Security Mechanisms in Unattended Wireless Sensor Networks

Yi Ren

Security Mechanisms in Unattended Wireless Sensor Networks

A Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree of *Philosophiae Doctor (Ph.D.)* in Information and Communication Technology

University of Agder
Faculty of Engineering and Science
2011

Doctoral Dissertation by the University of Agder 39
ISBN: 978-82-7117-700-3
ISSN: 1504-9272

Printed in the Printing Office, University of Agder
Kristiansand

Preface and Acknowledgements

This dissertation is a result of my Ph.D. research work carried out at the Dept. of Information and Communication Technology (ICT), University of Agder (UiA), Norway, from Aug. 2007 to July 2011. During Dec. 2010 to Mar. 2011, I am a visiting researcher at University of Pittsburgh (PITT), USA.

Here, I would like to thank and acknowledge the support and encouragement I received from many people while walking towards the goal of a Ph.D. degree. It was a great experience for me to study at UiA where I had the opportunities to train my research skills in various aspects, from understanding, knowledge, motivation to discipline. This dissertation would not be possible without the help of many people I would like to thank.

First of all, I would like to express my sincere gratitude to my dear principal supervisor, Prof. Vladimir A. Oleshchuk, who guided me into my current research area, discussed with me about my ideas and proof-read my papers. I am very grateful for his nice, admirable patience and broad knowledge in cryptography and security. He was always there to support me through necessary comments and guidance. He is a nice and humorous person and never pushes my work. I would also like to thank my co-supervisor Prof. Frank Y. Li. He always discusses with me on the technique details and gave me many valuable advices on both my research and my career. He sets an example as a diligent researcher and inspirational supervisor, moreover, an example of being a responsible person.

I would like to thank the S2EuNet project (http://s2eunet.org/) financed by the European Commission under the 7th Framework Program (FP7) for supporting my research stay at PITT, as a visiting researcher hosted by Prof. Vladimir I. Zadorozhny. I would like to thank Prof. Zadorozhny for giving me the opportunity to have an immense learning experience.

In addition, I would like to acknowledge supports from all faculty at the Dept. of ICT. Especially, I wish to thank Prof. Frank Reichert (the Dean of the Faculty of Engineering and Science), Prof. Andreas Prinz (the Head of the ICT department), Mrs. Trine Tønnessen (the former PhD program coordinator), and Tor Erik Christensen (from International education office) for their timely support.

My sincere gratitude goes to all the doctoral fellows at the Dept. of ICT at UiA. I am really grateful for their friendship and cares from deep inside.

They made my stay in Grimstad very pleasant and I have never felt lonely in abroad because of these lovely persons. Especially, I also would like to take this opportunity to thank my dear friends: Ali Chelli, Chee Lim Nge, Fang Chen, Fengxia Yan, Hongzhi Jiao, Hui Zhang, Ke Yu, Lei Jiao, Liang Zhang, Linbo Qing, Liping Mu, Nils Ulltveit-Moe, Pei Li, Peipei Liu, Ram Kumar, Selo Sulistyo, Trinh Hoang Nguyen, Wenjuan Wang, Xianghan Zheng, Xin He, Xuan Zhang, Yuanyuan Ma and Ziaul Haq Abbas. I always appreciate for their unique camaraderie, selfless cares, and every joyful moment they shared with me.

I am specially indebted to my parents Zhengguo Ren and Chunmei Cheng. They always understand and support on my choices. They love me more than themselves and have sacrificed too much to support me.

Finally, I would like to acknowledge Lånekassen for funding my research for these four years.

Yi Ren
December 2011
Grimstad, Norway

Abstract

Wireless Sensor Networks (WSNs) consisting of a large number of sensor nodes are being deployed in potentially hostile environments for applications such as forest fire detection, battlefield surveillance, habitat monitoring, traffic management, etc. One common assumption in traditional WSNs is that a trusted third party, i.e., a sink, is assumed to be always available to collect sensed data in a real time or near real time fashion. Although many WSNs operate in such an on-site mode, there are WSN applications that do not fit into the real time data collection mode. For example, data collection in Unattended WSNs (UWSNs) relies on the periodical appearance of a mobile sink. As most existing security solutions developed for traditional WSNs rely on the presence of a trusted third party, it makes them not applicable to UWSNs directly. This motivates the research on security mechanisms for UWSNs.

This dissertation contributes to security mechanisms in UWSNs from three important aspects, as, confidentiality and reliability, trust management, and capture resistance. The first aspect addresses data confidentiality and data reliability in UWSNs. We propose a data distribution scheme to provide forward secrecy, probabilistic backward secrecy and data reliability. Moreover, we demonstrate that backward secrecy of the *historical* data can be achieved through homomorphic encryption and key evolution. Furthermore, we propose a constrained optimization algorithm to further improve the above two data distribution schemes.

The second study introduces trust management in UWSNs. We propose a set of efficient and robust trust management schemes for the case of UWSNs. The Advanced Scheme utilizes distributed trust data storage to provide trust data reliability and takes the advantages of both Geographic Hash Table (GHT) and Greedy Perimeter Stateless Routing (GPSR) to find storage nodes and to route trust data to them. In this way, it significantly reduces storage cost caused by distributed trust data storage and provides resilience to node compromise and node invalidation.

The third study investigates how to detect a captured node and to resist node capture attack in UWSNs. We propose a node capture resistance and key refreshing scheme for UWSNs based on the Chinese remainder theorem. The scheme is able to provide forward secrecy, backward secrecy and collusion resistance for diminishing the effects of capture attacks.

Contents

List of Figures

List of Tables

Abbreviations

$\mathcal{ADV}$	Mobile Adversary
$\mathcal{MS}$	Mobile Sink
APB	Adaptive Polynomial-Based
AS	Advanced Scheme
CDD	Cooperative Distributed Detection
CRT	Chinese Remainder Theorem
DISH	DIstributed Self-Healing
DSP	Data Survival Probability
EHB	Enhanced Hash-Based
ELK	Efficient Large-group Key
ES	Enhanced Scheme
GHT	Geographic Hash Table
GPSR	Greedy Perimeter Stateless Routing
HEHSS	Homomorphic Encryption and Homomorphic Secret Sharing based scheme
HKS	Homomorphic encryption and Key evolution based Scheme
ID	Identification
LKH	Logical Key Hierarchy
MAC	Message Authentication Code

MUWSN Mobile Unattended Wireless Sensor Network

NS Naive Scheme

ODDS Optimized Data Distribution Scheme

OFCT One-way Function Chain Tree

OFT One-way Function Trees

ONE-PARA trust consensus with ONE PARAmeter similarity threshold function

PDA Personal Digital Assistant

PKC Public Key Cryptography

POSH Proactive co-Operative Self-Healing

RSSS Reed-Solomon codes and Secret Sharing based scheme

SDD Simple Distributed Detection

SHB Simple Hash-Based

SKC Symmetric Key Cryptography

SO Storage Overhead

T-PARA trust consensus with Three PARAmeter similarity threshold function

T-PARA-WF Three PARAmeters with Weighted Factors

TC Transmission Cost

TC-ONLY Trust Consensus only Approach

UWSN Unattended Wireless Sensor Network

WSN Wireless Sensor Network

XOR eXclusive OR

Chapter 1

Introduction

In this chapter, we present the motivation and the contributions of this dissertation.

1.1 Background and Motivation

In recent years, the area of Wireless Sensor Networks (WSNs) has been extremely popular in both the research community and industries [1]. A WSN usually consists of a large amount of different types of sensor nodes that are able to monitor a wide variety of ambient conditions such as temperature, humidity, vehicular movement, pressure, etc. These sensor nodes can be easily deployed at a low cost for military and homeland security applications such as battlefield surveillance, as well as for civilian areas such as environment monitoring, E-health, and industrial automation [99, 98].

Once a WSN is deployed, sensors may generate a large volume of data during its lifetime. These large amounts of data have to be stored somewhere for future data retrieval and data analysis. The traditional way to collect data sensed by sensor nodes is that sensor nodes forward the data to a sink (or base station) immediately once they sense them, referred to as the sense-and-transmit mode.

However, the sense-and-transmit mode suffers from several drawbacks. Firstly, sensors around the sink may deplete their energy faster than the other sensors in the network since the near-sink sensors need not only to deliver their own data to the sink but also to forward data originated from many other sensors that are farther away from the sink. As a consequence, the near-sink sensors may exhaust their energy quicker and then lose their functionality totally.

Secondly, the near-sink sensors would attract more attackers than the others because once a small number of near-sink sensors are compromised or stop functionality, the network would be partitioned. Thirdly, in areas hostile to human beings such as volcanic areas, underwater zones, battle fields, animal habitats, etc., sensors are usually deployed by airplanes or helicopters, creating the predicament of imprecise sensor location and coverage uncertainty. Due to the aforementioned reasons, it may be infeasible to deploy a fixed sink (or base station). Alternatively, another form of WSNs, Unattended Wireless Sensor Networks (UWSNs) have been proposed recently [23, 24], in which sensors cannot off-load data to a sink at will or in *real-time* since an *on-site* sink or a base station in the network does not exist. Instead, as shown in Fig. 1.1, a mobile sink (denoted as $\mathcal{MS}$ hereafter), e.g., cell phones, Personal Digital Assistant (PDAs), robots, etc., visits the network periodically to collect data.

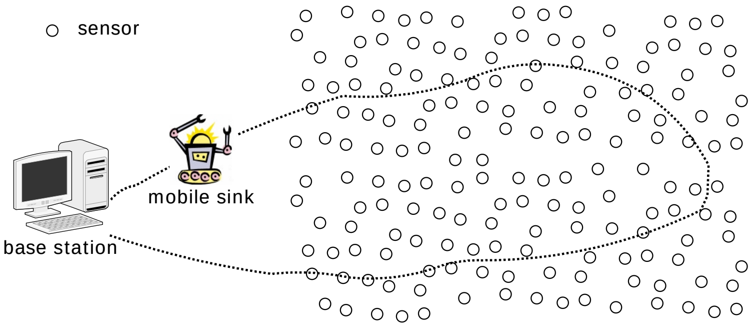


Figure 1.1: An illustration of UWSNs.

To guarantee the success of WSN applications, especially for those mission critical applications in hostile environments, many security mechanisms [70, 34, 50, 49, 81, 51, 69, 121, 39, 120, 36, 52, 92, 91] have been proposed. However, those schemes cannot be applied to UWSNs directly since they rely on the presence of an on-site trusted third party, i.e., a sink or a base station, for their mechanism design. These limitations motivate the design of security mechanisms in UWSNs [59, 23, 21, 103, 60, 57, 27, 58, 22, 24, 114, 25, 26, 28, 29, 53, 15]. A list of interesting topics for security in UWSNs are as follows:

1. How to provide data confidentiality? More specifically, forward secrecy and backward secrecy?

2. How to provide data reliability?

3. How to provide data integrity?

4. How to minimize resource consumption due to accumulated data? Specifically, data storage cost, Message Authentication Code (MAC) storage cost, and energy cost?
5. How to design light weight searchable encryption scheme for sensors?
6. How to design trust management schemes in UWSNs?
7. How to detect a captured sensor?
8. How to diminish the effects of compromised sensors?
9. How to protect the location privacy of an $\mathcal{MS}$?
10. How to avoid impersonating the behavior of a real $\mathcal{MS}$ in the event that it is captured (or comprised) by an adversary?
11. How to design the optimal travel route to collect accumulated sensor data with least security risk to $\mathcal{MS}$?

