabled verifiers as compared to the software based scheme such as [25]. At the same time, we saved the cost to equip all the nodes with the TPM chip as in [117].

5.5.5.2 Improvement in Node Capture Detection Probability

In the proposed TPIV protocol, the verification process considers the security against node memory addition that is not provided in the existing DAPP scheme. The TRAP protocol needs base station for every verification round.

Suppose:

Probability of an adversary placing additional memory in node = p_m

Probability of unavailability of base station = p_b

Probability of detection of victim of node capture attack = p_d

and, $\delta \ll 1$

Table 5.3 gives the improvement in the detection probability with proposed TPIV protocol over existing DAPP and TRAP protocols. For example: Consider p_m =

DAPP	TRAP	TPIV	Improvement with TPIV over DAPP	Improvement with TPIV over TRAP
$p_m.\delta$	$p_m.(1\text{-}\delta).(p_b.\delta)$	$p_m.(1\text{-}\delta)$	$p_m.(1\text{-}2\delta)$	$p_m.(1\text{-}\delta).(1 - p_b.\delta)$

Table 5.3: Improvement in Capture Detection Probability with TPIV

0.8, p_b = 0.7 and δ = 0.01 then,

Improvement with TPIV over DAPP = 0.8 *(1-2*0.01) = 0.784 (**78.4%**)

Improvement with TPIV over TRAP = 0.8* (1 - 0.01)*(1-0.7*0.01) = 0.786 (**78.6%**)

5.5.5.3 Experimental Results

The performance of the existing software based DAPP [25] and TPIV protocol are further compared using experimental and simulation results.

The network is simulated with Castalia simulator [20] for a field of size 100 by 100 with uniformly distributed varied number of nodes communicating with 25 meters of radio range. We first simulated the TPIV protocol with varied number of TVSs against a set of nodes and observed that with 10 TVSs we are able to achieve the optimum performance (Refer Figure 5.4).

As seen from Figure 5.4, when the number of TVSs are increased from 5 to 10, the percentage increase of nodes served is high, however, when we increase the number of servers beyond 10, then either the percentage increase is very low or it decreases. Even with a network of 500 nodes, more than 20% nodes are served at a time with the help of 10 TVSs. In a practical scenario, this suffices to deal with the captured nodes effectively.

We implemented the protocols with ATmega328 processor using Arduino Duemilanove controller board and ArduinoISP programmer. The experimental results

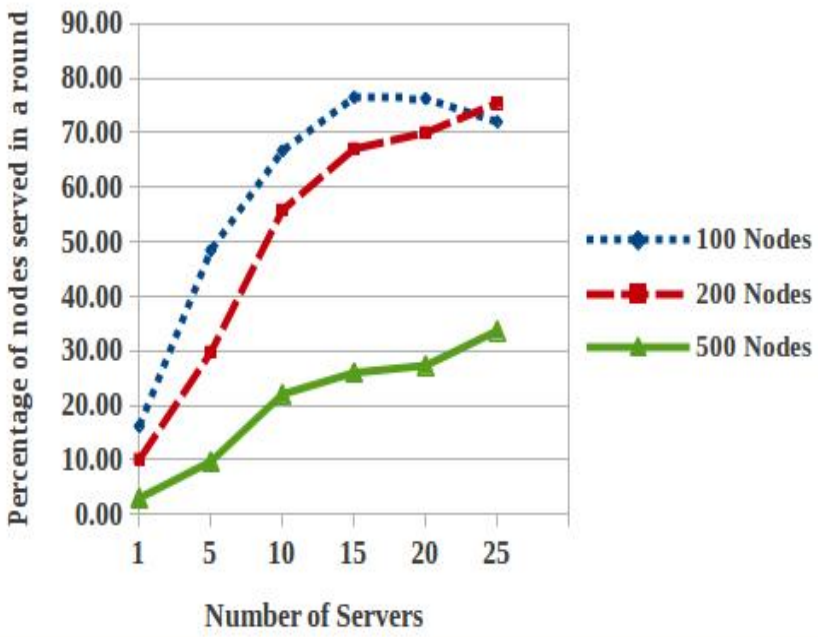


Figure 5.4: TVS Optimization

show that the computation cost incurred in executing the PIV with TPIV is significantly less than with DAPP [25] (Refer Figure 5.5) .

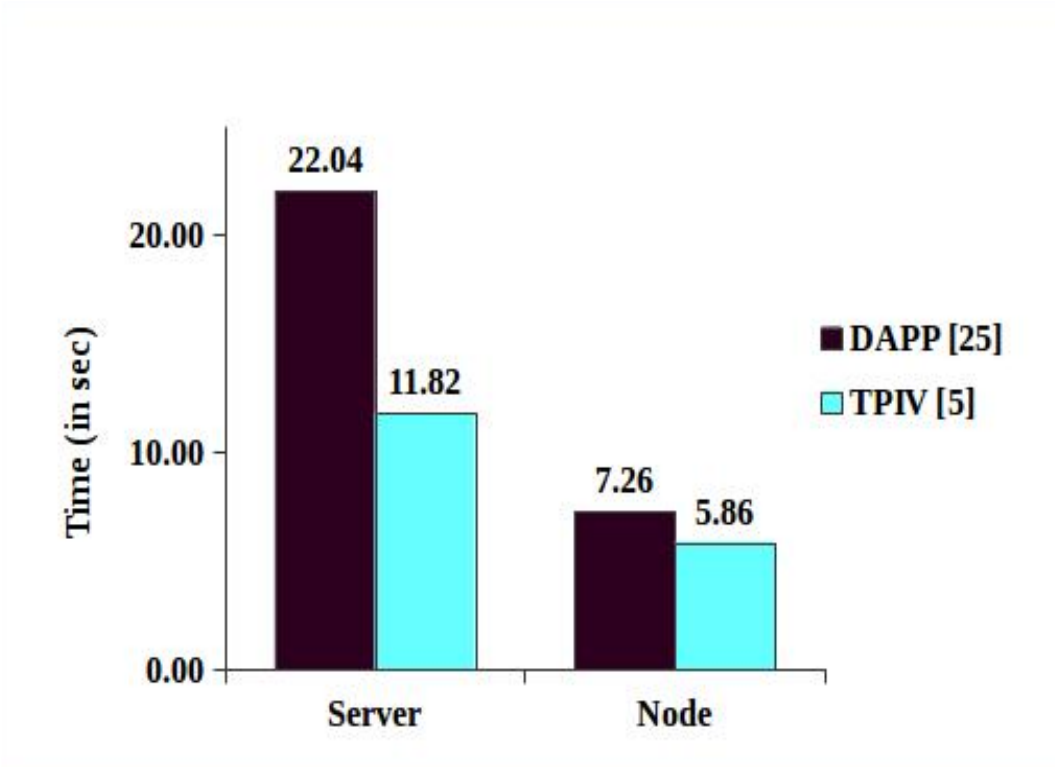


Figure 5.5: Comparison of Computation Cost for TPIV and DAPP

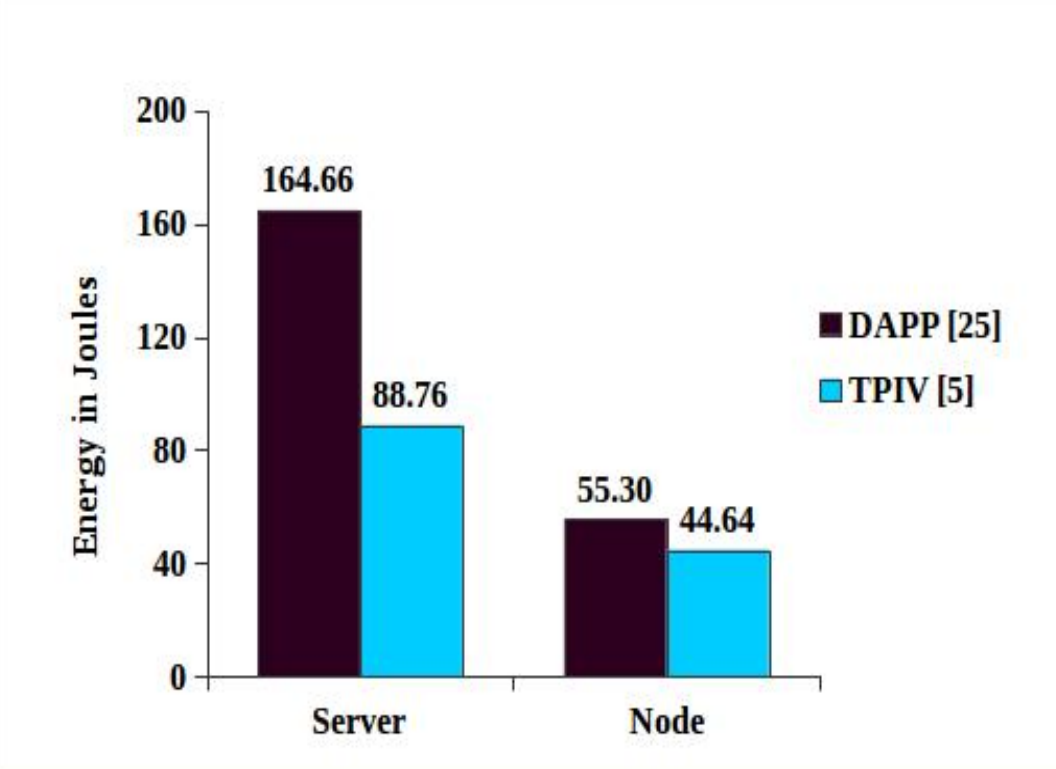
The simulation results also indicates the performance improvements with the TPIV protocol in terms of energy consumption and communication latency as seen in Figures 5.6(a) and 5.6(b).

As against the DAPP, the energy consumption with TPIV protocol drops down to nearly 50 % at verification server and at node end the reduction is almost 20 %. The communication latency is significantly low with TPIV.

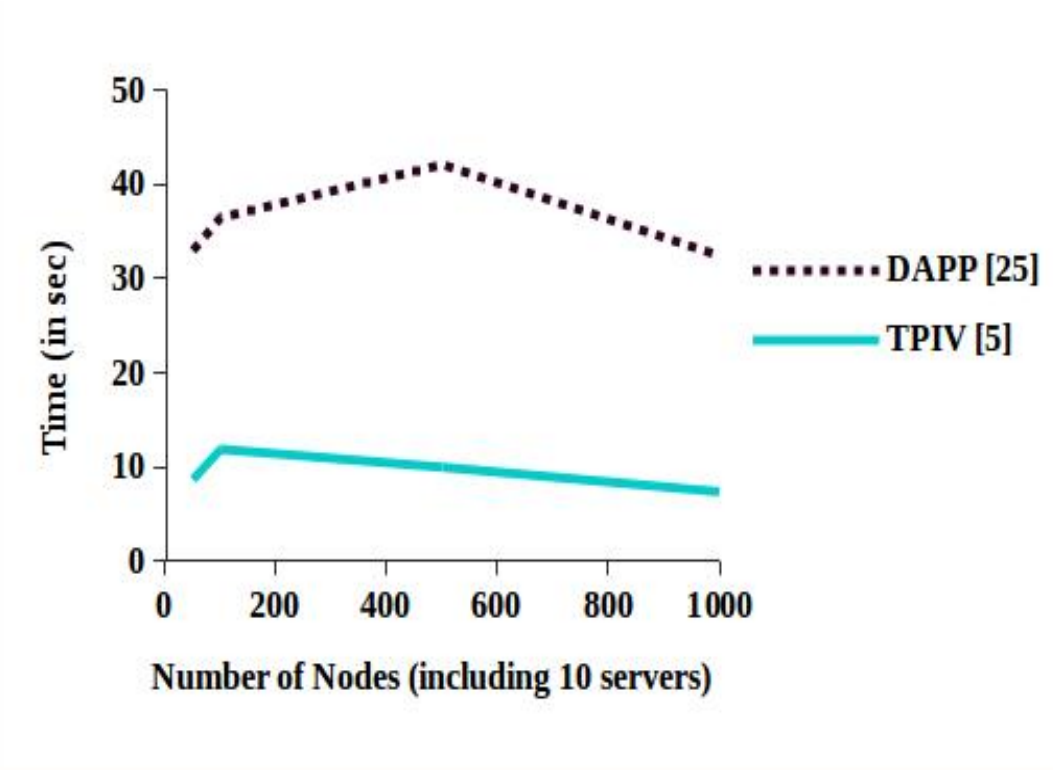
The reason for the reduction is, in DAPP protocol, four rounds of to and fro communication between the node and the server are required, whereas, TPIV protocol needs one round of communication. The sleep schedule of nodes affects the latency since TMAC as MAC protocol is used for simulation. For large number of nodes (say 1000), almost all the nodes remain awake for most of the time, as they are continuously getting signals from the neighboring nodes. In the TPIV protocol, an awake node can immediately respond back to the server challenge, while in DAPP, node has to wait for the server to collect authentication tickets from other servers. The reason for reduced computation cost is that, in DAPP, a node and the verifying server both compute polynomial based keys for all authentication ticket providing TIVSs. Also, MAC needs to be computed to verify the authentication tickets. In TPIV, a node computes only an un-keyed hash and 3 PRFs.

5.6 Conclusion

In this chapter, we discussed node capture detection in WSNs deployed in unattended harsh terrains. The objective of an adversary behind carrying out the node capture attack and various approaches proposed in the literature to model and detect the node capture are studied. A trusted platform module enabled program integrity verification protocol (TPIV) to detect the node capture attack in a distributed wireless sensor network setup ensures that only an authorized verifier can execute the verification. Through experimental results it is proved that the protocol does not allow a victim node to elude the verification process. The protocol prevents a captured node from revealing the secrets of other nodes. With the TPM enabled verifier sealing the program code of nodes, the protocol does



(a) Energy Consumption



(b) Communication Latency

Figure 5.6: Comparison of Energy Consumption and Communication Latency for TPIV and DAPP

not reveal node program code on verifier compromise. As evident from the performance analysis and experimental results, in comparison to the pure software based protocols, TPIV provides additional security with significant reduction in communication, computation and storage overhead on the nodes. The overall reduced cost of network deployment and maintenance is achieved by saving on the cost of having all the nodes enabled with TPMs.