1.2 Security Concerns

In this thesis, we try to answer the questions 1, 2, 6, 7 and 8. At the beginning, we explain the security concerns: intrusion-resilient data security and node capture resistance.

1.2.1 Intrusion-resilient Data Security

Due to the absence of an on-site trusted third party in UWSNs, the accumulated data in sensors impose new security challenges - forward secrecy, backward secrecy and data reliability.

To shed more light on those security issues in UWSNs, we investigate the security implications for UWSNs versus traditional [1] WSNs under the mobile adversary model [60, 21]. In this model, each sensor generates data in each round $r \in R$. There is an $\mathcal{ADV}$ roaming in the network, compromising and releasing sensors to enrich its knowledge on all collected data. To illustrate the model, we assume that a mobile adversary (denoted as $\mathcal{ADV}$ hereafter) compromises a sensor at round r_a, releases the sensor at round r_b, and compromises it again at round r_c, where $r_a < r_b < r_c$, as shown in Fig. 1.2. The data

[1] Here, the term *traditional* means that there is at least one on-site sink in the network.

generated between rounds r_l and r_a is denoted as $\mathcal{D}(r_l, r_a)$, so as $\mathcal{D}(r_a, r_b)$ and $\mathcal{D}(r_b, r_c)$, respectively. Between rounds r_a and r_b, the $\mathcal{ADV}$ is residing in the sensor, and we refer to this time interval as *reside period* T_{rp}.

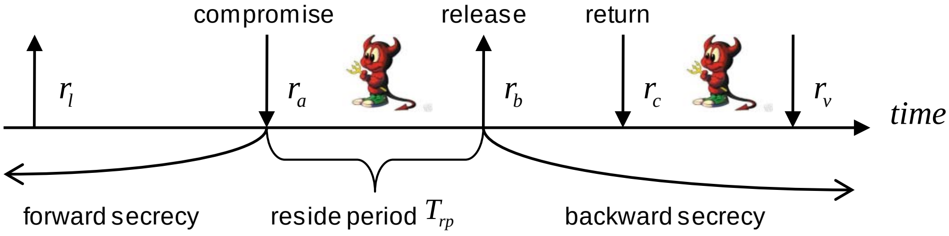


Figure 1.2: The mobile adversary model.

- In traditional WSNs, after generating data, the data are sent to and are kept in an on-site sink. The data generated before compromising (i.e., $\mathcal{D}(r_l, r_a)$) and between releasing and returning (i.e., $\mathcal{D}(r_b, r_v)$) are kept in the on-site sink, which is considered as trusted.
- In UWSNs, an $\mathcal{MS}$ collects data and leaves the network at round r_l, and comes back at round r_v to collect data again. The data generated between rounds r_l and r_v are accumulated and stored in the sensor locally. Consequently the data generated between rounds r_l and r_a (i.e., $\mathcal{D}(r_l, r_a)$) are obtained by the $\mathcal{ADV}$ if the sensor is compromised at round r_a. So do $\mathcal{D}(r_b, r_c)$ if the $\mathcal{ADV}$ compromises the sensor again at r_c.
- In this model, no matter whatever security mechanisms are employed, the data generated in the reside period T_{rp} are captured by the $\mathcal{ADV}$.

The security implications for traditional WSNs and UWSNs under the mobile adversary model are summarized in Table 1.1.

Table 1.1: Security implications under the mobile adversary model: traditional WSNs versus UWSNs.

	Traditional WSNs	UWSNs
$\mathcal{D}(r_l, r_a)$	sent to the on-site sink	captured by $\mathcal{ADV}$
$\mathcal{D}(r_a, r_b)$	captured by $\mathcal{ADV}$	captured by $\mathcal{ADV}$
$\mathcal{D}(r_b, r_c)$	sent to the on-site sink	captured by $\mathcal{ADV}$

As explained above, a fundamental security task in UWSNs is how to protect the data collected before the reside period and the data collected after the reside period from the $\mathcal{ADV}$. To carry out these tasks, a set of notions, forward secrecy, backward secrecy [2] and data reliability, need to be defined. Let c_i^r denote the ciphertext encrypted by a sensor s_i in round r.

Definition 1.1. (**Forward secrecy**) Forward secrecy is provided if for any $\mathcal{ADV}$, it is computationally infeasible for the $\mathcal{ADV}$ to obtain any information about ciphertext c_i^r $(r \in [r_l, r_a))$, even if a set of secret keys obtained during the reside period $T_{rp} = [r_a, r_b]$ are available.

Definition 1.2. (**Backward secrecy**) Backward secrecy is provided if for any $\mathcal{ADV}$, it is computationally infeasible for the $\mathcal{ADV}$ to obtain any information about ciphertext c_i^r $(r > r_b)$, even if a set of secret keys obtained during the reside period $T_{rp} = [r_a, r_b]$ are available.

Sensor crash attack is a general threat when a UWSN is deployed in hostile environments. Sensor nodes may lose their functionality due to natural disaster (e.g., flood, stone storm), energy depletion, or physical attacks by $\mathcal{ADV}s$.

Definition 1.3. (**Data reliability**) Data reliability is defined as the degree of resilience of data stored in sensor nodes against physical attacks on sensors. Data resilience is evaluated by calculating the fraction of sensor nodes that lose functionality.

1.2.2 Node Capture Resistance

In addition to protecting the accumulated data from $\mathcal{ADV}$ when sensors are compromised, it is important to 1) detect whether a (or which) sensor is captured from the network as early as possible; and 2) adopt security mechanisms to diminish the effects of the compromised sensors.

Indeed, the advance in robotics nowadays enables us to develop a variety of mobile devices [20] to form Mobile UWSNs (MUWSNs). Mobile devices, especially small robots with sensing, wireless communication, and mobility capability, are useful for applications such as adaptive sampling, static sensor deployment and event detection [56, 112].

Consider a scenario where both sink and sensor nodes are mobile, as shown in Fig. 1.3. A Sensor Equipped Aquatic Swarm (SEA Swarm) [105] is proposed

[2] In some literature, e.g., [31], forward secrecy and backward secrecy are defined in an opposite way to ours.

to support time critical applications, such as submarine tracking and harbor monitoring. The SEA Swarm consists of a large number of underwater sensor nodes, air-dropped to the venue of interest. Each node can dynamically control the depths using the bladders and on-board pressure gauges, operate and move as swarm with water current, searching for invasive submarine or scouting the waters around harbor or underwater mining facilities [44]. There are a few unmanned submarines (equipped with sink or base station) that can receive alert messages from sensor nodes. Assume that an invasive submarine intends to go through the surveillance area covered by an MUWSN. To do so, the submarine tries to capture a sensor from the network, re-programme it and re-deploy it in the network to make an unsurveilled hole in the network so that the submarine can go through the surveillance area undiscovered. Through the captured node, a large number of replica nodes can be generated and spread throughout the network based on the software and secret keys obtained from the captured node. These attacks enable replicas to participate in the network pretended as authorized nodes.

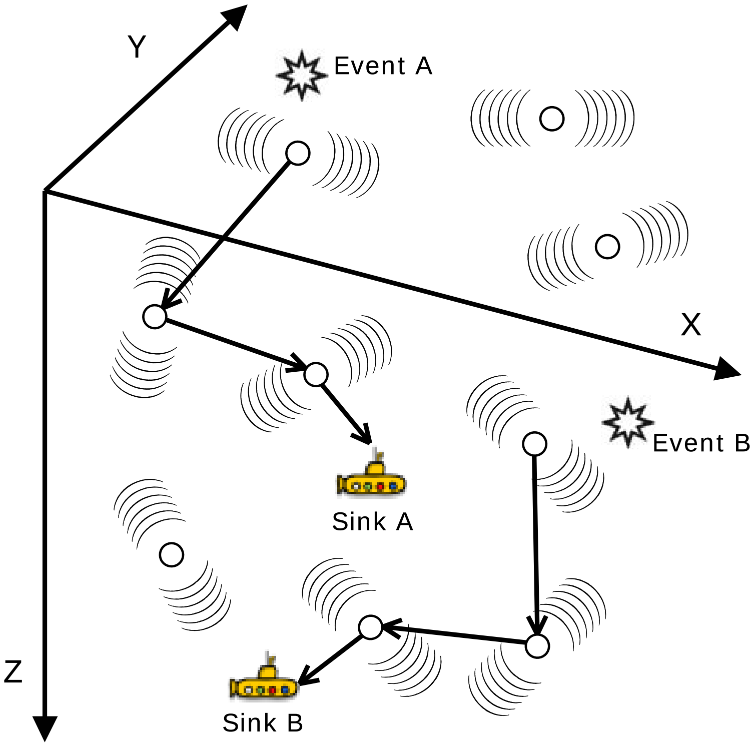


Figure 1.3: An example of a SEA Swarm: mobile nodes detect events and report them to the corresponding $\mathcal{MS}$/sink.

To diminish the effects of the aforementioned node capture attack, three issues have to be addressed. 1) Detection mechanism - how to detect node capture attack. In addition, upon detecting captured nodes, we should revoke the captured nodes from the network and add new nodes to maintain network connectivity. 2) Forward security - how to prevent the newly added nodes from reading any previously transmitted messages of the network. 3) Backward security - how to prevent the captured node from accessing future network communication.

1.3 Main Contributions

The main contribution of this dissertation (made by the author) is a set of novel security mechanisms dedicated for UWSNs. More specifically, this dissertation makes the following contributions:

Distributed Data Storage. We study how to provide both forward secrecy, backward secrecy and data reliability simultaneously under the mobile adversary model. We utilize key-evolution [6] and Reed-Solomon codes [74] to achieve forward secrecy, probabilistic backward secrecy and data reliability. Sensors update their secret key in each round using one way hash function and use the updated secret key to encrypt data sensed in that round. We further use data redundancy to provide data reliability. That is, the ciphertext encrypted in each round is encoded into n data shares based on (n, m) Reed-Solomon codes ($m \leq n$), and then the data shares are distributed to the sensor's neighbor nodes. The original ciphertext can be reconstructed if fewer than $n - m$ data shares are lost.

However, the key evolution and Reed-Solomon codes based scheme cannot provide perfect backward secrecy. We propose then a homomorphic encryption and (n, m) homomorphic secret sharing based scheme and demonstrate that backward secrecy of *historical* data can be provided. In the scheme, the data collected by sensor nodes are encoded into several shares according to the homomorphic secret sharing scheme [7]. Then, the encoded data shares are encrypted according to the homomorphic encryption scheme [14]. Due to the homomorphic property, the encrypted data share generated by a sensor node can be simply combined together in an encrypted form. Statistical data, such as expected value, variance value, can be easily computed based on those encrypted data values. Multiple encrypted data shares generated in a time interval can be stored in a unit data memory, which dramatically reduces the storage and transmission cost as compared with traditional replication-based schemes. In addition, our scheme ensures data reliability and confidentiality by using a secret sharing scheme which provides redundancy by dividing data into multiple shares and recovering data by retrieving a portion of the shares. Furthermore, our scheme that takes advantage of key evolution and homomorphic encryption can provide both forward secrecy and backward secrecy. Through detailed analysis it is demonstrated that our scheme has low storage and transmission cost as well as providing both forward secrecy and backward secrecy compared with other existing schemes and is suitable for resource-constraint UWSNs [93, 90].