On a successful detection of a node capture attack victim, the victim must be revoked from the network in order to avoid any further damage to the network. In the next chapter, we discuss a node revocation and key update protocol.

CHAPTER 6

Node Revocation and Key Update

Once a node is detected to be a victim of node capture attack with the help of program integrity verification as discussed in the previous chapter, the cluster head (PIVS) executes the node revocation and key update protocol to take out the victim from its cluster by updating the group session key. In this chapter, we discuss the node revocation and key update protocol that uses Chinese remainder theorem based secret sharing to secretly distribute the new session key to all non-compromised nodes in the cluster. The protocol works with significantly low overhead on resource constrained sensor nodes providing the security strengths in terms of secure node revocation, forward and backward secrecy and, resistance to collusion, replay and impersonation attacks.

6.1 Introduction

Once the node is detected to be a victim of node capture, it must be revoked from the network to avoid further harm to the network. Since the nodes are randomly deployed mostly in harsh environments, one of the efficient approaches to node revocation is to revoke the key used by the victim node so that it can not participate in any future network activities. To solve the vital issue of key revocation, there are different centralized and distributed approaches proposed in the literature [83] [45]. While the key revocation using centralized schemes require involvement of base station for each malicious node identification and revocation, the schemes suffer from a common threat of single point of failure. In the distributed schemes, the nodes are involved through voting mechanism to identify

and revoke malicious nodes, such schemes require the resource constrained nodes to take the overhead involved. The revocation accuracy is improved with the introduction of hybrid approaches that use the concept of grouping of nodes and take advantage of the features of both centralized and distributed approaches.

6.2 Centralized Approach to Node Revocation

In centralized revocation schemes, a central trusted authority, the base station, takes the responsibility of identifying a victim node and taking the revocation decisions. In [42], Eschenauer and Gilgor first proposed the concept of key revocation wherein, the central entity is a mobile controller node with large communication range. The central entity shares a pair-wise key with each node in the network before deployment. A signature key is generated by the controller node and shared secretly with each non-compromised node using pair-wise key between controller and the node. For the revocation, controller node broadcasts the message, containing all the key identifiers of the keys held by a compromised node, and signed by the signature key K_e. The non-compromised nodes verify the signature and remove the common compromised keys from their respective key rings. With this scheme, the central controller is severely overloaded, as for each revocation, it needs to first send the new signature key to each node individually. Even if one of the pair-wise key is compromised, the attacker can learn the signature key and thus can impersonate the controller node to send the fake broadcast. This basic scheme prompted the researchers to explore the key revocation in wireless sensor networks and this followed many centralized schemes for key revocation. The KeyRev scheme proposed in [126] revokes the compromised nodes by not allowing them to compute new session key k_j for session j from which the nodes are revoked. This scheme [126] uses an encryption key K_{enc} and a MAC key K_{mac} for secure communication in the network of n nodes. Both these keys are derived from the session key k_j. In this scheme, the setup server loads during setup phase, in each node u_i, for each session j, a personal secret obtained from a $2t$-degree masking polynomial. The setup server also selects a t-degree

polynomial $A_j(x)$ for each session j. For a session, a node u_i has to compute two polynomials $P_j(Id_i)$ and $Q_j(Id_i)$ received as $P_j(x)$ and $Q_j(x)$ in the broadcast message. These polynomials are obtained based on the revocation list $R_j = \{r_1, r_2, \ldots, r_w\}$ ($w \leq t$) of revoked nodes. Later, Wang *et al.* extended the KeyRev scheme and proposed mKeyRev [127] scheme in which m ($\ll n$) base stations are used and these base stations are distributed in such a way that each node has one-hop access to at least one base station.

Although, KeyRev and mKeyRev schemes prevent the impersonation attack as the encryption and MAC keys are bound to session key that is being updated after each session, there exists a vulnerability of same pattern attack due to fixed size of session interval, wherein the new key is distributed only when the new session begins. An attacker may use the time between a compromise and the start of new session to launch same pattern of attack again. In [61], Park *et al.* proposed dynamic level session (DLS) scheme that is an improved version of KeyRev to prevent same pattern attack by providing dynamic level sessions. In DLS, base station detects the compromised node and updates the session key for next session. The scheme uses dynamic size of session by using a session time configuration function wherein the base station distributes new session key immediately after a node compromise is detected instead of waiting for the active session to get over. The DLS time has less duration than the time an attacker takes to crack the session key. To revoke a compromised node, the base station selects a $2t$-degree masking polynomial randomly as $h(x) = h_0 + h_1x + \ldots + h_{2t}x^{2t}$. Polynomial $h(x)$ has t-revocation capability. Base station assigns a personal secret $S_i = h(i)$ to each node i in the network using a secure channel between the node and the base station. For a new session, a session key K_s is generated from pseudo random generator. The base station takes the set $R = \{r_1, r_2, \ldots, r_l\}$ ($l \leq t$) to be revoked, constructs revocation polynomial $g(x) = (x - r_1)(x - r_2) \ldots (x - r_l)$ and, sends the broadcast message as $\{R\} \bigcup \{w(x) = g(x).K_s + h(x)\}$. Any non-compromised node i can retrieve the session key by evaluating $K_s = (w(i) - h(i))/g(i)$. For a compromised node in $\{R\}$, $g(i) = 0$ and therefore, it will not be able to retrieve the key. In the DLS scheme, a node requires to evaluate polynomials to obtain the session key

distributed by the base station and, reveals the personal secret of a valid node to other valid nodes in the network.

A public key based approach was discussed in [85] wherein ECC is used. In [85], each node N is preloaded with a public-private key pair ($PK(N)$, $SK(N)$) with public key $PK(N) = SK(N).G$ for G as a generator on the elliptic curve. The sink possesses the key pair ($PK(S)$, $SK(S)$) and the public key $PK(S)$ is pre-deployed in each node. A network wide key NK is shared by all the nodes and each node can establish a pair-wise key $K_{DH}(N,S) = K_{DH}(S,N)$ with sink using Diffie-Hellman key exchange [38]. On detection of malicious nodes in the neighborhood of a node N, the sink unicasts the revocation message consisting of the list of malicious nodes and a nonce n_s encrypted with $K_{DH}(S,N)$. Receiving node N deletes session keys with all malicious neighbors and confirms the receipt of the revocation message to sink by sending back nonce n_s encrypted with $K_{DH}(N,S)$. Sink also updates the network key NK by unicasting a new key NK' to each non-compromised node I along with nonce $n_{(s,I)}$ using key $K_{DH}(S,I)$. Node I confirms the new key message reception by sending $n_{(s,I)}$ encrypted with $K_{DH}(I,S)$. In this scheme, each node needs to store its own key pair, public key of sink and network key. Each revocation requires a node to send confirmation for revocation message and for updated network key that involves ECC encryption/decryption. Therefore, Even though it is a centralized scheme, the overhead on nodes is high.

In general, the centralized revocation approaches suffer from single point of failure.

6.3 Distributed Voting Mechanism for Revoking a Victim Node

To address the issues in centralized approaches, the researchers have proposed different distributed approaches. In a distributed key revocation, the neighbors of a compromised node decide the revocation instead of some centralized authority. The neighbor nodes use some kind of voting mechanism and a node is confirmed to be revoked once the number of votes against a suspect exceeds a

pre-defined threshold. To address the inefficiencies of EG scheme [42], Chan *et al.* proposed CPS scheme in [23] with the underlying random pair-wise key distribution. Suppose a node i can establish a pair-wise key with m other nodes. These m nodes are called participants for node i and are assigned a random voting key k_i. Each participant node carries a pair-wise key with the target node i as well as a preloaded vote $hash(k_i)$ against node i. When a participant node finds the target node i to be malicious, it uses the vote $hash(k_i)$ to inform other participants. A Merkle tree [89] is used to authenticate the vote, so each participant also keeps log m hash values needed for authenticating a vote message. When a node receives and validates at least t votes against the target node i, node i is marked as "revoked" and the revocation message is passed on to all other network nodes. All the nodes remove the keys associated with revoked nodes from their key rings. An improvement over CPS scheme is proposed in [24] as CGPM scheme in which the voting and revocation decisions are performed by processing only one-hop local broadcast. The revocation is finalized by propagation of a network wide single short message. In CGPM, a random t degree polynomial $q(x)$ is chosen and its cryptographic hash is computed as $H(q(x))$. A revocation vote is generated as $E_{Mask_{jis}}\{(q_{is}(x_{jis}), x_{jis})\}$, where q_{is} is the random polynomial used for revocation against target node i in revocation session s and x_{jis} is the point used by participant j against node i for session j. $Mask_{jis}$ is the activation mask given by the target node i to the participant node j for session s for the purpose of decrypting the vote. When a compromise node i is detected by a neighbor j in revocation session s, node j locally broadcasts unencrypted vote $\{q_{is}(x_{jis}), x_{jis})\}$ and the Merkle [89] authentication values required for verification of the authenticity of this revocation vote. As soon as a neighbor node j receives and authenticates at least t votes, including its own vote, against a target node i, node j computes polynomial q_{is} and $H(q_{is})$ and broadcasts the same. All the nodes receiving this broadcast verify the message with the pre-stored value $H^2(q_{is})$. On successful verification, the nodes delete all the keys shared with target node i and mark i as revoked. The broadcast message is further distributed to reach the entire network.

In [26], Chao *et al.* proposed Blom's matrix [15] based scheme CYLL as an ex-

tension of CGPM scheme. In [26], each node is assigned a public matrix and a secret private matrix used for verification of vote in each voting session. For each revocation session, encrypted votes are preloaded in a node. The scheme uses a vote matrix that is a multiplication of public and private matrices. The votes are generated using the vote matrix, and corresponding column information in public matrix. A set of hash values is also stored by each node. The node uses the hash values for verification of a revocation decision. For m_i number of participants for a node i, and a threshold t of number of votes needed to revoke a node, the public matrix G_i of size t * (m_i + 1) is generated for a node i. Then private symmetric matrices $R_i(s)$ of order t * t are created ($1 \leq s \leq s_{total}$, s_{total} being the number of available revocation sessions against each node). Vote matrices $V_i(s) = (R_i(s) * G_i)^T$ (T denotes transpose of matrix) of size (m_i + 1) * t are then created. For a target node i, the j^{th} row $v_i^j(s)$ of matrix $V_i(S)$ corresponds to participant j. The encrypted votes, $E_{Mask_{ji}}(s)\{v_i^j(s)\}$ are pre-loaded in node j. Activation masks $Mask_{ji}$ are used by participant j to decrypt vote against target node i for session s. The hash values $H(v_i^\lambda(s))$ are also stored by each participant node j of target node i, where $V_i^\lambda(s)$ is the secret sharing row in matrix $V_i(s)$. The connection establishment phase is executed upon receiving a vote against a malicious node or when identifying a node as malicious. In this phase, the participating node and the target node exchange the activation masks to decrypt the vote. Upon receiving threshold number of vote against a target node, the target node is marked as revoked. If the malicious node refuses to exchange mask, participant node will not get the decrypted vote against the target node. After few attempts to establish connection, participating node disconnects the link with the target node and the target node degree is reduced. When the target node degree goes below a predefined threshold, central degree count mechanism is used to revoke the node.

The CPS [23], CGPM [24] and CYLL [26] schemes suffer from node-collusion attacks. In de-centralized schemes such as [95], a single node makes a revocation decision. In [95], the concept of "Suicide for common good" is used wherein a node detecting the compromised node broadcasts the revocation message involving identities of itself and the victim. Both the nodes are blacklisted by all the

nodes. With this scheme, a malicious node may generate false note against a valid node even before the malicious node itself is detected and thus the network may loose valid node. Also, even a valid node may mistakenly generate false alarm against a valid node and this results in loss of two valid nodes.

The distributed approaches, in general, need prior knowledge of node deployment and put heavy overhead on the resource constrained sensor nodes that participate in the voting mechanism to revoke a malicious node.

6.4 Hybrid Node Revocation Methods

To overcome the issues in centralized and distributed approaches, some hybrid node revocation protocols are proposed using multiple base stations or cluster heads for revocation. For example, in KSP [62], a hybrid key revocation scheme is proposed, that applies redundancy on Randomized Grid Based (RGB) scheme [109]. In RGB scheme, a virtual grid of size $m * m$ is considered in which a distinct group of symmetric bivariate is assigned to each row and each column. Each sensor node is then assigned a unique intersection on the grid and the shares of corresponding row and column are given to each node. A node can thus establish a direct key with any other node in its own row or column using the share from the common polynomial, or, can take help of its direct neighbors to establish key with a node on any other row or column. In the key revocation scheme based on RGB [62], the grid structure is extended from size $m * m$ to size $(m + l) * (m + l)$. Polynomial shares are also assigned to these additional grid intersections, but are not allotted to any of the nodes. The scheme also assumes the presence of mobile base stations (MBSs) in the network that may use PKC for secure MBS-MBS communication. Each sensor node has pair-wise keys to communicate with MBSs. The scheme suggests key revocation for nodes for which some of the keys are compromised. When a MBS identifies that some keys of a node are compromised, it broadcasts a revocation message to stop communication with that node. Then, MBS assigns a new intersection on the grid to the victim node with new identity and new polynomial shares. With multiple MBSs, the scheme solves the

problem of single point of failure and also the nodes are not involved in the revocation process. However, the KSP scheme does not handle the revocation of nodes captured by the adversary.

In [27], Chattopadhyay and Turuk also attempted to improve upon [23], CGPM [24] by proposing two key revocation schemes. The schemes in [27] assume that the network consists of hexagonal regions with unique identities $< i, j >$, i and j being the row and column of the region and having at most k nodes in each region. Some regions are designated as basic regions having the criteria that [i % 2 = 0 and j % 2 = 0 and i % 4 $\neq$ 0] or [i % 4 = 0 and j % 4 = 1]. If a region is not a basic region, then it is a non-basic region with at most two basic regions in its neighborhood. A set $f_1(x, y, z)$, $f_2(x, y, z)$ ... $f_n(x, y, z)$ of tri-variate polynomials is chosen with highest degree of x, y and z as $7k$. Each basic region has a unique tri-variate polynomial and a non-basic region gets at most two tri-variate polynomials that are drawn from the basic regions in their neighborhood. Each node with a unique identity u in a region is assigned shares of polynomial(s) of that region and a unique hash function. The node u in a region having polynomial $f_1(x, y, z)$ gets $auth_u^1(y, z) = f_1(u, y, z)$ as authentication polynomial and $verf_u^1(y, z) = f_1(x, u, z)$ as verification polynomial. To vote against a node v, node prepares a message M having the identity of v to all the neighbors of v and also to the base station. A node w that is a neighbor of node u within same or in the neighboring region shares a polynomial, say $f_i(x, y, z)$ with node u. Node u sends message M to node w along with a single value $p = auth_u^i(w, h(M))$. When w receives this message (M, p), it computes $q = verf_w^i(u, h(M))$ using its own verification polynomial. If $p = q$, node w is assured of the authenticity of the message sent by u and node w appends its own suspect list with the identity of node v, if not already present along with the accuser name. Node w increments the revocation counter against node v by 1. When the number of revocation counts at a node w against a suspect node v reaches the threshold value t, the node v is added to the blacklist maintained by node w. After putting node v in blacklist, node w stops communicating with v and delete all the shared keys with v as well as path keys created through v. When base station receives t votes against a node v, it pre-

pares revocation message containing identity of revoked node v, all the compromised keys and broadcasts the message along with authentication polynomials $\sum_{i=1}^{n} f_i(baseId, y, h(M))$, where $baseId$ is the identity of the base station. Each node after receiving the revocation broadcast from base station, checks the authenticity of the message using its own verification polynomial and, once confirmed, deletes all the compromised keys mentioned in the message M and puts the victim node in blacklist. The scheme prevents sybil attack and node replication attack as the list of compromised nodes exist in each node. Another variant of this scheme use monitor nodes in each region to decide and perform final revocation.

In [44], the hybrid key revocation scheme suggests using voting process among nodes, but global revocation is based on the process as given in [24]. The scheme [44] eliminates the need of having prior knowledge of node deployment. The base station autonomously generates and distributes shared secrets for nodes. For this purpose, base station stores a (t - 1) degree polynomial $f(x) = S_i + a_1.x + a_2.x^2 + \ldots + a_{t-1}.x^{t-1}$ in each node and thus each node has its own secret S_i. Each node also stores a hash value $H(S_i)$ and $Mask_i$ that is used to encrypt/decrypt a vote. Each node i stores a local link list that contains the information about each neighbor j with which the node i has established an encrypted link. The list contents are of the form $< ID_j, K_{ij} >$, where ID_j is the unique identity of node j and K_{ij} is the secret key between nodes i and j. Each node i also maintains a path-key list of the form $< ID_{N_1}, ID_{N_2}, \ldots, ID_{N_k} >$ that contains the details of each node j with which node i has established a secret path. In this list, ID_{N_1}, ID_{N_2}, ..., ID_{N_k} are unique identities of the intermediate nodes on the path between nodes i and j. An initially empty revocation list is maintained by each node as well. Two nodes having established a pair-wise key become voting members for each other. A unique secret share is generated as $S_{ij} = S_i + a_1.ID_j + a_2.ID_j^2 + \ldots + a_{t-1}.ID_j^{t-1}$ for each voting member j for node i. Node i sends $E_{K_{ij}}\{S_{ij}, Mask_i, H(S_i)\}$ to each partner node j and thus a partner node is eligible to vote against the node. In case, a partner node j is continuously refused for sending shared secret and mask by a partner node i, it is disconnected. A node with degree less than the pre-defined threshold is revoked by base station. To vote against a node i, part-

ner node j broadcasts $E_{Mask_i}\{ID_j, S_{ij}\}$ in both the current and the next session to ensure that a vote sent near the end of a session is also considered by local vote members for further dissemination. Any other voting member of i, on reception of this broadcast by j, decrypts the message using mask, stores the message contents and rebroadcasts the message. At the end of the session, if a node j receives t votes, it uses the shares to compute S_i, validates $H(S_i)$ value and clears the details of node i from local link list and adds node i in the revocation list. Node j also informs base station along with S_i encrypted with pair-wise key shared between base station and j. Base station verifies S_i using $H(S_i)$ and broadcasts the revocation to all including non-local voting members. The path-key list is also updated accordingly. In this scheme, since no session specific details are involved in vote message or revocation message, it is vulnerable to replay attack. Moreover, in all these schemes, a sensor node has to evaluate polynomials for sending and receiving a vote. Furthermore, nodes need to process a revocation message received from the base station.

In [55], another hybrid revocation protocol is proposed with voting mechanism by cluster nodes. The cluster head is responsible for broadcasting the final revocation message with its signature using the public key. The nodes need to store the public key of cluster head to verify the revocation message. In all the hybrid approaches discussed above, although, base station or cluster heads are responsible for final revocation, the nodes are involved in voting mechanism and therefore the schemes are more distributed and incur computation, communication and storage overheads on the nodes. Recently, Rams and Pacyna [105] had given a group key distribution with revocation capability (referred as RP scheme) by invalidating personal secret keys during user revocation. In [105], m random polynomials $s_1(x)$, $s_1(x)$, ..., $s_m(x)$ of degree $2t$ each and a t-degree polynomial $h_1(x)$ are selected by group manager. Each user $u_i \in CG_1$ is assigned a random unique index $x_i \in F_p$. Group manager sends to each user u_i, the user's personal secret $S_i = \{x_i, s_1(x), s_2(x), \ldots, s_m(x), h_1(x)\}$ using some secure communication channel. For a communication session j, group manager randomly, uniformly and independently selects an update polynomial $\delta(x) \in F_p(x)$, session key component $k_j \in F_p$

and a masking value $v_j \in F_p$. The update polynomial is such that $\delta_j(x) \notin \{\delta_1(x), \delta_2(x), \ldots, \delta_{j-1}(x)\}$. Group manager also computes $h_j(x) = h_1(x) + \sum_{i=1}^{j-1} \delta_l(x)$; $h_j(x) \notin \{h_1(x), h_2(x), \ldots, h_{j-1}(x)\}$. Next, a set of t distinct indexes $W_j = \{w_1(x), w_2(x), \ldots, w_t(x)\} \subseteq F_p$ is selected in a way that the set W_j includes all the nodes revoked for session j and does not include any node from communication group CG_j for session j. The revocation polynomial $r_j(x) = \prod_{x_i \in W_j} (x - x_i)$ is prepared and then the session key $K_j = g^{k_j}$, a value $z_j = g^{k_j + v_j . h_j(0)}$ and polynomial $\Delta_j(x) = \delta_j(x). r_j(x) + s_j(x)$ are computed. Finally the group manager constructs the broadcast component as $b_j = [g^{v_j}, z_j, \{w_l, g^{v_j . h_j(w_l)}\}_{w_l \in W_j}, \Delta_j(x)]$. From the broadcast, for extracting the session key K_j and update the personal secret data, a node u_i takes its personal secret $h_j(x_i)$ and first calculates $g^{v_j . h_j(x_i)} = (g^{v_j})^{h_j(x_i)}$. Node uses Lagrange's interpolation [35] with t points $\{w_1(x), w_2(x), \ldots, w_t(x)\}$ to obtain $g^{v_j . h_j(0)}$. The new session key K_j is then retrieved as $K_j = z_j / (g^{v_j . h_j(0)})$. To update the personal secret of node u_i, the node recovers $\delta_j(x_i) = \frac{\Delta_j(x_i) - s_j(x_i)}{r_j(x_i)}$ and computes $h_{j+1}(x_i) = h_j(x_i) + \delta_j(x_i)$.

The RP scheme [105] for hybrid node revocation does not put overhead of voting process on nodes, however, this scheme also uses polynomial based keys and involves exponent computation, Lagrange's interpolation and polynomial evaluation for revocation and key update that are costly for resource constrained sensor nodes.

Although, hybrid or semi-centralized approaches address the issues of centralized and distributed approaches, we observed the scope of reducing overhead on resource constrained sensor nodes by providing the key revocation with robust security in the context of node capture. The proposed node revocation and key update (NRKU) protocol is discussed in subsequent section.

6.5 Proposed Protocol for Node Revocation and Key Update (NRKU)

We present a node revocation and key update protocol that uses Chinese remainder theorem based group key distribution to update the session key for non-captured nodes and revoke the captured node from participating in future network activities.

6.5.1 Goals and Assumptions

The NRKU protocol aims to securely and efficiently withdraw the group key of a compromised node that is found to be a victim of node capture attack. On successful detection of node capture attack, the group session key must be updated in such a way that the victim node is not able to access the new key and therefore revoked from participating in further network activities.

We assume that within a cluster, the nodes communicate using a shared group key that is broadcasted by cluster head for each session. A cluster head runs key revocation and update protocol on node capture detection through TPM enabled Program Integrity Verification (TPIV) protocol [5].

The proposed protocol has two phases as explained below.

6.5.2 Initial Session Setup

In the beginning, the head of each cluster distributes initial session key to all the nodes within its cluster, assuming all the nodes in the cluster are non-compromised at this initial set-up phase. The initial session secret $K_0 \in Z_q^*$ and a nonce $N_0 \in Z_q^*$ are randomly selected by a cluster head. The cluster head generates an initial broadcast message. For this purpose, the cluster head takes the initial session key K_0, the initial group of nodes in the cluster as CG_0, the set of secrets $S = \{s_1, s_2, \ldots, s_d\}$ for all d nodes in the group, initial nonce N_0 and the initial empty list of revoked nodes R_0. Using these inputs, the algorithm AUTH_SKEY() is executed to obtain initial broadcast message B_0 = **Auth_SKEY**(K_0, CG_0, S, N_0, R_0).

The steps involved in AUTH_SKEY algorithm are given in Algorithm 6.1.

Algorithm 6.1 Authenticated Secret Key Distribution

Input : K_l, CG_l, S, N_l, R_l (for session l)

K_l - Group shared secret

CG_l - Set of nodes under a CH

S - Set of individual node secrets

N_l - Nonce

R_l - Set of nodes marked for revocation

Output : B_l (Broadcast message for session l)

1: **procedure** AUTH–SKEY(K_l, CG_l, S, N_l, R_l)
2: **for** each node $u_i \in CG_l$ **do**
3: Compute $b_{il} = (K_l \oplus s_i)\ mod\ s_i$
4: Compute $r_{il} = \lfloor (K_l \oplus s_i)/s_i \rfloor$
5: **end for**
6: Solve $X_l \equiv b_{il} \bmod s_i\ (\forall\ u_i \in CG_l)$
7: Solve $Y_l \equiv r_{il} \bmod s_i\ (\forall\ u_i \in CG_l)$
8: **for** each node $u_i \in CG_l$ **do**
9: Compute $c_{il} = PRF_{s_i}(ID_i, X_l, Y_l, R_l, N_l)$
10: **end for**
11: Solve $W_l \equiv c_{il} \bmod s_i\ (\forall\ u_i \in CG_l)$
12: Compute $B_l = (X_l, Y_l, W_l, R_l, N_l)$
13: **return** B_l
14: **end procedure**

6.5.3 Node Revocation and Key Update

The nodes in a cluster are under continuous monitoring by its cluster head. When the cluster head suspects a node to be a victim of node capture attack, the cluster head executes the program integrity verification protocol [5] with the suspect node to verify the node's program integrity. If the node capture attack is confirmed through the program integrity verification, cluster head adds the node in the revocation list. To start the next session, the cluster head prepares the revocation message to distribute revocation list and the new session group key. Although, cluster head broadcasts the revocation message, the keying material is prepared in such a manner that new key can not be extracted by any of the re-

voked nodes. Since a node revoked from the group is excluded from obtaining the new session group key, the revoked node can not participate in future group activities. The cluster head may choose to either update the node's personal secret manually and thus repair the node or it may choose to leave the node useless by depleting node's battery by unicasting a command involving very heavy computations.

We now discuss the details of the two sub modules involved in the node revocation and key update protocol, namely *revocation message broadcast* and *authentication and session key update*.

6.5.3.1 Revocation Message Broadcast

To construct the revocation message for a session j, the cluster head randomly selects a new session group key $K_j \in Z_q^*$. The cluster head also selects a random nonce $N_j \in Z_q^*$ to be used for the purpose of ensuring message freshness. The cluster head picks the set R_j of nodes revoked from this session j, the set J_j of nodes joining the group from session j, set CG_{j-1} of the group members that were authorized members in session j-1 and the set S of personal secrets of nodes. The current communication group CG_j includes all the new nodes joining the cluster from this session and excludes all the nodes revoked from this session. With K_j, N_j, R_j, CG_j and S, the cluster head executes AUTH_SKEY() procedure (Refer 6.1) to generate the broadcast B_j for session j. The cluster head intimates the non-captured nodes about revocation and distributes them a new session key by executing the *Revocation Message Broadcast* procedure given in Algorithm 6.2.

6.5.3.2 Authentication and Session Key Update

On receiving the broadcast message B_j from the cluster head for session j, a node u_i first authenticates the message and the sender. The cluster head provides the PRF value of all the components of the message along with the broadcast. This PRF value is computed for each individual non-captured node u_i that is member of communication group CG_j for session j using the node's secret s_i. After authenticity check with the help of PRF, the node checks if the current nonce N_j is

Algorithm 6.2 Revocation Message Broadcast

Input : CG_{j-1}, S, R_j, J_j (for session j)

CG_{j-1} - Set of nodes under a cluster head from previous session

S - Set of individual node secrets

R_j - Set of nodes marked for revocation from session j

J_j - Set of nodes newly joining from session j

Output : B_j (Broadcast message for session j)

1: **procedure** REV–MSG–BCAST(CG_{j-1}, S, R_j, J_j)
2: Choose a new group key $K_j \in_R Z_q^*$
3: Select a nonce $N_j \in_R Z_q$
4: Picks the sets J_j and R_j
5: Prepares set $CG_j = (CG_{j-1}/R_j) \bigcup J_j$
6: Sets B_j = **Auth_SKEY**(K_j, CG_j, S, N_j, R_j)
7: **return** B_j, CG_j // (for broadcast)
8: **end procedure**

greater than the previous nonce. The nonces chosen are in ascending order for each subsequent session and used to ensure the freshness of the broadcast. After authenticity and freshness check, a valid member node marks all the nodes in R_j as revoked. Now, the node proceeds to extract the new session key K_j from the broadcast message. Algorithm 6.3 provides the step-wise details of the authentication and session key update process.