Optimized data distribution. As mentioned above, the distributed data storage schemes take the advantage of either (n, m) secret sharing scheme or (n, m) Reed-Solomon codes, in which m of n data shares are required to reconstruct data, to add data redundancy for providing resilience to node invalidation and Byzantine failure. However, how to specify suitable values for m and n is another important question needed to be answered. To specify (n, m) in an optimized way, we propose a constrained optimization data distribution scheme in which suitable values of (n, m) can be selected to maximize security level and at the same time to maximize data reliability. We conducted simulations [88] to show the superiority of our scheme in comparison with several previous approaches developed for UWSNs [94, 89].

Trust management. In the optimized data distribution scheme, sensors' neighbor nodes are selected based on their security levels. That is, sensors prefer to keep their data on the neighbor nodes with higher security levels. We argue that the trustworthiness can be used as a measurement of sensors' security level. We propose a trust management scheme for efficient trust generation as well as scalable and robust trust data storage in UWSNs. A central issue for trust management in UWSNs is how to store trust data without relying on a trusted third party. We first consider two naive schemes as a first-step attempt to show the existing trust storage problems in UWSNs. After analyzing the shortcomings of those two naive schemes, an advanced scheme based on Geographic Hash Table (GHT) [72] is proposed. The advanced scheme provides a hash-table-like interface, enabling nodes *put* and *get* trust data to designated storage nodes based on node's identification (ID). Consequently, nodes do not need to know the storage nodes' IDs. Instead, they calculate a hash function to find the location of the storage node, reducing the storage cost significantly.

Moreover, we propose a set of similarity threshold functions to remove outliers from trust opinions, preventing attackers from generating false trust opinions to pollute trustworthiness. To the best of our knowledge, the design of trust management in UWSNs has not been studied in the literature from this perspective. Furthermore, we give a detailed analysis of the proposed scheme and conduct simulations using MATLAB to show that our scheme is efficient, robust, and scalable in UWSNs [101, 100].

Node capture resistance. We propose a Sensor Capture Resistance and Key Refreshing (SCARKER) scheme for Mobile UWSNs which improves a node capture detection scheme proposed by [19] by providing forward secrecy, backward secrecy as well as collusion resistance. To guarantee both forward security and backward security, upon detecting a captured sensor, the captured sensor will be revoked through updating group key for all sensors except the revoked sensor. We consider an asymmetric architecture of MUWSNs where sensors have extremely constrained computation capability and energy, while $\mathcal{MS}$ has more computational capability and available energy. The asymmetric architecture enables to allocate more computations to $\mathcal{MS}$. By allocating more computational cost to the $\mathcal{MS}$, each sensor only needs to perform one modular operation and one XOR operation for each group key update. To evaluate the performance of the proposed scheme, we implement it on a Sun SPOT [86] based sensor testbed and test it for different lengths of secret keys and various numbers of sensors. Experimental results show that the time for updating a new group key varies from 56 ms to 546 ms and energy consumption changes within 16.5 - 225 mJ, proportional to the length of secret keys and the number of sensors in a group. It demonstrates that our scheme significantly reduces the computational overhead for sensors, and is thus efficient and suitable for MUWSNs [95, 97].

1.4 Dissertation Outline

The rest of this dissertation is organized as follows:

Chapter 2 presents preliminary concepts.

In Chapter 3, we discuss data confidentiality and data reliability in UWSNs.

Chapter 4 introduces *historical* data confidentiality and data reliability in UWSNs.

Chapter 5 discusses the problem of how to optimally distribute data shares to the neighbor nodes of sensors.

In Chapter 6, we address trust management in UWSNs.

Chapter 7 presents a node capture resistance and key refreshing scheme, SCARKER, for MUWSNs.

Chapter 8 concludes this dissertation. We end up with a discussion of open research problems in the area of UWSNs.

Chapter 2

Preliminaries

This chapter describes a few preliminaries which constitute the blocks for the security mechanisms proposed in this dissertation.

2.1 Erasure Code

A (n, k) erasure code encodes a block of data into n fragments, which has $1/k$ the size of the original block, so that any k can be used to reconstruct the original block. An example of such erasure coding schemes is Reed-Solomon codes [74].

Reed Solomon Codes: For $q \geq n$, let $\alpha_1, ..., \alpha_n$ be n distinct elements of F_q. The Reed Solomon codes of message length k, with parameters $\alpha_1, ..., \alpha_n$ is defined as follows. Associate with a message $\mathbf{m} = \langle m_0, ..., m_{k-1}\rangle$ a polynomial $M(x) = \sum_{j=0}^{k-1} m_j x^j$. The encoding of $\mathbf{m}$ is the evaluation of M at n given points, i.e., $E(\mathbf{m}) = \langle M(\alpha_1), ..., M(\alpha_n)\rangle$. By construction, the Reed-Solomon codes have message with length k and block length n. The message can be reconstructed from any k blocks, and the codes can correct up to $t = \lfloor \frac{n-k+1}{2} \rfloor$ errors.

We give a formal definition of an n-party Reed-Solomon codes algorithm with data space $\mathcal{DATA}$ as a pair $\Pi = (Share^{RS}, Recover^{RS})$, where:

- $Share^{RS}$ is a probabilistic algorithm that takes an input $d \in \mathcal{DATA}$ and generates the n-vector $\mathcal{P} \overset{R}{\leftarrow} Share^{RS}(d)$, where $\mathcal{P} = \{p_1, p_2, \cdots, p_n\}$, R means random output, and $p_i \in \{0, 1\}^*$. If $d \notin \mathcal{DATA}$, $Share^{RS}$ returns $\perp$ ("undefined").

- $Recover^{RS}$ is a deterministic algorithm that takes input $\mathcal{P} \in (\{0, 1\}^* \cup$

$\Diamond)^n$, where $\Diamond$ represents a data share that has been missing (or is not available). The $Recover^{RS}$ outputs $Recover^{RS}(\mathcal{P}) \in \mathcal{DATA} \cup \perp$, where $\perp$ is a distinguished value, denoting failed recovery.

2.2 Secret Sharing

Secret sharing scheme is a threshold scheme in that without enough shares, the secret is information-theoretically secure. One of them is Shamir's (n, m) secret sharing scheme [79] based on polynomial interpolation, in which m of n shares are required to reconstruct secret.

The brief procedure is in the following. The secret Θ is in Z_p (p is a prime number and $p > n$). Each shareholder i is in the set P ($|P| = n$). All mathematical operations are in Z_p. To distribute Θ, select a polynomial $a(x)$ with degree $m - 1$ and a constant term Θ. Generate a share s_i for each i in P with $a(x) : s_i = \Theta + \sum_{j=1}^{m-1} a_j i^j$. s_i is also in Z_p. To reconstruct Θ, retrieve m coordinate pairs (i, s_i) of all i in authorized subset B of P ($|B| = m$) and use the pairs in the Lagrange interpolation formula to recover Θ: $\Theta = \sum_{i \in B} b_i s_i$, where $b_i = \prod_{j \in B, j \neq i} \frac{j}{j-i}$.

2.3 Homomorphic Secret Sharing

Informally, homomorphic secret sharing [7] is about combining shares of independent secrets in such as a way that reconstruction from the combined shares results in a combined secret. Let $\oplus$ and $\otimes$ be binary functions on elements of the secret domain S and of the share domain T respectively. We say that a (n, k) threshold scheme has the $(\oplus, \otimes)$ - homomorphism property if for all $I \subseteq \{1, 2, \cdots, n\}$ determines a set of functions F_I: $T^k \to S$, whenever data $D = F_I(D_{i_1}, ..., D_{i_k})$ and $D' = F_I(D'_{i_1}, ..., D'_{i_k})$ then $D \oplus D' = F_I(D_{i_1} \otimes D'_{i_k}, ..., D_{i_k} \otimes D'_{i_k})$. Shamir's secret sharing scheme is $(+, +)$ - homomorphism.

2.4 Homomorphic Encryption

A privacy homomorphism [10] is an encryption function which allows the encrypted data to be operated on without knowledge of the decryption function.

Let $E()$ denote a encryption function. Let M be the message space and C the ciphertext space such that M is a group under operation $\otimes$ and C is a group under operation $\oplus$. $E()$ is a $(\otimes, \oplus)$ - homomorphic encryption function if for $c_1 = E_{k1}(m_1)$ and $c_2 = E_{k_2}(m_2)$, there exists a key k such that $c_1 \otimes c_2 = E_k(m_1 \oplus m_2)$.

The authors in [14] proposed an additive homomorphic encryption scheme such that the addition of the encrypted messages equals the encryption of those messages' summation. Using this scheme, ciphertext $c = \mathrm{Enc}(m, k, M) = m+k \pmod M$, where k is a randomly generated keystream, and $k \in [0, M-1]$, m represent a message, and $m \in [0, M-1]$, M is a large integer. To decrypt the ciphertext, $Dec(c, k, M) = c - k \pmod M$. The addition of n ciphertexts,

$$\begin{aligned}\sum_{i=1}^{n} c_i &= \sum_{i=1}^{n} Enc(m_i, k_i, M) \\ &= \sum_{i=1}^{n} m_i + \sum_{i=1}^{n} k_i \pmod M \ .\end{aligned}$$

Thus,

$$\begin{aligned}Dec(\sum_{i=1}^{n} c_i) &= \sum_{i=1}^{n} Dec(Enc(c_i, k_i, M)) \\ &= \sum_{i=1}^{n} Enc(m_i, k_i, M) - \sum_{i=1}^{n} k_i \pmod M \ .\end{aligned}$$

2.5 Chinese Remainder Theorem (CRT)

If integers $n_1, n_2, \cdots, n_k$ are pairwise relatively primes, and $r_1, \cdots, r_k$ are integers, then the system of simultaneous congruences

$$\begin{cases} \mathcal{X} \equiv r_1 \pmod{n_1} \\ \mathcal{X} \equiv r_2 \pmod{n_2} \\ \quad\vdots \\ \mathcal{X} \equiv r_k \pmod{n_k}. \end{cases}$$

has exactly one solution modulo $N = \prod_{i=1}^{k} n_i$.

The value of $\mathcal{X}$ can be calculated as

$$\mathcal{X} = (\sum_{i=1}^{k} c_i \cdot r_i) \pmod N, \tag{2.1}$$

where $c_i = Q_i U_i$, $Q_i = \frac{N}{n_i}$ and U_i is its multiplicative inverse modulo n_i, i.e. $U_i = Q_i^{-1} \pmod{n_i}$.

One of the important features of the CRT is that it provides a way to hide tuples of small numbers within a larger number. Since $n_1, \cdots, n_k$ are randomly picked from an unlimited large pool of pairwise relatively prime positive integers, thus knowing one (or small part) of them gains little knowledge about the others. That is, $r_1, \cdots, r_k$ cannot be derived from $\mathcal{X}$ without knowing relatively pairwise primes $n_1, \cdots, n_k$, whereas $\mathcal{X}$ cannot be derived from $r_1, \cdots, r_k$ without knowing $n_1, \cdots, n_k$.

Chapter 3

Data Confidentiality and Data Reliability

In this chapter, we address secure and reliable distributed data storage in UWSNs by designing a secure and reliable data distributed storage scheme based on key evolution and (n, m) Reed-Solomon codes. The proposed scheme provides forward secrecy, probabilistic backward secrecy and reliability of data in absence of reliable nodes and communication channels.

3.1 Related Work

3.1.1 Data Secure Storage and Reliability

The authors in [23] identify, for the first time, the problem of data survival in UWSNs in the presence of a capable mobile adversary. Their idea is to move the sensed data to sensors' neighbors, which is referred to as a Data-Moving scheme. In the Data-Moving scheme, when a sensor node senses new data, it randomly selects one node in its neighbors and moves data to it. Data-Moving has two basic strategies: one is MOVE-ONCE, in which the sensed data are moved to one neighbor and then remains there; the other is KEEP-MOVING, in which the sensed data are moved to one neighbor and keep on moving to such neighbor's neighbor until offloading to a mobile sink. However, the KEEP-MOVING scheme has very large transmission cost caused by data moving. To protect high-value data, they use replication as one means to increase data survival probability. However, the tradeoff is between survival probability and storage overhead.

With the development of hardware technology, several new-generation sensors [64, 62] equipped several gigabytes and low-power energy consumption flash memory becomes available. The authors in [30] proposed a re-thinking WSN architecture where the sensed data are stored locally as "historical" data instead of sending them to the sink immediately. In doing so, the consuming energy for data transmission can be saved to maximize the sensor's life. However, gigabytes flash memory is not the best way to solve storage problem since sensed data grow over the sensor lifetime. Storage efficiency should be considered. Furthermore, data reliability was not addressed in [30].

The authors in [84] proposed a family of schemes for secure distributed data storage and retrieval, namely, the Simple Hash-Based (SHB) scheme, the Enhanced Hash-Based (EHB) scheme, and the Adaptive Polynomial-Based (APB) scheme. The SHB scheme requires a network controller for data decryption. Thus, the controller could become a transmission bottleneck. In the EHB scheme, the lifetime of sensor network is divided into n phases where the number of phases is fixed, and the data encryption keys should be initialized and preloaded before sensor nodes are deployed. These limitations hamper the scalability of the network. The APB scheme provides the scalability of the network. However, data reliability and data storage efficiency were not considered in any these three schemes.

The authors in [17] proposed a redundant residue number system to provide data confidentiality and reliability. data are encoded as $(h + r)$-tuples of residues using $h + r$ keys. Residues are distributed among the mobiles in the network. Recovering the original data requires at least h residues and the corresponding moduli. Data can be recovered even when up to $s \leq r$ residues are lost, and data confidentiality is ensured since recovering the original data requires knowledge of the entire set of moduli. Unfortunately, data storage and transmission efficiency issues were not addressed.

3.1.2 Forward Secrecy and Backward Secrecy

In recent years, many schemes have been proposed to achieve either forward secrecy [5, 42] only or both forward secrecy and backward secrecy [43, 33, 31, 32]. Generally, key evolution is a common approach in all of these schemes to provide forward secrecy. Its basic idea is that secret key $\mathcal{K}_i$ is updated by applying a secure hash function at each round r, e.g., $\mathcal{K}_i^r = h(\mathcal{K}_i^{r-1})$ $(r \geq 1 \,\text{and}\, \mathcal{K}_i^0 = \mathcal{K}_i)$. Because of the one-way property of the secure hash function, deriving the previous rounds' keys (pre-compromised) based on the current round key is infeasible.

However, the aforementioned public cryptography based approaches [43, 33, 31, 32] are not suitable for UWSNs because sensor nodes only use the public key of the sink to encrypt data in these approaches. Whereas, sensors are usually considered as resource constrained devices that cannot afford use of public key cryptography.

In the context of WSNs, [71] proposed a secure information aggregation protocol relaying on key evolution to provide forward secrecy but without guarantee of backward secrecy. The authors in [65] proposed a protocol to maintain secrecy between a pair of network neighbors. In [65], both forward secrecy and backward secrecy for keys shared pairwise by two sensors are provided. The security is based on assumption that two sensors cannot be compromised at the same time.

A more recent study [60] proposed a DIstributed Self-Healing (DISH) approach. The basic idea of DISH is that unattended sensors attempt to recover from *sick* states and maintain forward secrecy and backward secrecy of the collected data. In DISH, forward secrecy is provided through key evolution. However, DISH does not guarantee backward secrecy. Instead, it provides probabilistic backward secrecy which depends on conditions such as compromising capability of the mobile adversary (number of nodes it can compromise at a given time interval), and for how often the $\mathcal{MS}$ successively visits the network. The authors in [21] proposed Proactive co-Operative Self-Healing (POSH) scheme which has improved DISH [60]. The basic mechanism of self-healing in POSH is the same, which makes *sick* sensors receive random contributions from healthy peers to become healthy again. The difference is that the DISH scheme is referred to as a *pull* model because sensors request contributions from peers; whereas, the POSH scheme is referred to as a *push* model in which sponsors volunteering their contributions. In other words, in the *push* model, sensors send their contributions to randomly selected neighbors without receiving any requests.

Both DISH [60] and POSH [21] are proposed to provide forward secrecy and merely certain probabilistic backward secrecy in *ideal* networks where sensors and communication channels are reliable. However, they are not resilient to node failure and Byzantine failure. Aiming at solving this problem, the authors in [109] take advantage of (n, k) secret sharing and (n, m) Reed-Solomon codes, in which m (or k) of n data shares are required to reconstruct data, adding data redundancy to provide resilience to node invalidation and Byzantine failure. However, neither forward secrecy, backward secrecy nor how to choose m and n are addressed in their work.

In summary, none of the above mentioned schemes provide the overall requirements of forward secrecy, backward secrecy and data reliability needed for UWSNs.

3.2 Models and Assumptions

In this section, we present our assumption about the network model and the threat model, and define the design goals of the scheme.

3.2.1 Network Model

We consider a UWSN that consists of N sensor nodes, denoted as $s_i \in \mathbb{S}$, where $\mathbb{S} = \{s_i\}_{i=1}^{N}$. A node, s_i, has nb_i neighbors, which compose a neighbor node set, NB_i, for s_i. There is an $\mathcal{MS}$ that visits the UWSN periodically to collect data. The time interval between the current visit and the previous visit is denoted as T. Sensor s_i collects data at each round, and the data generated at round r is denoted as d_i^r. Let R denote the number of rounds between successive retrieving the sensed data. Once d_i^r is generated, it is stored locally, and waits until an authorized $\mathcal{MS}$ offloads it. Each sensor has the ability to perform *one-way* hashing and symmetric key encryption. Furthermore, $\mathcal{MS}$ is assumed to be a trusted party which cannot be compromised. Additionally, $\mathcal{MS}$ will re-instantiate the secret keys and reset the round counters when it visits the network.

A list of notations is summarized in Table 3.1.

3.2.2 Threat Model

The UWSNs can be attacked in many ways [54, 120, 36]. In this study, we focus on a mobile adversary [60] (denoted as $\mathcal{ADV}$ hereafter) that performs roaming in the UWSN while $\mathcal{MS}$ is absent. The $\mathcal{ADV}$ has capabilities as follows:

- Compromising ability: The $\mathcal{ADV}$ can compromise up to $k < N$ sensors during a time interval T. Once a sensor s_i is compromised by the $\mathcal{ADV}$,

Table 3.1: Notations.

$\mathcal{MS}$	mobile sink
$\mathcal{ADV}$	mobile adversary
N	number of sensor nodes in a UWSN
s_i	sensor i
NB_i	set of neighbor sensor nodes of s_i
nb_i	the number of nodes in NB_i
$\mathcal{K}_i^r$	secret key of sensor s_i in round r
T	time interval of two visiting
r	round index
R	number of rounds between successive sink visits
d_i^r	sensed data collected by sensor i at round r

all the storage content will be revealed, including secret keys, encrypted data, hash function, etc.; and all incoming and outgoing communication will be monitored.

- Topology knowledge: The $\mathcal{ADV}$ has the knowledge about the network topology. It can choose any node to compromise.

- No interference: The $\mathcal{ADV}$ would not interfere with the communications between nodes, nor rework any data sensed by, or stored by sensors it compromises. In other words, the $\mathcal{ADV}$ is *read-only* (or passive) attacker.

- Strictly local eavesdropping: The $\mathcal{ADV}$ is unable to monitor and record the communications of other nodes. It can only eavesdrop incoming and outgoing communications from and to the currently compromised nodes.

In addition to the attacks mentioned above, $\mathcal{ADV}$ has attack strategies as follows:

- *Read-only* $\mathcal{ADV}$*:* Its goal is to learn as much sensed data as possible.

- $\mathcal{ADV}\_\mathit{del}$*:* $\mathcal{ADV}\_\mathit{del}$ tries to prevent certain target data from reaching the sink. For example, in a nuclear emission monitoring application, sink will raise the alarm if one of the sensing nodes reports a value above a pre-specified threshold. $\mathcal{ADV}\_\mathit{del}$ thus aims to find that value and erase it before it ever reaches the sink. $\mathcal{ADV}\_\mathit{del}$ might be undetected if the sink tolerates some missing measurements (due to occasional errors or malfunctions).

- $\mathcal{ADV}\_md$: If the sink has no tolerance for lost data, $\mathcal{ADV}$ changes its strategy from $\mathcal{ADV}\_del$ to $\mathcal{ADV}\_md$, in the sense that $\mathcal{ADV}$ replaces the target data with a value within the threshold.

3.2.3 Design Goals

The design goals are to guarantee forward secrecy, backward secrecy and data reliability as defined in Section 1.1.

3.3 The Proposed Schemes

In this section, we investigate a family of schemes for secure and reliable data storage in UWSNs, named as Naive scheme, Advanced scheme and Enhanced scheme respectively, Each latter scheme improves over the previous one by addressing some of its limitations.

3.3.1 Naive Scheme (NS)

In the NS, a sensor node, s_i, generates a data d_i^r in round r, and stores it locally. To protect data, it can perform the following operations.

Step 1: Initially, $\mathcal{MS}$ picks a secret key denoted as $\mathcal{K}_m$. The initial key of sensor node s_i can be computed as $\mathcal{K}_i = h(\mathcal{K}_m||i)$ which is set up before the deployment of the UWSN. Here, $||$ stands for the concatenation operator. $\mathcal{MS}$ only needs to store one key, because it can use $\mathcal{K}_m$ to compute all the secret keys deployed in $s_i, 1 \leq i \leq N$.

Step 2: s_i generates a keyed hash value of data d_i^r generated in round r, using key $\mathcal{K}_i$. The keyed hash value, denoted as $MAC_i^r = h(d_i^r||\mathcal{K}_i)$ protects data integrity.

Step 3: The plaintext data, denoted as $d_i^r||MAC_i^r$, is encrypted by using key $\mathcal{K}_i$. The encryption of data, denoted as $Enc(\mathcal{K}_i||d_i^r||MAC_i^r)$, protects data confidentiality.

Discussion. Once sensor s_i is compromised at round r_a, the $\mathcal{ADV}$ holds the secret key $\mathcal{K}_i$ of this sensor. Consequently, all the data generated before the sensor is compromised is revealed, because all the data encrypted before round r_a are encrypted by the same key $\mathcal{K}_i$ which is now held by the $\mathcal{ADV}$. Thus, the forward secrecy is not provided. Moreover, once the $\mathcal{ADV}$ returns at rounds $r_c\,(r_b < r_a)$, it still can decrypt the data generated between round r_b and r_c, even if the $\mathcal{ADV}$ was not present in the time interval. Therefore, the backward secrecy is not provided either.

To solve the aforementioned problems, we further propose two schemes - Advanced scheme and Enhanced scheme.