Apart from revocation of a node, the protocol also provides the joining of new nodes in a cluster. As a node makes a move to a new cluster, the node informs its cluster head about this move. When this node requests another cluster head for joining that new cluster, the new cluster head gets the details of the node from the node's previous cluster head and upon ensuring the validity of the node, the new cluster's head includes the new node into its cluster. Suppose a node u_w moves from cluster C_1 and joins the cluster C_2 from session j. After validating node u_w with the help of head of previous cluster C_1, the head of cluster C_2 considers the node u_w while preparing its next broadcast message B_j. Firstly, it computes $b_{wj} = (K_j \oplus s_w) \; mod \; s_w$ and corresponding multiplication factor $r_{wj} = \lfloor (s_w \oplus K_j)/s_w \rfloor$ and then solves the congruence for X_j and Y_j including b_{wj} and r_{wj} respectively.

Algorithm 6.3 Authentication and Session Key Update

Input : B_j, s_i

B_j - Broadcast message received

s_i - Node secret shared with CH

Output : K_j (Session secret key)

1: **procedure** AUTH–KEY–UPDATE(B_j, s_i)
2: Pick X_j, Y_j, R_j, N_j from broadcast B_j
3: Compute $c'_{ij} = PRF_{s_i}(ID_i, X_j, Y_j, R_j, N_j)$
4: Pick W_j from broadcast B_j
5: Retrieves c_{ij} from W_j as $c_{ij} = W_j \bmod s_i$
6: **if** $c'_{ij} \neq c_{ij}$ **then**
7: REJECT B_j
8: **return**
9: **end if**
10: **if** $u_i \in R_j$ **then return**
11: **end if**
12: **if** $N_j > N_{j-1}$ **then**
13: Accept the broadcast message as valid
14: Mark the nodes in R_j as revoked
15: Compute $r_{ij} = Y_j \bmod s_i$
16: Compute $K_j = ((X_j \bmod s_i) + (r_{ij})) \oplus s_i$
17: **return** K_j
18: **else**
19: REJECT B_j **return**
20: **end if**
21: **end procedure**

Furthermore, C_2 computes $c_{wj} = PRF_{s_w}(ID_w, X, Y_j, R_j, N_j)$ and then solves the congruence for W_j to include c_{wj}. Therefore, from session j, node u_w becomes a valid member of cluster C_2 until it is revoked or the node itself moves to another cluster.

6.5.4 Security Strengths

The protocol for node revocation and key update discussed above provides secure key revocation along with resistance to node collusion, impersonation and replay attacks and, forward and backward secrecy. We prove the security claims in the

form of theorems with proof by contradiction using the attacker's knowledge at various stages.

Initial State: Initially, the attacker has the knowledge of the cluster heads and the nodes through the network topology. An attacker has access to the public channel and the public functions. Therefore, an attacker can read all the messages being communicated on the public channel and can also compute the public functions. At this state, the attacker does not know the secret of any node in the network:

Attacker(U)	... (6.1.1)
Not Attacker(s_i) $(1 \leq i \leq n)$	... (6.1.2)
Attacker($PRF()$)	... (6.1.3)
Attacker(L)	... (6.1.4)

6.5.4.1 Secure Node Revocation

Theorem 6.1. *A node revoked from a session does not have access to new session key and is not allowed to participate in any future sessions.*

Proof: As the protocol execution proceeds, we trace the knowledge of the attacker.

Intermediate State: The attacker has the personal secret s_e of a node u_e captured in session j. As the cluster head detects this capture, it prepares the list of revoked nodes as R_t before the start of the next session t. The cluster head also updates the member list CG_t to exclude revoked members and to include new joins.

Now, the broadcast message B_t containing new key K_t is prepared by the cluster head without involving the secrets associated with nodes in R_t. The Cluster head broadcasts message B_t on the public channel. Thus:

Attacker(s_e)	... (6.2.1)
Not Attacker(s_i) $(\forall u_i \in CG_t)$	... (6.2.2)
Attacker(B_t)	... (6.2.3)

We now claim that the attacker with s_e has obtained the key K_t and to prove this claim, we trace back the protocol steps as below:

Attacker(K_t)

$\Rightarrow$ Attacker(((X_t mod s_i) + r_{it}) $\oplus$ s_i)

[from step 16 of Algorithm 6.3]

$\Rightarrow$ Attacker(X_t) $\wedge$ Attacker(s_i) $\wedge$ Attacker(r_{it})

$\Rightarrow$ Attacker(X_t) $\wedge$ Attacker(s_i) $\wedge$ Attacker(Y_t mod s_i)

[from step 15 of Algorithm 6.3]

$\Rightarrow$ Attacker(X_t) $\wedge$ Attacker(s_i) $\wedge$ Attacker(Y_t)

$\Rightarrow$ Attacker(B_t) $\wedge$ Attacker(s_i)

[from step 12 of Algorithm 6.1]

$\Rightarrow$ TRUE $\wedge$ FALSE

[from assertions 6.2.3 and 6.2.2 respectively]

$\Rightarrow$ FALSE

From the above trace, it is evident that an attacker, who could capture a node u_e in session j, is not able to obtain the key K_t for any future session t ($> j$), when the node u_e is revoked. The node once revoked from the group does not get membership in any future session t ($> j$), unless the node is taken back as a new valid user with new credentials, this claim holds for all the future sessions.

The contradiction, therefore, proves the secure revocation capability in the proposed protocol. □

6.5.4.2 Forward and Backward Secrecy

Forward secrecy ensures that even when one of the long-term keys is compromised in the future, a session key derived from a set of long-term keys will not be compromised. Backward secrecy ensures that the past session keys or long-term keys will not be revealed on compromise of a session key.

In the context of group key distribution [104], forward secrecy ensures that any number of users revoked before session j colluding together can not recover any of the future session keys K_j, K_{j+1}, ..., K_m. (m is the total number of sessions). Thus, $\forall u_i \in \bigcup R_l$ ($1 \leq l \leq j-1$), collusion will not reveal any key K_w ($j \leq w \leq m$).

Similarly, backward secrecy implies that any number of users joining from session j colluding together can not recover any of the past session keys K_1, K_2, ..., K_{j-1} [104]. This implies that $\forall u_i \in CG_p$ ($j \leq p \leq m$) and $u_i \notin CG_l$ ($1 \leq l \leq j$-1), collusion will not reveal any key K_w ($1 \leq w \leq j$-1).

Theorem 6.2. *The protocol ensures forward and backward secrecy.*

Proof: From *Theorem 6.1*, a node revoked in session j, $u_e \in R_j$, can not access K_l ($j \leq l \leq m$). Now, what remains to prove is that all the revoked nodes in set $\bigcup R_p$ ($1 \leq p \leq j$-1) together can not reveal key K_l ($j \leq l \leq m$).

Intermediate State: Up to this stage, j-1 sessions have passed and we have the set of all the nodes revoked up to session j-1 i.e. $\bigcup R_p$ ($1 \leq p \leq j$-1).

Now, cluster head broadcasts B_j. Thus:

Attacker(s_i)	($\forall u_i \in \bigcup R_p$)	... (6.3.1)
Not Attacker(s_i)	($\forall u_i \in U \setminus \bigcup R_p$)	... (6.3.2)
Attacker(B_j)		... (6.3.3)

Assume that the attacker has obtained the session key K_j for session j with all the above mentioned knowledge of public functions, public channel, network topology and secret keys of revoked nodes (as in assertions 6.1.1 to 6.1.4 and 6.3.1 to 6.3.3). Let us trace back this scenario:

Attacker(K_j)

$\Rightarrow$ Attacker(((X_j mod s_i) + $r_{i,j}$) $\oplus$ s_i) (for some $u_i \in CG_j$)

[from step 16 of Algorithm 6.3]

$\Rightarrow$ Attacker(X_j) $\wedge$ Attacker(s_i) $\wedge$ Attacker(r_{ij})

$\Rightarrow$ Attacker(X_j) $\wedge$ Attacker(s_i) $\wedge$ Attacker(Y_j mod s_i)

[from step 15 of Algorithm 6.3]

$\Rightarrow$ Attacker(X_j) $\wedge$ Attacker(s_i) $\wedge$ Attacker(Y_j)

$\Rightarrow$ Attacker(B_j) $\wedge$ Attacker(s_i)

[from step 12 of Algorithm 6.1]

$\Rightarrow$ TRUE $\wedge$ FALSE

[from assertions 6.3.3 and 6.3.2 respectively]

$\Rightarrow$ FALSE

This shows that even with s_i ($\forall\ u_i \in \bigcup R_p$, 1$\leq\ p \leq$ j-1), the attacker can not get K_l ($j \leq l \leq m$) i.e. the collusion of revoked nodes does not reveal the future session keys and protocol ensures forward secrecy.

On the similar lines, we can prove that even with s_i ($\forall\ u_i \in \bigcup R_l$, $j \leq l \leq m$), an attacker can not get K_p ($1 \leq p \leq$ j-1) i.e. the collusion of new joining nodes does not reveal the past session keys, therefore, providing backward secrecy in the protocol. □

6.5.4.3 Resistance to Node Collusion Attack

Node collusion resistance ensures that the key K_w for a session w (j $\leq$ w $\leq$ t) can not be recovered by the collusion of all users of the group $R_j \cup J_{t+1}$.

Theorem 6.3. *The proposed protocol resists the node collusion attack*

Proof: As the broadcast message B_w for session w (j $\leq$ w $\leq$ t) is available on public channel and the nodes in groups R_j and J_{t+1} collude:

Attacker(K_l)	((0 $\leq$ l $<$ j) and (t $<$ l))	... (6.4.1)
Attacker(s_i)	($\forall u_i \in R_j \cup J_{t+1}$)	... (6.4.2)
Not Attacker(s_i)	($\forall u_i \notin R_j \cup J_{t+1}$)	... (6.4.3)
Attacker(B_w)	(j $\leq$ w $\leq$ t)	... (6.4.4)

Let us now assume that the node collusion attack is successful and the key K_w for a session w (j $\leq$ w $\leq$ t) is accessible to the collusion of all users of the group $R_j \cup J_{t+1}$. Therefore:

Attacker(K_w) (for j $\leq$ w $\leq$ t)

$\Rightarrow$ Attacker(((X_w mod s_i) + $r_{i,w}$) $\oplus$ s_i) (for some $u_i \in CG_w$)

[from step 16 of Algorithm 6.3]

$\Rightarrow$ Attacker(X_w) $\wedge$ Attacker(s_i) $\wedge$ Attacker($r_{i,w}$)

$\Rightarrow$ Attacker(X_w) $\wedge$ Attacker(s_i) $\wedge$ Attacker(Y_w mod s_i)

[from step 15 of Algorithm 6.3]

$\Rightarrow$ Attacker(X_w) $\wedge$ Attacker(s_i) $\wedge$ Attacker(Y_w)

$\Rightarrow$ Attacker(B_w) $\wedge$ Attacker(s_i) (for some $u_i \in CG_w$)

[from step 12 of Algorithm 6.1]

$\Rightarrow$ TRUE $\wedge$ FALSE

[from assertions 6.3.3 and 6.3.2 respectively]

$\Rightarrow$ FALSE

This contradiction fails our assumption of successful node collusion attack. □

6.5.4.4 Resistance to Impersonation and Replay Attacks

The protocol also provides resistance to impersonation and replay attacks. Impersonation implies that an attacker can claim to be a valid group member. However, in our protocol, the congruence value X_w for a session w that contains the session key K_w is computed only for valid group members involving their respective personal secrets. Therefore, even if an attacker gets X_w, can not retrieve the key K_w unless it has obtained the secret s_i of a valid group member u_i. The protocol thus resists impersonation attack. For resisting replay attack, each group key broadcast message includes a nonce N_j for a session j. The nonces are in increasing order for each later session i.e. $N_j < N_{j+1}$. Before proceeding for key retrieval, a node checks the nonce to ensure that it is greater than the previous nonce. Therefore, an adversary can not re-transmit an old message in some later session.

6.5.4.5 Existing Revocation Protocols v/s NRKU

Major security features of the proposed protocol are compared with some of the existing protocol as shown in Table 6.1. We observed that the centralized DLS scheme [61] neither provides resilience to replay and impersonation attack nor it resists the node collusion attack. The distributed approach proposed as CT scheme in [27] is resilient to replay and impersonation attacks, however, the resistance to node collusion depends on the degree k of the polynomial used in the scheme. Both DLS and CT schemes do not discuss about the forward and backward secrecy of the key. The RP scheme for hybrid revocation presented in [105]

is secure against replay and impersonation attacks, however, as the RP scheme is also polynomial based, the node collusion resistance as well as forward and backward secrecy are limited to the degree k of the polynomial.

Schemes → / Features ↓	DLS [61] (Centralized)	CT [27] (Distributed)	RP [105] (Hybrid)	Proposed [2] (Hybrid)
Resilience to Replay attack	No	Yes	Yes	Yes
Resilience to Impersonation attack	No	Yes	Yes	Yes
Resistance to Node Collusion Attack	No	up to k-collusion	up to k-collusion	Yes
Forward and Backward Secrecy	–	–	k-forward k-backward	Yes

Table 6.1: Comparison of Security Features of Revocation Protocols

In the table 6.1, k is the degree of polynomial used in revocation schemes based on polynomials. '–' is an indication that the respective work does not consider the feature.

6.5.5 Performance Boost with NRKU

With the given node revocation and key update protocol, the **computation overhead** is almost negligible at node end. This is due to the fact that a node only needs to perform basic XOR and *mod* operations to retrieve the session key from the broadcast message.

The protocol proposes to keep the personal secret of a node stored as incompressible bit strings in the program memory of the node [5], thus the **storage overhead** is only to keep a group key for each session.