3.3.2 Advanced Scheme (AS)

Since the sensor keeps the secret key unchanged in NS, all encrypted data can be read by $\mathcal{ADV}$, no matter the data are generated before or after the compromised period. In other words, the $\mathcal{ADV}$ only needs to compromise the sensors once and gets the secret key consequently. Therefore, we need to change the secret key after each round. The AS is designed as follows:

Since hash function has *one-way* property, s_i updates its secret key at each round, that is $\mathcal{K}_i^r = h(\mathcal{K}_i^{r-1})$, $r \geq 1$. This means that the $\mathcal{ADV}$ which holds the secret key $\mathcal{K}_i^r$ in compromised time interval $[r_a, r_b]$ $(r \in [r_a, r_b])$ cannot derive any previous key $\mathcal{K}_i^{\hat{r}}$, $(\hat{r} < r_a)$, where $\mathcal{K}_i^r$ is the secret key of sensor s_i at round r, and $h(\cdot)$ is a *one-way* hash function. Similar to NS, after s_i generates new data d_i^r at round r, it firstly encrypts the d_i^r with $\mathcal{K}_i^r$ by $Enc(\mathcal{K}_i^r||d_i^r||MAC_i^r)$, and stores it locally. It then updates its secret key as $\mathcal{K}_i^{r+1} = h(\mathcal{K}_i^r)$, and finally the $\mathcal{K}_i^r$ is securely erased. $\mathcal{MS}$ can easily compute the corresponding secret key of s_i at round r for $\mathcal{K}_i^r = h^{r-1}(\mathcal{K}_i) = h^{r-1}(\mathcal{K}_m||i)$.

Discussion. Since the secret key $\mathcal{K}_i^r$ updates itself at each round, the $\mathcal{ADV}$ cannot derive the key from previous rounds (before the sensor was compromised) due to the *one-way* property of $h(\cdot)$. Thus, forward secrecy is provided. However, the $\mathcal{ADV}$ which holds the secret key $\mathcal{K}_i^r$, $r \in [r_a, r_b]$ still can derive the future key which will be used in the following rounds. In other words, if the $\mathcal{ADV}$ returns at round r_c $(r_b < r_c)$, it still can decrypt the data which was encrypted in time interval $[r_b, r_c]$, by mimicking key updates. Therefore, the backward secrecy is not guaranteed. Furthermore, since all the generated data are stored locally in the sensor nodes, data would be lost if sensor nodes lose functionality due to physical attacks launched by $\mathcal{ADV}$ as defined in Subsection 3.2.2. Consequently, data reliability is not provided. To guarantee both forward secrecy and backward secrecy as well as data reliability, we propose the Enhanced scheme.

3.3.3 Enhanced Scheme (ES)

We observe that data encrypted by symmetric encryption cannot guarantee backward secrecy. It holds as long as a sensor relies only on itself for security. As will be discussed later, backward secrecy can be *probabilistically* achieved if sensors cooperate with their neighbors. Moreover, we believe that data reliability is also provided if sensors cooperate with their neighbors since we could add data redundancy to provide data reliability. To do so, a straightforward way is to send data replication to their neighbor nodes. Consequently sensors are resilient to physical attacks since data replication is still available in the network even when some fraction of sensor nodes totally lose functionality. However, the effect is that data replication would cause large storage overhead since the size of data replication is the same as the original data, which is not affordable in storage constrained sensor nodes. Therefore, the new scheme should provide data reliability with low storage overhead.

To reduce storage requirement, the (n, m) Reed-Solomon codes can divide a block of data into n shares, each has $\frac{1}{m}$ the size of the original block. The original data can be reconstructed if m or more data shares are available. We take the advantage of (n, m) Reed-Solomon codes to provide physical attack resilience with low storage cost.

The ES that satisfies the mentioned above requirements contains the following steps:

Step 1: System initialization.

$\mathcal{MS}$ picks a secure hash function, denoted as $h(\cdot)$, and a secret key denoted as $\mathcal{K}_m$. Before deploying each sensor node, $\mathcal{MS}$ preloads to the sensor hash function $h(\cdot)$, and initial data encryption key $\mathcal{K}_i$ for each sensor, s_i. Here, $\mathcal{K}_i$ is computed as $h(\mathcal{K}_m||i)$. In the end of each round, the round index r and the encryption key $\mathcal{K}_i^r$ are updated as $\mathcal{K}_i^r = h(\mathcal{K}_i^{r-1})$, where $r = 1, 2, ...$ and $\mathcal{K}_i^0 = \mathcal{K}_i$. Again, $\mathcal{MS}$ only needs to store a single secret key $\mathcal{K}_m$ and all round keys $\mathcal{K}_i^r$ can be derived when needed.

Step 2: Distributed data storage.

Each sensor s_i firstly generates a keyed hash value with round key $\mathcal{K}_i^r$ as $MAC_i^r = h(d_i^r||\mathcal{K}_i^r)$, and then a plaintext data that consists of d_i^r, MAC_i^r, and values r and s_i, denoted as $PLtext_i^r = d_i^r||MAC_i^r||r||s_i$, is encrypted by using updated key $\mathcal{K}_i^r$. The encryption data are denoted as

$$\begin{aligned} ENtext_i^r &= Enc(\mathcal{K}_i^r, PLtext_i^r) \\ &= Enc(\mathcal{K}_i^r || d_i^r || MAC_i^r || r || s_i). \end{aligned}$$

Thus, the integrity and forward secrecy of the sensed data are provided. d_i^r is included in

$$m_i^r \;=\; ENtext_i^r || r || s_i.$$

Step 3: Data shares *generation.*

Sensor s_i employs (n, m) Reed-Solomon codes to split $ENtext_i^r$ into n data shares, denoted as a set $\mathcal{P}_i = \{p_{i,1}^r, p_{i,2}^r, \cdots, p_{i,n}^r\}$.

Step 4: Data distribution.

Sensor s_i selects top n security level neighbors in set NB_i (e.g., s_j) based on node selection scheme (more details in Chapter 5), and sends one randomly selected distinct data share $m_{i,j}^r = p_{i,j}^r || r || s_i$ to s_j by using pairwise secret key $\mathcal{K}_{i,j}$ to encrypt the packet.

$$s_i \rightarrow s_j : Enc(\mathcal{K}_{i,j}, m_{i,j}^r).$$

After the data are distributed, the original data are securely erased.

Step 5: Data reconstruction.

$\mathcal{MS}$ collects m data shares from nodes and reconstructs data using (n, m) Reed-Solomon codes.

3.4 Analysis

We analyze in this section the efficiency of ES in terms of efficiency and security.

3.4.1 Efficiency Analysis

3.4.1.1 Computation Cost

At each round, the data source node s_i needs to perform two hash operations, to update key $\mathcal{K}_i^r = h^{r-1}(\mathcal{K}_m || i)$ and then to compute $MAC_i^r = h(d_i^r || \mathcal{K}_i^r)$, and two symmetric encryptions $ENtext_i^r = Enc(\mathcal{K}_i^r, PLtext_i^r)$ and $Enc(\mathcal{K}_{i,j}, m_{i,j}^r)$. To generate distributed data shares, s_i encodes $ENtext_i^r$ into

n data shares using Reed-Solomon codes. Let γ and β denote the size of $\mathcal{K}_m||i$ and $d_i^r||\mathcal{K}_i^r$, respectively. The total computation cost at the data source node s_i is $Hash_\gamma^1 + Hash_\beta^1 + SymEnc^2 + RSCoding^1$, where $Hash_\gamma^1$ denotes one hash operation with input size of γ, $SymEnc^2$ denotes two symmetric encryptions, and $RSCoding^1$ denotes one Reed-Solomon codes operation. The computation cost at data share holders is only one symmetric decryption operation.

3.4.1.2 Storage Overhead and Communication Overhead

We regard symbols s_i, $\mathcal{K}_i^r$, and r as the elements of the Galois Field $GF(2^q)$ (e.g., $q = 8, 16$). After s_i generates data shares, s_i would send $Enc(\mathcal{K}_{i,j}, \{p_{i,j}^r, \{r, s_i\}\})$ to one of its neighbors, s_j, where $p_{i,j}^r$ is the output of coding $ENtext_i^r$ using Reed-Solomon codes. Assume $ENtext_i^r$ contains ϕ symbols. Due to the property of Reed-Solomon codes, the size of $p_{i,j}^r$ is $\frac{\phi}{m}$. In addition, each node needs $nb_i * q$ bits storage overhead to maintain a probability vector. Thus, the communication overhead during the distribution is approximately $n * (\frac{\phi}{m} + 2) * q$ bits, and it requires $(\frac{\phi}{m} + nb_i + 2) * q$ bits storage overhead to keep the data shares at each data holder.

3.4.2 Security Analysis

In this subsection, we analyze the security aspects of ES with respect to Read-only $\mathcal{ADV}$, $\mathcal{ADV}\_del$ and $\mathcal{ADV}\_md$.

3.4.2.1 *Read-only* $\mathcal{ADV}$

Proposition 3.1. *The Enhanced scheme provides forward secrecy.*

Proof. Since the secret key $\mathcal{K}_i^r$ of a sensor s_i is updated as $\mathcal{K}_i^r = h(\mathcal{K}_i^{r-1})$, where $r = 1, 2, ...$ and $\mathcal{K}_i^0 = \mathcal{K}_i$, $\mathcal{ADV}$ cannot derive the previous key from the current key due to the *one-way* property of hash function $h(\cdot)$. Hence the $\mathcal{ADV}$ cannot decrypt the data encrypted and stored in the previous rounds. Therefore, the forward secrecy is provided. □

Lemma 3.2. *The backward secrecy of the sensor s_i can be compromised by $\mathcal{ADV}$, if and only if the following three conditions are satisfied.*

1. sensor s_i is compromised by the $\mathcal{ADV}$;

2. the $\mathcal{ADV}$'s compromising ability $k > m$;

3. the $\mathcal{ADV}$ compromised at least m neighbor nodes of s_i that store the corresponding data shares.

Proof. Recall that, according to Definition 1.2, the backward secrecy of s_i is the security of data generated and encrypted after the *reside period* T_{rp}. As shown in Fig. 1.2, sensor s_i is compromised by an $\mathcal{ADV}$ at round r_a. After that, the $\mathcal{ADV}$ resides in s_i and releases s_i at round r_b, the time interval between round r_a and r_b is defined as *reside period* T_{rp}. In other words, if the backward secrecy of the s_i is compromised, meaning that the data generated and encrypted after round r_b is compromised by the $\mathcal{ADV}$ which is based on the secrecy obtained during T_{rp}. Thus, if the $\mathcal{ADV}$ wants to compromise the backward secrecy of s_i, it has to firstly compromise s_i to get the secret key $\mathcal{K}_i^{r_a}$ and hash function that used to generate future keys. Secondly, the $\mathcal{ADV}$ must have enough compromising ability ($k > m$). Thirdly, the $\mathcal{ADV}$ has to compromise at least m neighbor nodes of s_i that stores the corresponding data shares to recover encrypted data $ENtext_i^{\hat{r}}$ ($\hat{r} > r_b$). Finally, the $\mathcal{ADV}$ should have ability to decrypt the $ENtext_i^{\hat{r}}$ to obtain $PLtext_i^{\hat{r}}$ by using secret key $\mathcal{K}_i^{r_a}$ obtained in T_{rp}. Therefore, the backward secrecy of s_i can be compromised by an $\mathcal{ADV}$, if and only if all *Conditions* 1, 2, and 3 are satisfied. □

Theorem 3.3. *The ES provides backward secrecy with probability*

$$Pr_{ES} = 1 - Pr(C1) \cdot Pr(C2) \cdot Pr(C3),$$

where $C1$*,* $C2$*, and* $C3$ *refer to Conditions* 1*,* 2*, and* 3*, respectively, and* $Pr(C1)$ *is the probability that Condition 1 is satisfied, so do* $Pr(C2)$ *and* $Pr(C3)$*.*

Proof. Sketch of proof. □

3.4.2.2 $\mathcal{ADV}$_*del*

Proposition 3.4. $\mathcal{MS}$ *can recover the original data if the number of failure message in* n *is fewer than threshold value* $n - m$*. The probability of successful data recovery as a function of random node failure is*

$$Pr_{rst\_mf} = 1 - \sum_{t=n-m+1}^{n} \begin{pmatrix} n \\ t \end{pmatrix} Pr_{mf}^{t}(1 - Pr_{mf})^{n-t}$$

where Pr_{mf} is the probability that a message has a random failure.