The **communication overhead** is seen as the number of bits transmitted and received by a node. In this protocol, a node does not transmit any message for node revocation and key update and receives 5 * log q (log q - size of the key in bits) bits of single broadcast.

The comparison of performance parameters is summarized in Table 6.2. In the table, k is the degree of the polynomial in the protocols that use polynomial based

primitives, m is for maximum number of sessions and d is for the number of votes transmitted/received in distributed schemes.

Schemes → / Features ↓	DLS [61] (Centralized)	CT [27] (Distributed)	RP [105] (Hybrid)	Proposed [2] (Hybrid)
Computation Cost	O(k)	O(k^3)	O(k^2)	O(1)
Storage Cost	O(1)	O(k^2)	O(mk)	O(1)
Communication Cost	O(k)	O(d)	O(1)	O(1)

Table 6.2: Performance Comparison of Revocation Protocols

We compared the node energy consumption of NRKU protocol [2] and the latest hybrid RP protocol for revocation [105] using the Castalia simulator [20]. From Figure 6.1, it is evident that the energy consumption in NRKU protocol is constantly low irrespective of the network size in comparison to the RP protocol.

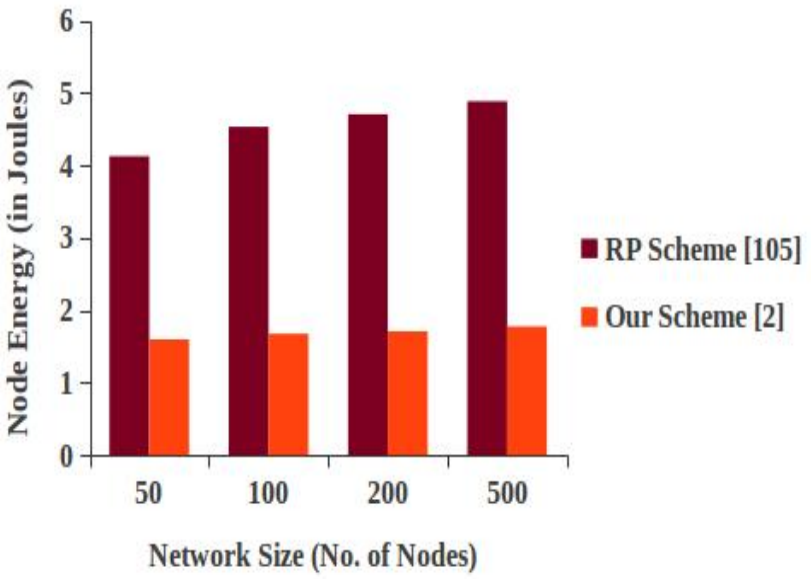


Figure 6.1: Comparison of Energy Consumption for NRKU and RP Protocol

Also, we implemented both the protocols on ATMega 328 processor using Arduino Duemilanove controller board and ArduinoISP programmer for computation cost. The experimental results in Figure 6.2 shows that with NRKU protocol, the computation time reduces to almost 90 %.

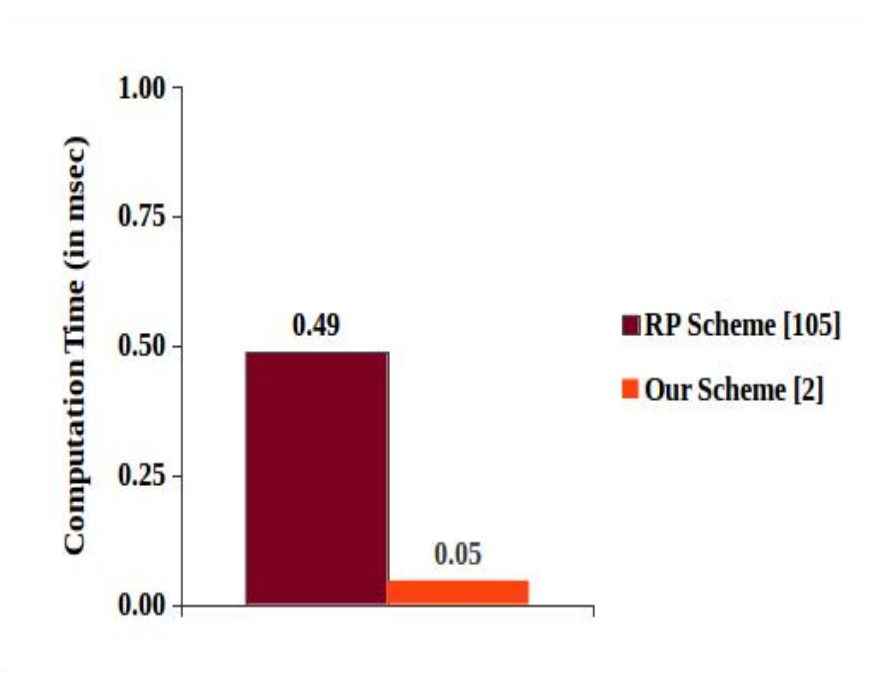


Figure 6.2: Comparison of Computation Cost for NRKU and RP Protocol

6.6 Conclusion

In this chapter, we discussed about the vitality of node revocation and key update aspects in relation to node capture attack in a WSN. When a node becomes a victim of node capture attack, it serves the purpose of an adversary to damage the network from within as an insider attacker. Therefore, it is imperative to exclude such node from the network operations. A viable approach to node revocation is to revoke the secret key of such captured node and provide a new key to the non-captured nodes within the group. The existing key revocation schemes are based on centralized, distributed or hybrid approaches. The centralized approaches suffer from the threat of single point of failure. Moreover, the base station requires to communicate with nodes to gather information about the malicious node, and inform nodes on revocation decisions resulting in delays that gives time to an attacker for performing more attacks. The distributed approaches, on the other hand, are vulnerable to node collusion attack and also demand the resource constrained nodes to actively participate in voting mechanism for revocation resulting in overhead. Hybrid revocation schemes attempted

to address the security issues in centralized and distributed approaches. However, although most of the hybrid approaches put the revocation decision on base station but nodes are involved in voting to identify the nodes for revocation. In this chapter, we presented a node revocation and key update protocol using Chinese remainder theorem based secret sharing. The protocol provides an efficient and secure way to revoke a node when it is detected to be a victim of node capture attack. The proposed protocol not only provides secure node revocation but also ensures forward and backward secrecy and, resistance to node collusion, replay and impersonation attacks. We compared the performance of the proposed node revocation and key update protocol with some of the existing protocols and it is evident from the analytical comparisons and, from the experiments and simulation results that the protocol out performs by providing the required security with significantly reduced computation, communication and storage costs.

CHAPTER 7

Conclusion and Future Work

We have presented a comprehensive approach to deal with node capture detection in a WSN in a secure and efficient manner. The proposed solution integrates the key establishment, node capture detection and node revocation in a clustered mobile sensor network and can be used for any application with an unattended deployment that demands security against the threat of node capture attack.

We first discussed an authenticated pair-wise key establishment and key update protocol for WSN in dynamic environments [4]. We used a multi-polynomial share based scheme to establish a master secret key between a pair of nodes. Using this master secret, a pair of nodes can generate a fresh pair-wise key. The protocol provides node authentication without involvement of the base-station and without any additional communication overhead; in the node discovery phase itself the nodes are authenticated and initial key establishment takes place. Moreover, the protocol implements a key update mechanism that allows two nodes to negotiate a new pair-wise key anytime during the communication. The protocol is proven secure against impersonation, replay, worm hole, sink hole and known-key attacks. The protocol provides data confidentiality through a dynamic secret key and ensures forward secrecy and mutual key control. The underlying multi polynomial scheme provides a very good resilience against node capture attack.

The pair-wise key establishment is used for communication between base station and cluster heads. Within a cluster of nodes, we proposed a group session key managed by the respective cluster head. We then discussed a self-healing and mutual healing enabled group key distribution protocol. During key distribution, a node may miss out on one or more broadcasts. We discussed self-healing to get

such missing information wherein, using the recent broadcast keying material, a node extracts the key used in the previous sessions, if that node was an authorized member of the session. Furthermore, if the node misses out the recent broadcast, it seeks help of its neighboring nodes and obtains the missing key through mutual healing. We provided an efficient approach for generating the broadcast information and key extraction from the keying broadcast material using Chinese remainder theorem based secret sharing. Our protocol provides self-healing and mutual-healing capability along with additional security features such as resistance to impersonation attack and key message confirmation with authentication in mutual healing. The required security of group-key distribution along with self-healing and mutual-healing capability is achieved at significantly low storage, computation and communication overhead, which is a significant achievement in WSN environment.

Since the unattended deployment of resource constrained sensor nodes in hazardous environments leaves the nodes exposed to node capture attack, we next discussed a protocol to verify the integrity of a sensor node program to detect node capture attack, that is an adversary physically capturing, reprogramming and redeploying a node, in a distributed WSN setup. The protocol provides authentication of a node and verification of the program integrity that helps detecting the node capture attack with less overhead as compared to the existing program integrity verification protocols [5]. Prior to node program integrity verification, a node can authenticate the verification server. Each verification server is equipped with trusted platform module (TPM), so the protocol overcomes the threat of attacker knowing the node program stored at the verifier and also ensures that capture of a node does not reveal the secret of any other node in the network. Additional security such as protecting node secrets from compromised verifier and protection against node memory addition is provided as compared to the pure software based protocols with significant reduction in communication, computation and storage overhead on the nodes. Moreover, the cost of equipping all the nodes with the TPM chip is saved, resulting in the overall reduced cost of network deployment and maintenance.

At the end, we discussed an efficient and secure protocol for node revocation and key update after node capture detection [2]. The proposed protocol provides secure node revocation and also ensures the forward and backward secrecy, resistance to node collusion, replay and impersonation attacks.

The security claims in all the proposed protocols are verified using analytical reasoning, theorem proving technique and formal analysis with ProVerif tool. The performance improvements claimed in the protocols are the results of analytical comparisons as well as the simulation and experiments carried out. We used the Castalia simulator for simulating the performance of the protocols in the real-time networks and carried out experiments on ATmega328 processor using Arduino Duemilanove controller board and ArduinoISP programmer.

In pair-wise key establishment and key update protocol, it is shown that the computation cost of session key update remains constant and does not get affected even if the degree of polynomial to establish the master secret is increased to enhance the node capture resilience. When degree of polynomial is high (≥ 200), the session key update cost is significantly low as compared to one time initial session key establishment. The node energy consumption for session key update protocol is less than half the energy required for initial key establishment irrespective of the network size. The experimental results for self-healing and mutual-healing protocols reveal that when the computation cost for self-healing and mutual healing is compared with the existing protocol, the cost is significantly reduced with proposed bilinear pairing based approach and is almost negligible with the proposed symmetric key based healing. In the proposed TPIV protocol for node capture detection, we proved through the experiment that the adversary can not elude the program integrity verification even when he puts additional memory in the captured node. The computation cost, energy consumption and communication latency is reduced with TPIV as compared to the existing DAPP protocol. The experiment carried out to compare the performance of the proposed node revocation and key update (NRKU) protocol with the existing RP protocol shows that the computation overhead and node energy consumption with NRKU is notably low.

To summarize, this thesis presents a secure and efficient solution for detection of node capture attack supported by secure key establishment, self healing and revocation of a victim of node capture from the network. The proposed solution can be used in any clustered mobile sensor network for applications where nodes are unattended after deployment and are vulnerable to node capture attack.

In the thesis, we have used various mathematical and cryptographic primitives. In the pair-wise key update protocol, the initial session key set up is proposed using Diffie-Hellman (DH) key exchange that is considered to be computationally heavier for resource constrained sensor nodes. Although the pair-wise DH key exchange is proposed to be executed only once between a pair of resource rich cluster heads, there is a scope of improvement and the DH key exchange can be replaced with a more efficient method.

We have given the bi-linear pairing based protocol for healing enabled group key distribution. The pairing based cryptography is also considered to be computationally heavy. Although, we have proposed a symmetric key based protocol, we see the scope of considering more efficient public-key based approach to replace the pairing-based approach.

The proposed framework mandates the deployment of cluster heads equipped with trusted platform module. In future, we plan to ease this assumption by using software based implementation for establishing a trust base. We further look forward to explore the feasibility of equipping a node for self-defence against the node capture attack.

The node revocation and key update protocol provides mitigation to revoked node in terms of manually repairing the node. We plan to explore the possibility of providing solution for remotely repairing the victim of node capture attack to reuse the node.

Furthermore, with the emergence of cyber physical system (CPS), the interactions amongst humans and objects in the physical and in the virtual world is going to be an enriching experience. Wireless Sensor network is one of the key component of CPS wherein the intelligence would be built with multiple dimensions of sensing data across multiple sensor networks and the Internet. Within a CPS, a

WSN is able to participate and leave in dynamic fashion wherein the activation-deactivation of sensors may also be mission-specific. CPS would involve intra-WSN, cross-domain communications with varying level of coverage and connectivity for different WSNs. A mix of static as well as dynamic sensor nodes with controlled and uncontrolled mobility may be used to collect data in a CPS application. In order to make proper use of the intelligence gathered from the sensing data, emphasis on obtaining knowledge from multiple sensing domains would be given in CPS. Security and privacy would be a vital issue with CPS given that the sensing data is gathered from multiple WSNs. In our future research endevours, we would like to explore and identify the security threats in WSNs in the context of cyber physical systems and work towards providing effective and efficient solutions to resolve the identified security issues.

Bibliography

[1] Agrawal, S. and Das, M. L. "Mutual Healing Enabled Group-key Distribution Protocol in Wireless Sensor Networks". In: *Elsevier Journal of Computer Communications* , Vol. 112(C) (2017).

[2] Agrawal, S. and Das, M. L. "Node Revocation and Key Update Protocol in Wireless Sensor Networks". In: *Proceedings of IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)* , pp. 1–6 (2016).

[3] Agrawal, S., Patel, J., and Das, M. L. "Pairing Based Mutual Healing in Wireless Sensor Networks". In: *Prooceeding of 8th IEEE International Conference on Communication Systems and Networks (COMSNETS)* , pp. 1–8 (2016).

[4] Agrawal, S., Roman, R., Das, M. L., Mathuria, A., and Lopez, J. "A Novel Key Update Protocol in Mobile Sensor Networks". In: *Proceeding of 12th International Conference on Information Systems Security* , Vol. Springer LNCS 7671: 194–207 (2012).

[5] Agrawal, S., Das, M. L., Mathuria, A., and Srivastava, S. "Program Integrity Verification for Detecting Node Capture Attack in Wireless Sensor Network". In: *Proceedings of 11th International Conference on Information Systems Security (ICISS)* , Vol. Springer LNCS 9478: 419–440 (2015).

[6] Akyildiz, I., Su, W., Sankarasubramaniam, Y., and Cayirci, E. "Wireless Sensor Networks: A Survey". In: *Journal of Computer Networks* , Vol. 38(4): 393–422 (2002).

[7] Albrecht, M., Gentry, C., Halevi, S., and Katz, J. "Attacking Cryptographic Schemes Based on "Perturbation Polynomials"". In: *Proceedings of the 16th*

ACM conference on Computer and Communications Security (CCS) , pp. 1–10 (2009).

[8] *Arduino Platform*. Accessed: 2016-10-11. URL: https://www.arduino.cc.

[9] Atakli, I., Hu, H., Chen, Y., Ku, W., and Su, Z. "Malicious Node Detection in Wireless Sensor Networks using Weighted Trust Evaluation". In: *Proceedings of the 2008 Spring Simulation Multiconference* , pp. 836–843 (2008).

[10] Becher, A., Benenson, Z., and Dornseif, M. "Tampering with Motes: Real-world Physical Attacks on Wireless Sensor Networks". In: *Proceedings of the Third International Conference on Security in Pervasive Computing (SPC)* , Vol. Springer LNCS 3934: 104–118 (2006).

[11] Bellare, M. and Rogaway, P. *Pseudorandom Functions*. Accessed: 2017-17-02. URL: https://cseweb.ucsd.edu/~mihir/cse207/w-prf.pdf.

[12] Benenson, Z., Cholewinski, P., and Felix, C. "Vulnerabilities and Attacks in Wireless Sensor Networks". In: *Wireless Sensor Network Security - Cryptology and Information Security Series* , Vol. 1: 22–43 (2007).

[13] Bhaskar, P. and Pais, A. "A Chinese Remainder Theorem Based Key Management Algorithm for Hierarchical Wireless Sensor Network". In: *International Conference on Distributed Computing and Internet Technology (ICDCIT)* , Vol. Spinger LNCS 8956: 311–317 (2015).

[14] Blanchet, B., Smyth, B., and Cheval, V. *ProVerif 1.88: Automatic Cryptographic Protocol Verifier: ProVerif User Manual and Tutorial (2013)*. URL: https://www.bensmyth.com/files/ProVerif-manual-version-1.88.pdf.

[15] Blom, R. "An Optimal Class of Symmetric Key Generation Systems". In: *Proceedings Of the EUROCRYPT 84 Workshop on Advances in Cryptology: Theory and Application of Cryptographic Techniques* , pp. 335–338 (1985).

[16] Blundo, C., D'Arco, P., Santis, A. De, and Listo, M. "Design of Self-Healing Key Distribution Schemes". In: *Designs, Codes and Cryptography: An International journal* , Vol. 32(1): 15–44 (2004).

[17] Blundo, C., Santis, A. De, Herzberg, A., Kutten, S., Vaccaro, U., and M. Yung, Moti. "Perfectly-Secure Key Distribution for Dynamic Conferences". In: *Proceedings of 12th Annual International Cryptology - Advances in Cryptology* , pp. 471–486 (1993).

[18] Bonaci, T., Bushnell, L., and Poovendran, R. "Node Capture Attacks in Wireless Sensor Networks: A System Theoretic Approach". In: *Proceedings of IEEE 49th International Conference on Decision and Control* , pp. 6765–6772 (2010).

[19] Boneh, D. and Franklin, M. "Identity-based Encryption from the Weil Pairing". In: *Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology* , LNCS Vol. 2139: 213–229 (2001).

[20] Boulis, A. *Castalia: A simulator for Wireless Sensor Networks and Body Area Networks: Castalia User's Manual (Ver - 3.2(2011).* URL: https://www.scribd.com/document/78901825/Castalia-User-Manual.

[21] Brown, E., Errthum, E., and Fu, D. *Weil Pairing vs. Tate Pairing in IBE Systems*. Accessed: 18 Jul 2017. URL: http://course1.winona.edu/eerrthum/Papers/WeilVsTate.pdf.

[22] Cao, Z. and Liu, L. "On the Disadvantages of Pairing-based Cryptography". In: *IACR Cryptology ePrint Archive* , Vol. 2015: 84–92 (2015).

[23] Chan, H., Perrig, A., and Song, D. "Random Key Predistribution Schemes for Sensor Networks". In: *Proceedings of IEEE Symposium on Security and Privacy* , pp. 197–213 (2003).

[24] Chan, H., Gligor, V., Perrig, A., and Muralidharan, G. "On the Distribution and Revocation of Cryptographic Keys in Sensor Networks". In: *IEEE Transactions on Dependable and Secure Computing* , Vol. 2(3): 233–247 (2005).

[25] Chang, K. and Shin, K. "Distributed Authentication of Program Integrity Verification in Wireless Sensor Networks". In: *ACM Transactions on Information and System Security (TISSEC)* , Vol. 11(3): 1–35 (2006).

[26] Chao, C., Yang, C., Lin, P., and Li, J. "Novel Distributed Key Revocation Scheme for Wireless Sensor Networks". In: *Journal of Security and Communication Networks* , Vol. 6(10): 1271–1280 (2013).

[27] Chattopadhyay, S. and Turuk, A. "A Scheme for Key Revocation in Wireless Sensor Networks". In: *International Journal on Advanced Computer Engineering and Communication Technology* , Vol. 1(2): 16–20 (2012).

[28] Cheikhrouhou, O. "Secure Group Communication in Wireless Sensor Networks: A Survey". In: *Journal of Network and Computer Applications* , Vol. 61: 115–132 (2015).

[29] Chen, X., K. Makki, K. Yen, and Pissinou, N. "Sensor Network Security: A Survey". In: *Proceedings of IEEE Communications Surveys and Tutorials* , Vol. 11(2): 57–73 (2009).

[30] Choi, Y., Kang, J., and Nyang, D. "Proactive Code Verification Protocol in Wireless Sensor Network". In: *Proceedings of the 2007 International Conference on Computational Science and Its Applications (ICCSA)* , pp. 1085–1096 (2007).

[31] Chuang, P., Chang, S., and Lin, C. "A Node Revocation Scheme Using Public-Key Cryptography in Wireless Sensor Networks". In: *Journal of Information Science and Engineering* , Vol. 26: 1859–1873 (2010).

[32] Conti, M. "Capture Detection". In: *Secure Wireless Sensor Networks* , Vol. 65: 53–73 (2016).

[33] Conti, M., Pietro, R., Mancini, L., and Mei, A. "Emergent Properties: Detection of the Node-Capture Attack in Mobile Wireless Sensor Networks". In: *Proceedings of the First ACM Conference on Wireless Network Security (WiSec)* , pp. 214–219 (2008).

[34] Dang, Q. "Recommendation for Applications Using Approved Hash Algorithms". In: *NIST Special Publication 800-107, Revision 1* (2012).

[35] Davis, P. "Interpolation". In: *Interpolation and Approximation, published by Dover Publications* , pp. 24–55 (1975).

[36] De, P., Liu, Y., and Das, S. K. "Deployment Aware Modeling of Node Compromise Spread in Wireless Sensor Networks". In: *ACM Transactions on Sensor Networks* , Vol. 5(3): 413–425 (2009).

[37] Delgosha, F. and Fekri, F. "A Multivariate Key Establishment Scheme for Wireless Sensor Networks". In: *IEEE Transactions on Wireless Communications* , Vol. 8(4): 1814–1824 (2009).

[38] Diffie, W. and Hellman, M. "New directions in Cryptography". In: *IEEE Transactions on Information Theory* , Vol. 22(6): 644–654 (1976).

[39] Ding, W., Laha, B., and Yenduri, S. "First Stage Detection of Compromised Nodes in Sensor Networks". In: *Proceedings of IEEE Sensors Applications Symposium* , pp. 20–24 (2010).

[40] Dolev, D. and Yao, A. "On the Security of Public Key Protocols". In: *IEEE Transactions on Information Theory* , Vol. 29(12): 198–208 (1983).

[41] Dutta, R., Mukhopadhyay, S., and Dowling, T. "Enhanced Access Polynomial based Self-healing Key Distribution". In: *Security in Emerging Wireless Communication and Networking Systems* , LNICS Vol. 42: 13–24 (2010).

[42] Eschenauer, L. and Gligor, V. "A Key-Management Scheme for Distributed Sensor Networks". In: *Proceedings of the 9th ACM Conference on Computer and Communications Security* , pp. 41–47 (2002).

[43] Gao, Q. "The Chinese Remainder Theorem And The Prime Memory System". In: *Proceedings of the 20th Annual International Symposium on Computer Architecture* , pp. 337–340 (1993).

[44] Ge, M. and Choo, K. "A Novel Hybrid Key Revocation Scheme for Wireless Sensor Networks". In: *Proceedings of 8th International Conference on Network and System Security (NSS)* , pp. 462–475 (2014).

[45] Ge, M., Choo, K., Wu, H., and Yu, Y. "Survey on Key Revocation Mechanisms in Wireless Sensor Networks". In: *Journal of Network and Computer Applications* , Vol. 63(C): 24–38 (2016).

[46] Ghafoor, A., Sher, M., Imran, M., and Saleem, K. "A Lightweight Key Freshness Scheme for Wireless Sensor Networks". In: *Proceedings of 12th International Conference on Information Technology - New Generations* , pp. 169–173 (2015).

[47] Guo, S., Leung, V., and Qian, Z. "A Permutation-Based Multi-Polynomial Scheme for Pairwise Key Establishment in Sensor Networks". In: *Proceedings of IEEE International Conference on Communications (ICC)* , pp. 1–5 (2010).

[48] Guo, S. and Qian, Z. "A Compromise Resilient Pair-wise Rekeying Protocol in Hierarchical Wireless Sensor Networks". In: *Smart Wireless Sensor Networks* , Vol. InTechOpen 18: 315–326 (2010).

[49] Gupta, V., Millard, M., Fung, S., Zhu, Y., Gura, N., Eberle, H., and Shantz, S. C. "Sizzle: A Standards-based End-to-End Security Architecture for the Embedded Internet". In: *Proceedings of 3rd IEEE International Conference on Pervasive Computing and Communications* , pp. 247-256 (2005).

[50] Hayouni, H. and Hamdi, M. "Energy Efficient Key Management Scheme for Clustered Hierarchical Wireless Sensor Networks". In: *Proceedings of the IEEE 12th International Conference on Networking, Sensing and Control* , pp. 105–109 (2015).

[51] He, T., Krishnamurthy, S., Luo, L., Yan, T., Gu, L., Stoleru, R., Zhou, G., Cao, Q., Vicaire, P., Stankovic, J., Abdelzaher, T., Hui, J., and Krogh, B. "VigilNet: An Integrated Sensor Network System for Energy-efficient Surveillance". In: *ACM Transactions on Sensor Networks* , Vol. 2(1): 1–38 (2006).

[52] Ho, J. "Distributed Detection of Node Capture Attacks in Wireless Sensor Networks". In: *Smart Wireless Sensor Networks* (2010).

[53] Hong, D. and Kang, J. "An Efficient Key Distribution Scheme with Self-Healing Property". In: *Journal of Communications* , Vol. 9(8): 759–761 (2005).

[54] *IRIS Datasheet*. Accessed: 2017-15-02. URL: http://www.memsic.com/userfiles/files/Datasheets/WSN/IRIS_Datasheet.pdf.

[55] Jiang, Y., Zhang, R., and Du, X. "A New Efficient Random Key Revocation Protocol for Wireless Sensor Networks". In: *Proceedings of International Conference on Parallel and Distributed Computing, Applications and Technologies* , pp. 233–238 (2013).

[56] Juang, P., Oki, H., Wang, Y., Martonosi, M., Peh, L., and Rubenstein, D. "Energy-efficient Computing for Wildlife Tracking: Design Tradeoffs and Early Experiences with ZebraNet". In: *ACM SIGPLAN Notices: Session on Emerging Systems* , Vol. 37(10): 96–107 (2002).

[57] Junior, W., Hao, T., Wong, C., and Loureiro, A. "Malicious Node Detection in Wireless Sensor Networks". In: *Proceedings of 18th International Parallel and Distributed Processing Symposium* , Vol. 4: 24–30 (2004).

[58] Karlof, C., Sastry, N., and Wagner, D. "TinySec: A Link Layer Security Architecture for Wireless Sensor Networks". In: *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems* , pp. 162–175 (2004).

[59] Katz, J. and Lindell, Y. "Private-key Encryption". In: *Introduction to Modern Cryptography, Chapman & Hall/CRC - Cryptography and Network Security Series, ISBN: 978-1-4665-7026-9* , pp. 77–79 (2015).

[60] Khiabani, H., Idris, N., and Manan, J. "Leveraging Remote Attestation to Enhance the Unified Trust Model for WSNs". In: *Proceedings of International Conference on Cyber Security* , pp. 139–143 (2012).

[61] Kim, D., Sadi, M., and J, Park. "DLS : Dynamic Level Session Key Revocation Protocol for Wireless Sensor Networks". In: *Proceedings of International Conference on Information Science and Applications (ICISA)* , pp. 1–8 (2010).

[62] Kim, D., Sadi, M., and Park, J. "A Key Revocation Scheme for Mobile Sensor Networks". In: *Proceedings of Frontiers of High Performance Computing and Networking ISPA 2007 Workshops* , pp. 41–49 (2007).

[63] Kim, J., Caytiles, R., and Kim, K. "A Review of the Vulnerabilities and Attacks for Wireless Sensor Networks". In: *Journal of Security Engineering* , Vol. 9(3): 241–250 (2012).

[64] Kim, J., Han, Y., Park, S., and Chung, T. “N-Dimensional Grid-Based Key Predistribution in Wireless Sensor Networks”. In: *Proceedings of International Conference on Computational Science and Its Applications* , pp. 1107–1120 (2007).