Proof. Let $Pr(A)$, $Pr(B_t)$ represent the probability that $\mathcal{MS}$ cannot recover the original data due to random node failure, and the probability that t nodes are failed, respectively. It is easy to see that

$$Pr(B_t) = \begin{pmatrix} n \\ t \end{pmatrix} Pr_{mf}^t (1 - Pr_{mf})^{n-t} \ .$$

Recall that the basic idea of (n, m) Reed-Solomon codes is that the original data can be reconstructed if at least m data shares are available. In other words, the original data cannot be reconstructed if t ($t \in [n - m + 1, n]$) data shares are not available. That is, $Pr(A|B_t) = 1$ when $t \in [n - m + 1, n]$ and $Pr(A|B_t) = 0$ when $t \in [0, n - m]$. We have

$$\begin{aligned} Pr(A) &= \sum_{t=0}^{n} P(B_t)P(A|B_t) \\ &= \sum_{t=0}^{n-m} P(B_t)P(A|B_t) + \sum_{t=n-m+1}^{n} P(B_t)P(A|B_t) \\ &= \sum_{t=n-m+1}^{n} P(B_t)P(A|B_t) \\ &= \sum_{t=n-m+1}^{n} \begin{pmatrix} n \\ t \end{pmatrix} Pr_{mf}^t (1 - Pr_{mf})^{n-t} \ . \end{aligned}$$

Therefore we have the probability of successful data recovery as a function of random node failure

$$\begin{aligned} Pr_{rst\_mf} &= Pr(\overline{A}) = 1 - Pr(A) \\ &= 1 - \sum_{t=n-m+1}^{n} \begin{pmatrix} n \\ t \end{pmatrix} Pr_{mf}^t (1 - Pr_{mf})^{n-t}. \end{aligned}$$

□

Fig. 3.1 shows the analysis results for given (n, m) parameters.

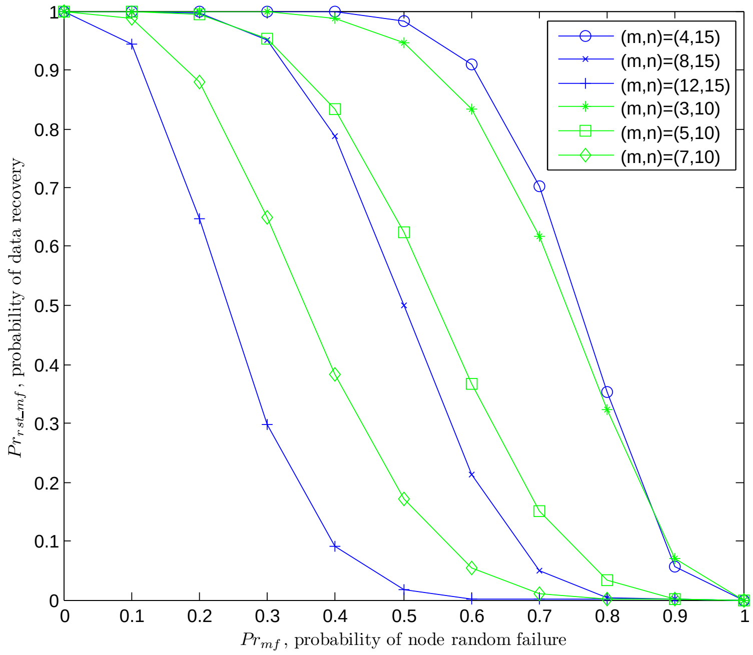


Figure 3.1: Probability of data recovery due to node random failure.

3.4.2.3 $\mathcal{ADV}\_md$

Proposition 3.5. *Given Pct_m, the percentage of the data shares modified by $\mathcal{ADV}\_md$, the original data can still be reconstructed upon the first retrieval with a probability*

$$Pr^1_{rcs} = \frac{(n(1-Pct_m))!(n-m)!}{n!(n(1-Pct_m)-m)!}$$

where $n(1-Pct_m) \geq m$.

Proof. The sample space is the total combination of data retrievals, i.e., $\begin{pmatrix} n \\ m \end{pmatrix}$. The event that the original data are successful reconstructed is

$$Pr^1_{rcs} = \begin{pmatrix} n(1-Pct_m) \\ m \end{pmatrix}.$$

Thus the probability of successful data reconstruction upon the first retrieval is

$$Pr_{rcs}^{1}=\frac{\begin{pmatrix} n(1-Pct_m) \\ m \end{pmatrix}}{\begin{pmatrix} n \\ m \end{pmatrix}}=\frac{(n(1-Pct_m))!(n-m)!}{n!(n(1-Pct_m)-m)!} \ .$$

□

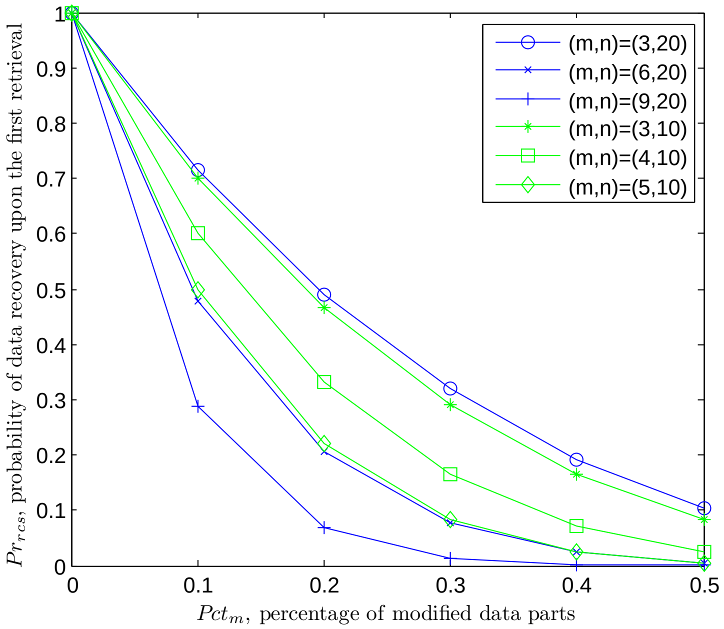


Figure 3.2: Probability of data recovery upon the first retrieval.

Fig. 3.2 depicts the analysis results for given (n, m) parameters.

Finally, Table 3.2 summarizes the comparison results with a few existing schemes in terms of forward secrecy, backward secrecy, resilient to Node Failure (RNF) and Resilient to Message Failure (RMF). The results show that ES has the best performance among these studied schemes.

3.5 Summary

In this chapter, we propose secure and reliable distributed data storage in UWSNs by taking advantages of key evolution and Reed-Solomon codes, which

Table 3.2: Security and performance comparison result among existed work and proposed schemes in terms of forward secrecy, backward secrecy, *RNF*, and *RMF*.

	forward secrecy	backward secrecy	***RNF***	***RMF***
DISH [60]	Yes	Probabilistic	No	No
POSH [21]	Yes	Probabilistic	No	No
Wang *et al.* [109]	Partial	Probabilistic	Yes	Yes
ES	Yes	Probabilistic	Yes	Yes

has computational security and yet maintains the optimal data size. The security analysis shows that ES provides forward secrecy, probabilistic backward secrecy as well as resiliency to message failure and node failure against Read-only $\mathcal{ADV}$, $\mathcal{ADV}\_del$, and $\mathcal{ADV}\_md$ attacks.

Chapter 4

Historical Data Confidentiality and Reliability

In the previous chapter we proposed and demonstrated that forward secrecy and data reliability can be provided by using key evolution and data distribution. However, by these means we could achieve only probabilistic backward secrecy. In this chapter, we propose a homomorphic encryption and homomorphic secret sharing based scheme to provide backward secrecy of *historical* data. The historical data includes historical aggregation or locally created results after data processing, e.g., the average temperature during last three months; the highest and lowest humidity degree during last 24 hours; or more specifically, the average concentration of a chemical element in soil during last half a year [116].

4.1 Network Model, Threat Model, Design Goals and Evaluation Metrics

In this chapter we continue using the network model and threat model defined in Subsection 3.2.1 and in Subsection 3.2.2, respectively.

UWSNs paradigm is motivated by scenarios whereby not real-time information, but digest information is of interest. Our goal thus is to design a secure and efficient as well as low storage cost data storage scheme for UWSNs. Following evaluated parameters will be used in the subsequent section.

Recall that d_i^r denotes data value collected by sensor s_i at round r. Then $\{d_i^r\}_{r=1}^{R}$ denotes a sequence of data sensed during R rounds. The average value can be calculated as

$$E(d_i) = \frac{\sum_{r=1}^{R} d_i^r}{R}. \tag{4.1}$$

The variance of data can be computed as

$$Var_i = \frac{\sum_{r=1}^{R} (d_i^r - E(d_i))^2}{R}. \tag{4.2}$$

Data Survival Probability (DSP) is the probability of the valid data survival against the $\mathcal{ADV}\_del$ and $\mathcal{ADV}\_md$.

Our design goal is to maximize DSP with respect to $\mathcal{ADV}\_del$ and $\mathcal{ADV}\_md$.

Storage Overhead (SO) is the number of units data memory per node used in a UWSN given that the network size is fixed.

The design goal is to minimize SO of the proposed scheme. Since $\mathcal{ADV}\_del$ and $\mathcal{ADV}\_md$ do not effect SO, we neglect their effect. In our scheme, we consider the storage overhead of the whole sensor network rather than a single sensor node. For example, if a d_i^r generated by a sensor node s_i at round r is stored in two sensor nodes s_i and s_j, the SO of d_i^r is $2 * \lceil log_2(d_i^r) \rceil$, where d_i^r is sensed data collected by sensor s_i at round r.

Transmission Cost (TC) is a value defined as how many units of data transmitted when sensor nodes distribute sensed data over R rounds and how many units of data transmitted when $\mathcal{MS}$ retrieves R rounds data.

4.2 The Proposed schemes

In this section, we present a family of schemes for efficient and reliable distributed data storage. The goal is to increase DSP, decrease SO and TC while maintaining data confidentiality. We first introduce two naive schemes, analyze and discuss their weaknesses. Then we present two enhanced schemes to improve them.

4.2.1 Naive Scheme I: DATA-MOVING

Firstly, we consider Do-Nothing scheme as a baseline, in which no specific schemes are employed to define attacks. As we mentioned above, $\mathcal{ADV}\_del$ and $\mathcal{ADV}\_md$ would compromise data integrity. Assuming that $\mathcal{ADV}\_del$ and $\mathcal{ADV}\_md$ are independent events, after these attacks, the sensor nodes is out of operation. The DSP is:

$$DSP = 1 - Pr^{del} - Pr^{md},$$

where Pr^{del} is the probability of $\mathcal{ADV}\_del$ success, Pr^{md} is the probability of $\mathcal{ADV}\_md$ success, and $Pr^{del} + Pr^{md} \leq 1$.

In R rounds, the SO caused by s_i is: $SO = \sum_{r=1}^{R} \lceil log_2(d_i^r) \rceil$. Since $log_2(d_i^r)$ is stored locally at s_i, thus TC of distributing data are equal to 0. TC of retrieving data are equal to $\sum_{r=1}^{R} \lceil log_2(d_i^r) \rceil$.

The authors in [23] proposed a Data-Moving scheme which can protect certain specified data from $\mathcal{ADV}\_del$. More specifically, the Data-Moving scheme has two strategies: MOVING-ONCE and KEEP-MOVING. As their names suggest, KEEP-MOVING enables data moving between sensors in each round, whereas MOVING-ONCE makes data moving from the sensor it generated to another sensor, and staying there until the $\mathcal{MS}$'s retrieval. Keeping data moving makes $\mathcal{ADV}\_del$ difficult to find and locate the sensor in which the specified data stored. As the data move once in each round, however, KEEP-MOVING scheme causes very large TC, where

$$TC = \sum_{j=1}^{R} \sum_{r=1}^{j} \lceil log_2(d_i^r) \rceil + \sum_{r=1}^{R} \lceil log_2(d_i^r) \rceil .$$

Computation of expected value and variance. To compute expected value and variance, the $\mathcal{MS}$ needs to retrieve plaintext data $d_i^1, \cdots, d_i^r, \cdots, d_i^R$ from s_i. Then the expected value and variance can be computed easily according to Eq. (4.1) and Eq. (4.2).

Proposition 4.1. *The MOVING-ONCE and KEEP-MOVING scheme have the same DSP and SO as the Do-Nothing scheme in presence of $\mathcal{ADV}\_del$ and $\mathcal{ADV}\_md$.*

Proof. (sketch) Since each new sensed data are KEEP-MOVING in the UWSN, each sensor node should keep several data at a certain time. If a sensor node is compromised, the data moved to this node is also compromised. Thus, for the whole network, the volume of data compromised by $\mathcal{ADV}\_del$ is the same as in the scheme MOVING-ONCE and KEEP-MOVING. Therefore, the DSP is the same. The new sensed data is just moved to neighbors in the MOVING-ONCE and KEEP-MOVING scheme. Thus, the SO of MOVING-ONCE and that of KEEP-MOVING scheme are the same as Do-Nothing scheme. □

Proposition 4.2. *DATA-MOVING scheme provides neither forward secrecy nor backward secrecy.*

Proof. The basic idea of DATA-MOVING scheme is to protect some specified data from $\mathcal{ADV}$. Even data are not encrypted in the DATA-MOVING scheme, keeping the data moving in each round makes it for $\mathcal{ADV}$ difficult to find them. However our goal is to provide forward secrecy and backward secrecy of all the data. Once sensors are compromised in DATA-MOVING scheme, all the data stored in them are revealed. Thus the DATA-MOVING scheme provides neither forward secrecy nor backward secrecy. □

4.2.2 Naive Scheme II: Data Redundancy Based Scheme

As discussed in the previous subsection, DATA-MOVING scheme does not improve DSP and SO, and even increases TC. To improve DSP, we can use data redundancy policy, in which the data are stored locally and their copy is sent to the neighbors. If a sensor node is physically destroyed (or data modified) by $\mathcal{ADV}\_del$ (or $\mathcal{ADV}\_md$), $\mathcal{MS}$ can retrieve the copy of original data from the neighbors of sensors. Therefore, better data reliability will be achieved. Note that storing data replica on neighbor nodes increases DSP, however, increases SO of the whole network.

To solve this problem, the authors in [109] proposed a Reed-Solomon codes and Secret Sharing based scheme (RSSS). Since both (n, m) secret sharing scheme and (n, k) Reed-Solomon codes can divide data into n shares ($m < n, k < n$), any combination more than m (or k) can reconstruct the original data. In other words, even some data shares stored in neighbor nodes are physically destroyed (or modified) by $\mathcal{ADV}\_del$ (or $\mathcal{ADV}\_md$), $\mathcal{MS}$ still can recover the original data if more than m shares are available. The advantage of (n, m) secret sharing scheme is that it can provide high level security, where any combination of less than m secret shares cannot reveal the secret. The weak point of the scheme is that the size of each encoded data share is still the same as the size of original data. Compared with secret sharing scheme, the size of each share encoded by (n, k) Reed-Solomon codes is $1/n$ of original data, which is much smaller than the size of shares in secret sharing schemes. However, Reed-Solomon codes cannot provide security as secret sharing scheme does.

Combining the advantages of both Reed-Solomon codes and secret sharing scheme, a hybrid scheme was proposed, in which data d_i^r sensed in round r are split into n data shares by (k, n) Reed-Solomon codes denoted as $pt_{i,1}^r, pt_{i,2}^r, \cdots, pt_{i,n}^r$, and secret key is divided into n shares by (m, n) secret sharing scheme denoted as $S_{i,1}^r, S_{i,2}^r, \cdots, S_{i,n}^r$. The sensor s_i then randomly selects n neighbor nodes from NB_i and sends the $pt_{i,t}^r$ and $S_{i,t}^r$ to them. To recover the original data, $\mathcal{MS}$ only needs to retrieve at least m shares of data and k secret key shares. Thus, $RSSS$ reduces the SO and provides certain data reliability.

Discussion. We observe that $RSSS$ has the following weaknesses.

- To retrieve data generated in R rounds, $\mathcal{MS}$ has to retrieve data R times from a sensor node that stores R rounds data. The sensor node can also pack R rounds data into one packet, and send it to $\mathcal{MS}$. However, the length of the packet is still $R * \lceil log_2(d_i^r) \rceil$. This will increase TC as well.
- The encrypted data shares $Enc_{\mathcal{K}_i^r}(pt_{i,t}^r)$ which are generated by s_i during r rounds still need storage cost $r * \lceil log_2(Enc_{\mathcal{K}_i^r}(pt_{i,t}^r)) \rceil$ in neighbor node s_j, where $\mathcal{K}_i^r$ is the secret key of s_i during round r. As r increases, the SO increases too.
- $RSSS$ provides neither forward secrecy nor backward secrecy.

From discussion above, we have the following observations. Before $\mathcal{MS}$ retrieves the data, the sensor s_i generates R rounds data. So the storage cost of encrypted data in sensor memory can be denoted as $\sum_{r=1}^{R} \lceil log_2(d_i^r) \rceil$. If the value R tends to be large, e.g., 1000000, the storage cost will increase dramatically. Since our goal is to get expected value and variance of data, we transform Eq. (4.2) as following:

$$\begin{aligned} Var_i &= \frac{\sum_{r=1}^{R}(d_i^r)^2}{R} - \left(\frac{\sum_{r=1}^{R} d_i^r}{R}\right)^2 \\ &= \frac{\sum_{r=1}^{R}(d_i^r)^2}{R} - \frac{(\sum_{r=1}^{R} d_i^r)^2}{R^2} \qquad (4.3) \end{aligned}$$

Given $Sum_i^R = \sum_{r=1}^{R} d_i^r$ and $SumSquare_i^R = \sum_{r=1}^{R}(d_i^r)^2$, Eq. (4.1) and Eq. (4.3) can be further denoted as

$$E(d_i) = \frac{Sum_i^R}{R}, \qquad (4.4)$$

and

$$Var_i = \frac{SumSquare_i^R}{R} - \frac{(Sum_i^R)^2}{R^2}. \qquad (4.5)$$

In other words, to compute $E(d_i)$ and Var_i, $\mathcal{MS}$ needs to get Sum_i^R, $SumSquare_i^R$, and R instead. Note that the storage cost of sensors can be largely reduced if Sum_i^R and $SumSquare_i^R$ are stored, instead of storing the encryption of data generated in each rounds.

Moreover, we observe that if the sensor keeps the secret unchanging, all encrypted data can be read by $\mathcal{ADV}$, no matter the data are generated before or after the compromised period, since it only needs to compromise the sensor once and gets the secret key consequently. Thus, these observations motivate us to design next scheme that can reduce storage cost and provide both forward secrecy and backward secrecy.

4.2.3 Homomorphic Encryption and Key Evolution Based Scheme (HKS)

Let $\mathcal{K}_i^r$ be the secret key of sensor s_i in round r. To deal with the limitations in RSSS, we need to change the secret key after each round to guarantee that the $\mathcal{ADV}$ who holds the secret key $\mathcal{K}_i^r$ in compromised period $r \in [r_a, r_b]$ cannot derive the secret key $\mathcal{K}_i^{\hat{r}}$ in the previous rounds $\hat{r} \in [0, r_a)$. We utilize key evolution approach [6], i.e., the secret key of a sensor node is updated by its owner. The secret key of s_i in round r is computed as $\mathcal{K}_i^r = h(\mathcal{K}_i^{r-1})$, where $h(\cdot)$ is an one-way hash function ($\mathcal{K}_i^{r-1}$ is then securely erased.). After the secret key $\mathcal{K}_i^r$ is updated by s_i at the end of round r, $\mathcal{ADV}$ cannot derive the previous round's key before the sensor was compromised (due to one-way property of $h(\cdot)$). Moreover, $\mathcal{MS}$ can easily compute the corresponding secret key of sensor s_i in round r for $\mathcal{K}_i^r = h^{r-1}(\mathcal{K}_i) = h^{r-1}(\mathcal{K}_m||i)$. Here, $||$ stands for the concatenation operator.

In addition, we utilize additive homomorphic encryption [14] to encrypt new sensed data such that sum of encrypted data (denoted as $SumEnc = \sum_{r=1}^{R} Enc(d_i^r)$) corresponds to encryption of those data summation (denoted as $EncSum = Enc(\sum_{r=1}^{R} d_i^r)$). To encrypt a data d_i^r, the sensor s_i do

$$Enc(d_i^r) = Enc(d_i^r, \mathcal{K}_i^r, M) = d_i^r + \mathcal{K}_i^r \pmod M,$$

where $d_i^r \in [0, M-1]$ and M is a large integer. Due to the properties of additive homomorphic encryption, we can obtain $Enc(\sum_{r=1}^{R} d_i^r) = \sum_{r=1}^{R} Enc(d_i^r)$ and $Enc(\sum_{r=1}^{R} (d_i^r)^2) = \sum_{r=1}^{R} Enc((d_i^r)^2)$. Thus $EncSum$ and $EncSumSquare =$

$Enc(\sum_{r=1}^{R}(d_i^r)^2)$ can be obtained by

$$EncSum = Enc(\sum_{r=1}^{R} d_i^r) = \sum_{r=1}^{R} Enc(d_i^r) = SumEnc,$$

and

$$EncSumSquare = Enc(\sum_{r=1}^{R}(d_i^r)^2) = \sum_{r=1}^{R} Enc((d_i^r)^2) = SumEncSquare.$$

To decrypt the $EncSum = Enc(\sum_{r=1}^{R} d_i^r)$ and $EncSumSquare = Enc(\sum_{r=1}^{R}(d_i^r)^2$, similar to Eq. (4.6) and Eq. (4.7), after computing the corresponding secret key as $\sum_{r=1}^{R} \mathcal{K}_i^r = \sum_{r=1}^{R} h^{r-1}(\mathcal{K}_m||i)$, $\mathcal{MS}$ can get $SumEnc = \sum_{r=1}^{R} Enc(d_i^r)$ and $SumEncSquare = \sum_{r=1}^{R} Enc((d_i^r)^2)$ as following

$$\begin{aligned} Sum &= Dec(EncSum) = Dec(SumEnc) \\ &= Dec(\sum_{r=1}^{R} Enc(d_i^r)) \\ &= \sum_{r=1}^{R} Enc(d_i^r) - \sum_{r=1}^{R} \mathcal{K}_i^r (mod\ M) \\ &= \sum_{r=1}^{R} d_i^r \end{aligned} \tag{4.6}$$

and

$$\begin{aligned} SumS &= Dec(EncSumSquare) = Dec(SumEncSquare) \\ &= Dec(\sum_{r=1}^{R} Enc((d_i^r)^2)) \\ &= \sum_{r=1}^{R} Enc((d_i^r)^2) - \sum_{r=1}^{R} \mathcal{K}_i^r (mod\ M') \\ &= \sum_{r=1}^{R} (d_i^r)^2, \end{aligned} \tag{4.7}$$

where $(d_i^r)^2 \in M'$ and M' is a large integer.