[65] Koblitz, N. “Elliptic Curve Cryptosystems”. In: *Mathematics of Computation* , Vol. 48(177): 203–209 (1987).

[66] Koblitz, N. and Menezes, A. “Pairing-Based Cryptography at High Security Levels”. In: *Proceedings of 10th IMA International Conferenc on Cryptography and Coding* , Vol. Springer LNCS 3796: 13–36 (2005).

[67] Krauß, C., Stumpf, F., and Eckert, C. “Detecting Node Compromise in Hybrid Wireless Sensor Networks using Attestation Techniques”. In: *Proceedings of 4th European Workshop on Security and Privacy in Ad-hoc and Sensor Networks (ESAS)* , pp. 203–217 (2007).

[68] Kumar, V. and Das, M. L. “Securing Wireless Sensor Networks with Public Key Techniques”. In: *Journal of Ad Hoc and Sensor Wireless Networks* , Vol. 5: 189-201 (2007).

[69] Kyungah, S. “The Risks of Compromising Secret Information”. In: *Proceedings of 4th International Conference on Information and Communications Security (ICICS)* , pp. 122–133 (2002).

[70] Lamont, L., Toulgoat, M., Deziel, M., and Patterson, G. “Tiered Wireless Sensor Network Architecture for Military Surveillance Applications”. In: *Proceedings of Fifth International Conference on Sensor Technologies and Applications (SENSORCOMM)* , pp. 288–294 (2011).

[71] Lazos, L. and Poovendran, R. “SeRLoc: Secure Range-independent Localization for Wireless Sensor Networks”. In: *Proceedings of the 3rd ACM Workshop on Wireless Security (WiSe)* , pp. 21–30 (2004).

[72] Li, T., Song, M., and Alam, M. “Compromised Sensor Nodes Detection: A Quantitative Approach”. In: *Proceedings of 28th International Conference on Distributed Computing Systems Workshops* , pp. 352–357 (2008).

[73] Li, X. and Yang, D. “A Quantitative Survivability Evaluation Model for Wireless Sensor Networks”. In: *Proceedings of the IEEE International Conference on Networking, Sensing and Control* , pp. 727–732 (2006).

[74] Liao, Y., Lei, C., and Wang, A. “A Robust Grid-Based Key Predistribution Scheme for Sensor Networks”. In: *Proceedings of 4th International Conference on Innovative Computing, Information and Control (ICICIC)* , pp. 760–763 (2009).

[75] Lin, C. and Wu, G. “Enhancing the Attacking Efficiency of the Node Capture Attack in WSN: A Matrix Approach”. In: *Journal of Supercomputing* , Vol. 66(2): 989–1007 (2013).

[76] Lin, X. “CAT: Building Couples to Early Detect Node Compromise Attack in Wireless Sensor Networks”. In: *Proceedings of IEEE Global Telecommunications Conference (GLOBECOM)* , pp. 7–12 (2009).

[77] Liu, D., Ning, P., and Du, W. “Detecting Malicious Beacon Nodes for Secure Location Discovery in Wireless Sensor Networks”. In: *Proceedings of 25th IEEE International Conference on Distributed Computing Systems (ICDCS)* 609-619 (2005).

[78] Liu, D., Ning, P., and Li, R. “Establishing Pairwise Keys in Distributed Sensor Networks”. In: *ACM Transactions on Information and System Security (TISSEC)* , Vol. 8(1): 41–77 (2005).

[79] Liu, D., Ning, P., and Sun, K. “Efficient Self-healing Group Key Distribution with Revocation Capability”. In: *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS)* , pp. 231–240 (2003).

[80] Liu, Y., Harn, L., and Chang, C. “An Authenticated Group Key Distribution Mechanism Using Theory of Numbers”. In: *International journal of Communication Systems* , Vol. 27(11): 3502–3512 (2014).

[81] Lopez, J., Roman, R., and Alcaraz, C. “Analysis of Security Threats, Requirements, Technologies and Standards in Wireless Sensor Networks”. In: *Tutorial Lectures on Foundations of Security Analysis and Design V* , Vol. Springer LNCS 5705: 289–338 (2009).

[82] Mainwaring, A., Polastre, J., Szewczyk, R., and Culler, D. "Wireless Sensor Networks for Habitat Monitoring". In: *Proceedings of 1st ACM Workshop on Sensor Networks and Applications* , pp. 88–97 (2002).

[83] Mall, D., Konaté, K., and Pathan, A. "Key Revocation in Wireless Sensor Networks: A Survey on a Less-addressed Yet Vital Issue". In: *International Journal of Ad Hoc Ubiquitous Computing* , Vol. 18(1/2): 3–22 (2015).

[84] Mansour, I., Chalhoub, G., and Lafourcade, P. "Key Management in Wireless Sensor Networks". In: *Journal of Sensor and Actuator Networks* , Vol. 4: 251-273 (2015).

[85] Mansour, I., Chalhoub, G., Lafourcade, P., and Delobel, F. "Secure Key Renewal and Revocation for Wireless Sensor Networks". In: *Proceedings of 39th Annual IEEE Conference on Local Computer Networks* , pp. 382–385 (2014).

[86] Mathews, M., Song, M., Shetty, S., and McKenzie, R. "Detecting Compromised Nodes in Wireless Sensor Networks". In: *Proceedings of 8th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing* , pp. 273–278 (2007).

[87] M.Conti, Pietro, R, Mancini, L., and Mei, A. "Mobility and Cooperation to Thwart Node Capture Attacks in MANETs". In: *EURASIP Journal on Wireless Communications and Networking* , Vol. 2009: 8:1–8:13 (2009).

[88] Menezes, A., Vanstone, S., and Oorschot, P. In: *Handbook of Applied Cryptography, published by CRC Press, Inc., ISBN: 0849385237* , pp. 321–383 (1996).

[89] Merkle, R. "Protocols for Public Key Cryptography". In: *Synopsis on Security and Privacy* , pp. 122–134 (1980).

[90] Mignotte, M. "How to Share a Secret". In: *Proceedings of the Workshop on Cryptography* , pp. 371–375 (1982).

[91] Miller, S., Neuman, B., Schiller, J., and Saltzer, J. "Kerberos Authentication and Authorization System". In: *Project Athena Technical Plan* (1987).

[92] Mishra, A. K. and Turuk, A. K. "A Comparative Analysis of Node Replica Detection Schemes in Wireless Sensor Networks". In: *Journal of Network and Computer Applications* , Vol. 61: 21–32 (2016).

[93] Mishra, A. K. and Turuk, A. K. "Adversary Information Gathering Model for Node Capture Attack in Wireless Sensor Networks". In: *Proceedings of International Conference on Devices and Communications (ICDeCom)* , pp. 1–5 (2011).

[94] Mittal, R., Agrawal, S., and Das, M. L. "Secure Node Localization in Clustered Sensor Networks with Effective Key Revocation". In: *Emerging Innovations in Wireless Networks and Broadband Technologies, published by IGI Global* , pp. 12–41 (2016).

[95] Moore, T., Clulow, J., Nagaraja, S., and Anderson, R. "New Strategies for Revocation in Ad-Hoc Networks". In: *Proceedings of 4th European Workshop on Security and Privacy in Ad-hoc and Sensor Networks* , pp. 232–246 (2007).

[96] Nasiraee, H., Bagherzadeh, J., and Nasiraee, M. "A New Self-healing Group Key Distribution Scheme". In: *Proceedings of the 12th International Iranian Society of Cryptology Conference on Information Security and Cryptology(ISCISC)* , pp. 85–90 (2015).

[97] Ochir, O., Minier, M., Valois, F., and Kountouris, A. *Resilient networking in wireless sensor networks*. Accessed: 18 Jul 2017. URL: http://arxiv.org/abs/1003.5104.

[98] Paar, C. and Pelzl, C. "Introduction to Cryptography and Data Security". In: *Understanding Cryptography: A Textbook for Students and Practitioners, Published by Springer, ISBN: 978-3-642-04100-6* , pp. 1–27 (2010).

[99] Park, T. and K.Shin. "Soft Tamper-Proofing via Program Integrity Verification in Wireless Sensor Networks". In: *IEEE Transactions on Mobile Computing* , Vol. 4(3): 297–309 (2005).

[100] Perrig, A., Szewczyk, R., Wen, V., Culler, D., and Tygar, J. "SPINS : Security Protocols for Sensor Networks". In: *Journal of Wireless Networks* , Vol. 8: 521-534 (2002).

[101] Peyraviana, M. and Kshemkalyanib, A. "On Probabilities of Hash Value Matches". In: *Elsevier Journal of Computers and Security* , Vol. 17(2): 171–176 (1998).

[102] Pietro, R., Ma, D., Soviente, C., and Tsudik, G. "Self-healing in Unattended Wireless Sensor Networks". In: *ACM Transactions on Sensor Networks* , Vol. 9(1): 7 (2013).

[103] Pottie, G. and Kaiser, W. "Wireless Integrated Network Sensors". In: *Communications of the ACM* , Vol. 43(5): 551–558 (2000).

[104] Rams, T. and Pacyna, P. "A Survey of Group Key Distribution Schemes with Self-healing Property". In: *IEEE Communications Surveys and Tutorials* , Vol. 15(2): 820–842 (2013).

[105] Rams, T. and Pacyna, P. "Self-healing Group Key Distribution with Extended Revocation Capability". In: *Proceedings of International Conference on Computing, Networking and Communications (ICNC)* , pp. 347–353 (2013).

[106] *RaspBerry Pi Node*. Accessed: 2017-15-02. URL: https://www.raspberrypi.org/products/.

[107] Rivest, R., Shamir, A., and Adleman, L. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems". In: *Magazine: Communications of the ACM* , Vol. 21(2): 120–126 (1978).

[108] Roman, R., Lopez, J., Alcaraz, C., and Chen, H. "SenseKey–Simplifying the Selection of Key Management Schemes for Sensor Networks". In: *Proceedings of IEEE Workshops of International Conference onAdvanced Information Networking and Applications (WAINA)* , pp. 789–794 (2011).

[109] Sadi, M., Park, J., and Kim, D. "Randomized Grid Based Scheme for Wireless Sensor Network". In: *Proceedings of the Second European Conference on Security and Privacy in Ad-Hoc and Sensor Networks (ESAS)* , Vol. Springer LNCS Vol. 3813: 91–101 (2005).

[110] Selavo, L., Wood, A., Cao, Q., Sookoor, T., Liu, H., Srinivasan, A., Wu, Y., Kang, W., Stankovic, J., Young, D., and Porter, J. "LUSTER: Wireless Sensor Network for Environmental Research". In: *Proceedings of the 5th International Conference on Embedded Networked Sensor Systems* , pp. 103–116 (2007).

[111] Seshadri, A., Luk, M., Perrig, A., Doorn, L., and Khosla, P. "SCUBA: Secure Code Update By Attestation in Sensor Networks". In: *Proceedings of ACM workshop on Wireless Security (WiSe)* , pp. 85–94 (2006).

[112] Seshadri, A., Perrig, A., Doorn, L., and Khosla, P. "SWATT: SoftWare-based ATTestation for Embedded Devices". In: *Proceedings of IEEE Symposium on Security and Privacy* , pp. 272-282 (2004).

[113] Shi, E. and Perrig, A. "Designing Secure Sensor Networks". In: *IEEE Wireless Communications* , Vol. 11(6): 38–43 (2004).

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

[115] Tague, P. and Poovendran, R. "Modeling Adaptive Node Capture Attacks in Multi-hop Wireless Networks". In: *Journal of Ad Hoc Networks* , Vol. 5(6): 801–814 (2007).

[116] Tague, P. and Poovendran, R. "Modeling Node Capture Attacks in Wireless Sensor Networks". In: *Proceedings of IEEE 46th Annual Allerton Conference on Communication, Control, and Computing* , pp. 1221–1224 (2008).

[117] Tan, H., Hu, W., and Jha, S. "A TPM-enabled Remote Attestation Protocol (TRAP) in Wireless Sensor Networks". In: *Proceedings of the 6th ACM workshop on Performance Monitoring and Measurement of Heterogeneous Wireless and Wired Networks (PM2HW2N)* , pp. 9–16 (2011).

[118] *The Secure Sockets Layer (SSL) Protocol Version 3.0*. Accessed: 2017-17-02. URL: https://tools.ietf.org/html/rfc6101?ref=driverlayer.com.

[119] Tian, B., Han, S., and Dillon, T. "An Efficient Self-Healing Key Distribution Scheme". In: *Proceedings of International Conference on New Technologies, Mobility and Security* , pp. 1–5 (2008).

[120] Tian, B., Han, S., Hu, J., and Dillon, T. "A Mutual-healing Key Distribution Scheme in Wireless Sensor Networks". In: *Journal of Network and Computer Applications* , Vol. 34(1): 80–88 (2011).

[121] Tian, B., Chang, E., Dillon, T., Han, S., and Hussain, F. "An Authenticated Self-healing Key Distribution Scheme based on Bilinear Pairings". In: *Proceedings of 6th IEEE Consumer Communications and Networking Conference (CCNC)* , pp. 1–5 (2009).

[122] Tomlinson, A. "Introduction to the TPM". In: *Smart Cards, Tokens, Security and Applications* , pp. 155–172 (2008).

[123] Trivedi, K., Kim, D., and Ghosh, R. "Resilience in Computer Systems and Networks". In: *Proceedings of the International Conference on Computer-Aided Design (ICCAD)* , pp. 74–77 (2009).

[124] Wang, F., Chang, C., and Chou, Y. "Group Authentication and Group Key Distribution for Ad Hoc Networks". In: *International Journal of Network Security* , Vol. 17(2): 199–207 (2015).

[125] Wang, H. "On the Security of Some Self-healing Key Distribution Schemes". In: *Proceedings of the International Conference on Multimedia Information Networking and Security* , pp. 777–780 (2010).

[126] Wang, Y. and Ramamurthy, B. "KeyRev : An Efficient Key Revocation Scheme for Wireless Sensor Networks". In: *Proceeding of IEEE International Conference on Communications* , pp. 1260–1265 (2007).

[127] Wang, Y., Ramamurthy, B., and Xue, Y. "A Key management Protocol for Wireless Sensor Networks with Multiple Base Stations". In: *Proceedings of IEEE International Conference on Communications* , pp. 1–6 (2008).

[128] *WASPMote*. Accessed: 2017-15-02. URL: http://www.libelium.com/products/waspmote/.

[129] Watro, R., Kong, D., Cuti, S., Gardiner, C., Lynn, C., and Kruus, P. "TinyPK: Securing Sensor Networks with Public Key Technology". In: *Proceedings of the 2nd ACM Workshop on Security of Ad Hoc and Sensor Networks* , pp. 59–64 (2004).

[130] Wood, A., Virone, G., Doan, T., Cao, Q., Selavo, L., Wu, Y., Fang, L., He, Z., Lin, S., and Stankovic, J. "ALARM-NET: Wireless Sensor Networks for Assisted-living and Residential Monitoring". In: *Technical Report of Wireless Sensor Network Research Group, Department of Computer Science, University of Virginia* (2006).

[131] Yang, Y., Zhou, J., Deng, R.H., and Bao, F. "Better Security Enforcement in Trusted Computing Enabled Heterogeneous Wireless Sensor Networks". In: *Journal of Security and Communication Networks* , Vol. 4: 11–22 (2011).

[132] Yu, Y., Michael, F., Bernhard, P., Paul, S., and Alberto, F. "Resilience Strategies for Networked Malware Detection and Remediation". In: *Proceedings of Network and System Security* , pp. 233–247 (2012).

[133] Yuan, T., Jianqing, M., Zhong, Y., and Zhang, S. "Self-healing Key Distribution with Limited Group Membership Property". In: *Proceedings of First International Conference on Intelligent Networks and Intelligent Systems* , pp. 309–312 (2008).

[134] Yuan, T., Jianqing, M., Zhong, Y., and Zhang, S. "Self-healing Key Distribution with Revocation and Collusion Resistance for Wireless Sensor Networks". In: *Proceedings of International Multi-symposiums on Computer and Computational Sciences* , pp. 83–90 (2008).

[135] Zhang, F., Naini, R., and Susilo, W. "An Efficient Signature Scheme from Bilinear Pairings and Its Applications". In: *Proceedings of 7th International Workshop on Theory and Practice in Public Key Cryptography* , pp. 277–290 (2004).

[136] Zhang, Q., Yu, T., and Ning, P. "A Framework for Identifying Compromised Nodes in Wireless Sensor Networks". In: *ACM Transactions on Information and System Security (TISSEC)* , Vol. 11(3): 12:1–12:37 (2008).

[137] Zhang, W., Tran, M., Zhu, S., and Cao, G. "A Random Perturbation-based Scheme for Pairwise Key Establishment in Sensor Networks". In: *Proceedings of the 8th ACM international symposium on Mobile Ad-hoc Networking and Computing (MobiHoc)* , pp. 90–99 (2007).

[138] Zhang, X., HeEmail, J., and Wei, Q. "EDDK: Energy-Efficient Distributed Deterministic Key Management for Wireless Sensor Networks". In: *EURASIP Journal on Wireless Communications and Networking* , Vol. 2011(1): 765143:1–765143:11 (2010).

[139] Zhao, J. "On Resilience and Connectivity of Secure Wireless Sensor Networks Under Node Capture Attacks". In: *IEEE Transactions on Information Forensics and Security* , Vol. 12(3): 557–571 (2016).

[140] Zheng, J. and Jamalipour, A. "Introduction to Wireless Sensor Networks". In: *Wireless Sensor Networks: A Networking Perspective, Published by John Wiley and Sons, ISBN: 978-0-470-16763-2* , pp. 34–51 (2009).

[141] Zheng, X., Huang, C., and Matthews, M. "Chinese Remainder Theorem Based Group Key Management". In: *Proceedings of the 45th Annual Southeast Regional Conference* , pp. 266-271 (2007).

[142] Zhou, J. and Ou, Y. "Key Tree and Chinese Remainder Theorem Based Group-key Distribution Scheme". In: *Journal of the Chinese Institute of Engineers* , Vol. 32(7): 967–974 (2009).

[143] Zhu, S., Setia, S., and Jajodia, S. "LEAP: Efficient Security Mechanisms for Large-scale Distributed Sensor Networks". In: *Proceedings of 10th ACM Conference on Computer and Communications Security (CCS)* , pp. 62–72 (2003).

[144] Zia, T. and Zomaya, A. "Security Issues in Wireless Sensor Networks". In: *Proceedings of International Conference on Systems and Networks Communications (ICSNC)* , pp. 6–9 (2006).

[145] Zou, X. and Dai, Y. "A Robust and Stateless Self-Healing Group Key Management Scheme". In: *Proceedings of International Conference on Communication Technology* , pp. 1–4 (2006).

Appendix 1: ProVerif Tool

ProVerif is a tool for verification of cryptographic protocols. The cryptographic protocols, as concurrent programs, interact using public communication channels for achieving some objective related to security. It is assumed that these public channels are controlled by the attacker who can read, modify, delete, and inject messages. The attacker is also supposed to be having the ability of manipulating data, if it has the necessary keys. The environment can also capture a dishonest participant's behavior. The input language of ProVerif allows the cryptographic protocols and associated security goals to be coded in a formal manner. Then, ProVerif can automatically verify the claimed security features. With cryptography, it is considered that an attacker can perform cryptographic operations only when in possession of the secret keys used to secure the communication.

Proverif Script

A ProVerif script to test the protocol for authentication and confidentiality is given below:

```
type mkey.
type G.


const g : G[data].
```

```
fun exp(G, bitstring):G.
fun mac(bitstring, mkey): bitstring.
fun g_to_bs(G):bitstring.
fun bs_to_G(bitstring):G.
fun poly(bitstring, bitstring):mkey[private].

equation forall x:bitstring, y:bitstring;
exp(exp(g, x), y) = exp(exp(g, y), x).
equation forall x:bitstring, y:bitstring;
poly(x,y) = poly(y,x).

free c : channel.
free nA: bitstring[private].
free nB: bitstring[private].
free idA:bitstring.
free idB:bitstring.
free k1: bitstring[private].
free k2: bitstring[private].

event acceptsA(mkey).
event acceptsB(mkey).
event termA(mkey).
event termB(mkey).
```

```
query attacker(nA).
query attacker(nB).
query attacker(idA).
query attacker(idB).
query attacker(k1).
query attacker(k2).
query x:mkey; event(termA(x)) ==> event(acceptsB(x)).
query x:mkey; event(termB(x)) ==> event(acceptsA(x)).

(* Process for node A *)

let NodeA() = out(c, idA);
(* 1. A sends out its ID in plain *)

in(c, (idX:bitstring, mgX:bitstring, macX:bitstring));
(* 6. A receives a response from some node X *)

let kAX = poly(idA, idX) in
(* 7. A computes shared key with responding node X *)

let macA = mac((idX, mgX), kAX) in
(* 8. A computes mac using the supplied values and *)
(* check with the supplied mac value *)

if macA = macX then
```

```
(* 9. if verifies, accpets B's response *)

let k1 = g_to_bs(exp(bs_to_G(mgX), nA)) in
event acceptsA(kAX);
(* 10. A computes the new shared key and *
(* accpet the shared key*)

let gA = exp(g, nA) in
(* 11. A now computes g^nA = gA using its nonce *)
(* nA and converts the result into bitstring type *)

let mgA = g_to_bs(gA) in
out(c, (idA, gA, mac((idA, mgA, mgX), kAX)));
event termA(kAX).
(* 12. A sends confirmation to node X and terminates *)

(* Process for node B *)
let NodeB() =
in(c, idX:bitstring);
(* 2. B receives a msg with some node ID *)

let kBX = poly(idB, idX) in
event acceptsB(kBX);
(* 3. B computes the shared polynomial based key *)
(* with node X and accepts to communicate with A*)
```

```
let gB = exp(g, nB) in
let mgB = g_to_bs(gB) in
(* 4. B takes a nonce value, computes g^nB = gB, *)
(* since it is of type G, converted into bitstring type *)

out(c, (idB, mgB, mac((idB, mgB), kBX)));
(* 5. B responds back with idB, gB, and mac of *)
(* both values using shared key kBX *)

in(c, (idX1:bitstring, mgX:bitstring, macX:bitstring));
(* 13. B receives confirmation from requesting node *)

let macB = mac((idX1, mgX, mgB), kBX) in
if macB = macX then
(* 14. B computes mac using the supplied values, *)
(* checks with supplied mac value *)

let k2 = g_to_bs(exp(bs_to_G(mgX), nB)) in
event termB(kBX);
0.
(* 15. B computes the new shared key and terminates *)

(* Main process execution *)
process
```

```
( NodeA() | NodeB() )
```

ProVerif Output

We present the result of the execution of the script given above as follows:

```
Linear part:
poly(x_12,y_13) = poly(y_13,x_12)
exp(exp(g,x),y) = exp(exp(g,y),x)
Completing equations...
Completed equations:
exp(exp(g,x),y) = exp(exp(g,y),x)
poly(x_12,y_13) = poly(y_13,x_12)
Convergent part:
Completing equations...
Completed equations:
Process:
(
{1}out(c, idA);
{2}in(c, (idX: bitstring,mgX: bitstring,macX: bitstring));
{3}let kAX: mkey = poly(idA,idX) in
{4}let macA: bitstring = mac((idX,mgX),kAX) in
{5}if (macA = macX) then
{6}let k1_46: bitstring =
   g_to_bs(exp(bs_to_G(mgX),nA)) in
{7}event acceptsA(kAX);
```

```
{8}let gA: G = exp(g,nA) in
{9}let mgA: bitstring = g_to_bs(gA) in
{10}out(c, (idA,gA,mac((idA,mgA,mgX),kAX)));
{11}event termA(kAX)
) | (
{12}in(c, idX_47: bitstring);
{13}let kBX: mkey = poly(idB,idX_47) in
{14}event acceptsB(kBX);
{15}let gB: G = exp(g,nB) in
{16}let mgB: bitstring = g_to_bs(gB) in
{17}out(c, (idB,mgB,mac((idB,mgB),kBX)));
{18}in(c, (idX1: bitstring,mgX_48: bitstring,
    macX_49: bitstring));
{19}let macB: bitstring = mac((idX1,mgX_48,mgB),kBX) in
{20}if (macB = macX_49) then
{21}let k2_50: bitstring =
    g_to_bs(exp(bs_to_G(mgX_48),nB)) in
{22}event termB(kBX)
)
-- Query event(termB(x_51)) ==> event(acceptsA(x_51))
Completing...
Starting query event(termB(x_51)) ==>
event(acceptsA(x_51)) goal reachable:
begin(acceptsA(poly(idB[],idA[]))) ->
end(termB(poly(idB[],idA[])))
```

```
RESULT event(termB(x_51)) ==>
event(acceptsA(x_51)) is true.
-- Query event(termA(x_832)) ==> event(acceptsB(x_832))
Completing...
Starting query event(termA(x_832)) ==>
event(acceptsB(x_832)) goal reachable:
begin(acceptsB(poly(idB[],idA[]))) ->
end(termA(poly(idB[],idA[])))
RESULT event(termA(x_832)) ==> event(acceptsB(x_832))
is true.
-- Query not attacker(k2[])
Completing...
Starting query not attacker(k2[])
RESULT not attacker(k2[]) is true.
-- Query not attacker(k1[])
Completing...
Starting query not attacker(k1[])
RESULT not attacker(k1[]) is true.
-- Query not attacker(idB[])
Completing...
Starting query not attacker(idB[])
goal reachable: attacker(idB[])
1. The attacker initially knows idB[].
attacker(idB[]).
A more detailed output of the traces is available with
```

```
set traceDisplay = long.
out(c, idA) at {1}
The attacker has the message idB.
A trace has been found.
RESULT not attacker(idB[]) is false.
-- Query not attacker(idA[])
Completing...
Starting query not attacker(idA[])
goal reachable: attacker(idA[])
1. The attacker initially knows idA[].
attacker(idA[]).
A more detailed output of the traces is available with
set traceDisplay = long.
out(c, idA) at {1}
The attacker has the message idA.
A trace has been found.
RESULT not attacker(idA[]) is false.
-- Query not attacker(nB[])
Completing...
Starting query not attacker(nB[])
RESULT not attacker(nB[]) is true.
-- Query not attacker(nA[])
Completing...
Starting query not attacker(nA[])
RESULT not attacker(nA[]) is true.
```

Appendix 2: Publications

Journals

S. Agrawal, M. L. Das and J. Lopez. "Detection of Node Capture Attack in Wireless Sensor Networks". (Manuscript submitted).

S. Agrawal and M. L. Das. "Mutual Healing enabled Group-key Distribution Protocol in Wireless Sensor Networks". (Manuscript Accepted for publication in Elsevier Journal of Computer Communications).

Conferences

S. Agrawal, M. L. Das. "Node Revocation and Key Update Protocol". In: *Proceedings of 10th IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)* (2016).

S. Agrawal, J. Patel and M. L. Das. "Pairing Based Mutual Healing in Wireless Sensor Networks". In: *Proceedings of 8th International Conference on Communication Systems and Networks (COMSNETS)*, pp. 1-

8 (2016).

S. Agrawal, M. L. Das, A. Mathuria and S. Srivastava. "Program Integrity Verification for Detecting Node Capture Attack in Wireless Sensor Network". In: *Proceedings of 11th International Conference on Information Systems Security (ICISS)*, Vol. Springer LNCS 9478: 419-440 (2015).

S. Agrawal, M. L. Das, R. Roman, A. Mathuria and J. Lopez. "A Novel Key Update Protocol in Mobile Sensor Networks". In: *Proceedings of 8th International Conference on Information Systems Security (ICISS)*, Vol. Springer LNCS 7671: 194-207 (2012).

Book Chapter

R. Mittal, S. Agrawal, and M. L. Das. "Secure Node Localization in Clustered Sensor Networks with Effective Key Revocation". In: *Emerging Innovations in Wireless Networks and Broadband Technologies, published by IGI Global*, pp. 12-41, (2016).