Algorithm 1 shows the homomorphic encryption and decryption operation process in sensor s_i. For more detailed information about homomorphic encryption, please refer to Section 2.4.

Algorithm 1: Homomorphic encryption and Key evolution based Scheme

```
Input  : $d_i^r$
Output: $EncSum_i^r$, $EncSumSquare_i^r$
/* sensor $s_i$ starts round r. */;
collect new sensed data $d_i^r$;
compute $Enc(d_i^r) = Enc(d_i^r, \mathcal{K}_i, M) = d_i^r + \mathcal{K}_i \pmod{M}$;
compute $Enc((d_i^r)^2) = Enc((d_i^r)^2, \mathcal{K}_i, M') = (d_i^r)^2 + \mathcal{K}_i \pmod{M'}$;
if $r = 1$ then
    $SumEnc_i^r = Enc(d_i^r)$;
    $SumEncSquare_i^r = Enc((d_i^r)^2)$;
else
    compute $SumEnc_i^r \leftarrow SumEnc_i^{r-1} + Enc(d_i^r)$;
    compute $SumEncSquare_i^r \leftarrow SumEncSquare_i^{r-1} + Enc((d_i^r)^2)$;
end
store $SumEnc_i^r$, $SumEncSquare_i^r$ on local storage;
/* end round r. */;
```

Computation of expected value and variance. After $\mathcal{MS}$ retrieves the $SumEnc$ and $SumEncSquare$, it can decrypt them by Eq. (4.6) and Eq. (4.7) to get Sum and $SumSquare$ using the corresponding secret key $\sum_{r=1}^{R} \mathcal{K}_i^r$. Then, the expected value and variance can be easily computed by Eq. (4.4) and Eq. (4.5).

Proposition 4.3. *The storage cost of Sum_i^R and $SumSquare_i^R$ is lower than storing the encryptions of data generated in each rounds, if $R > 1$ and $log_2(d_{max}) > 1$, where $d_{max} = MAX(d_i^r)$.*

Proof. The storage cost of Sum_i^R and $SumSquare_i^R$ can be denoted as

$$\begin{aligned} & log_2(SumSquare_i^R) + log_2(Sum_i^R) \\ = \;& log_2\big(\sum_{r=1}^{R}(d_i^r)^2\big) + log_2\big(\sum_{r=1}^{R} d_i^r)\big) \\ \leq \;& log_2(R * (d_{max})^2) + log_2(R * d_{max}) \\ = \;& log_2 R + 2log_2(d_{max}) + log_2 R + log_2(d_{max}) \\ = \;& 2log_2 R + 3log_2(d_{max}) \;\; . \end{aligned}$$

The storage cost of the encryptions of data generated in each rounds can be denoted as

$$\begin{aligned}
&\sum_{r=1}^{R} log_2((d_i^r)^2) + \sum_{r=1}^{R} log_2(d_i^r) \\
&= 2\sum_{r=1}^{R} log_2(d_i^r) + \sum_{r=1}^{R} log_2(d_i^r) \\
&= 3\sum_{r=1}^{R} log_2(d_i^r) \\
&\leq 3Rlog_2(d_{max}) \ .
\end{aligned}$$

When $R > 1$ and $log_2(d_{max}) > 1$, $3Rlog_2(d_{max}) - 2log_2 R - 3log_2(d_{max}) > 0$. Therefore, we have

$$log_2(SumSquare_i^R) + log_2(Sum_i^R) < \sum_{r=1}^{R} log_2((d_i^r)^2) + \sum_{r=1}^{R} log_2(d_i^r),$$

if $R > 1$ and $log_2(d_{max}) > 1$. □

In UWSN, given that R tends to a very large value, e.g., 1000000, and $\lceil log_2(d_i^r) \rceil$ is a fixed small value, the storage cost is reduced significantly (as shown in Fig. 4.1) if Sum_i^R and $SumSquare_i^R$ are stored instead. Thus, these results motivate us to design a data summation scheme that can reduce storage cost.

Proposition 4.4. *HKS can provide forward secrecy as well as backward secrecy.*

Proof. Since key evolution is adopted in HKS, the $\mathcal{ADV}$ cannot derive the previous key from the current key it holds due to the one way property of hash function, thus the $\mathcal{ADV}$ cannot decrypt the data encrypted in previous rounds, which means the forward secrecy is guaranteed. Moreover, the data generated in previous rounds are stored as $SumEnc$ and $SumEncSquare$ in sensor nodes. To decrypt them, the corresponding secret key $\sum_{r=1}^{R} \mathcal{K}_i^r$ is needed, which consists of all keys generated in each round. Since the previous keys cannot be derived from current key, the corresponding key $\sum_{r=1}^{R} \mathcal{K}_i^r$ cannot be obtained by $\mathcal{ADV}$, that means backward secrecy is guaranteed. Therefore, HKS can guarantee both forward secrecy and backward secrecy. □

Discussion. As UWSNs operate in unattended environment, sensor nodes may fail due to various reasons such as battery depletion, natural disaster, random failure, etc. Once a sensor loses its functionality totally, all the data it has stored cannot be retrieved by $\mathcal{MS}$. To provide data reliability, we propose the scheme in the following subsection.

4.2.4 Homomorphic Encryption and Homomorphic Secret Sharing Based Scheme ($HEHSS$)

We observe that HKS can provide both forward secrecy and backward secrecy as well as low SO and TC, but cannot guarantee data reliability.

Next we propose a Homomorphic Encryption and Homomorphic Secret Sharing based scheme ($HEHSS$). It takes the advantage of both $RSSS$ and HKS for providing both forward secrecy and backward secrecy as well as keeping low SO, TC, and DSP (reliability).

The scheme contains following steps:

Step 1: System initialization. Initially, the sink picks a secret key denoted as $\mathcal{K}_m$. A sensor node s_i initiates a round index r to 1, as well as shares a secret key $\mathcal{K}_i^1$ with the sink. The $\mathcal{K}_i^1 = h(\mathcal{K}_m||i)$ is originally derived from the secret key $\mathcal{K}_m$ by using a hash function h. The round index r and the encryption key $\mathcal{K}_i^r$ are updated after each round, i.e., $r \leftarrow r+1$, and $\mathcal{K}_i^r = h(\mathcal{K}_i^{r-1})$. Thus, the sink only needs to store a single secret key $\mathcal{K}_m$ and secret key $\mathcal{K}_i^r$ for each round can be derived as needed.

Step 2: Distributed data storage. s_i employs (n,m) homomorphic secret sharing scheme to split d_i^r into n shares, denoted as $S_{i,1}^r, S_{i,2}^r, ..., S_{i,n}^r$, secondly selects n neighbors in NB_i based on node selection scheme addressed in Chapter 5, and finally distributes randomly share $S_{i,t}^r$ to s_j by using current $\mathcal{K}_i^r$ to encrypt the packet.

$$s_i \mapsto s_j : \{Enc_{\mathcal{K}_i^r}(S_{i,t}^r), r, i\}, t \in [1,n],$$

where $Enc_{\mathcal{K}_i^r}(S_{i,t}^r) = S_{i,t}^r + \mathcal{K}_i^r\,(mod\,M)$.

In contrast to the data storage in scheme $RSSS$ [109], s_j adds the encrypted shares $Enc_{\mathcal{K}_i^r}(S_{i,t}^r)$ to previously received shares $\sum_{r=1}^{R} Enc_{\mathcal{K}_i^{r-1}}(S_{i,t}^{r-1})$, when s_j receives $\{Enc_{\mathcal{K}_i^r}(S_{i,t}^r), r, i\}$ from s_i. For the purpose of comparison, Table 4.1 shows the differences between data storage in $RSSS$ and $HEHSS$. We observe that $HEHSS$ can store encrypted data generated during R rounds into a single memory unit, which drastically reduces SO. Furthermore, $\mathcal{MS}$ only needs to retrieve a data block $\{\sum Enc_{\mathcal{K}_i^R}(S_{i,t}^R), R, i\}$, when it periodically visits the

```
Algorithm 2: Calculate $y = \sum_{r=1}^{R} d_i^r = Sum$
  Require: $m \geq 0$
      $S_{i,1}^r, S_{i,2}^r, \cdots, S_{i,n}^r \Leftarrow HSSEncode(d_i^r)$
  Ensure : $\sum_{r=1}^{R} d_i^r = Sum$
      $EncSumSquare_t = SumEncSquare_t, t \in [1, n]$
      $EncSum \leftarrow$
  $HSSRecover(EncSumSquare_{i,1}^r, \cdots, EncSumSquare_{i,m}^r)$,
      $Sum \leftarrow Dec(EncSum)$
```

UWSN and retrieves data. Therefore, $HEHSS$ also reduces TC.

Step 3: Data reconstruction. $\mathcal{MS}$ collects m shares from nodes and reconstructs data using homomorphic secret sharing scheme and homomorphic encryption. Algorithm 2 briefly describes the algorithm executed at $\mathcal{MS}$ when receiving sum of encrypted shares (the sum of data square can be obtained by using the same procedure).

Theorem 4.5. *In $HEHSS$, $\mathcal{MS}$ can reconstruct the data correctly.*

Proof. (Sketch) The homomorphic encryption can also be combined with homomorphic secret sharing, e.g., there are an operation $\oplus$ on the shares, and an operation $\otimes$ on the encrypted shares such that for all participants

$$Enc_{\mathcal{K}_i^r}(S_{i,t}^r) \otimes Enc_{\mathcal{K}_i^r}(S_{i,t'}^r) = Enc_{\mathcal{K}_i^r}(S_{i,t}^r \oplus S_{i,t'}^r)$$

Thus by decrypting the encrypted data shares $Enc_{\mathcal{K}_i^r}(S_{i,t}^r) \otimes Enc_{\mathcal{K}_i^r}(S_{i,t'}^r)$, the recovered secret will be equal to $S_{i,t}^r \oplus S_{i,t'}^r$, assuming that the underlying secret sharing scheme is $\oplus$ homomorphic. □

Remarks: Our scheme can be customized via concrete secret sharing scheme

Table 4.1: Difference of data storage between $RSSS$ and $HEHSS$

$\{Enc_{\mathcal{K}_i^1}(S_{i,t}^1), 1, i\}$
$\vdots$
$\{Enc_{\mathcal{K}_i^r}(S_{i,t}^r), r, i\}$
$\vdots$
$\{Enc_{\mathcal{K}_i^R}(S_{i,t}^R), R, i\}$

Data storage in RSSS [109]

$\{\sum Enc_{\mathcal{K}_i^R}(S_{i,t}^R), R, i\}$

Data storage in HEHSS

and encryption schemes according to the design requirements. Normally, any additive homomorphic secret sharing schemes can be combined with additive homomorphic encryption schemes.

4.3 Analysis and Numerical Results

In this section, we investigate numeric performance of the proposed schemes with respect to $\mathcal{ADV}\_del$ and $\mathcal{ADV}\_md$.

Table 4.2 illustrates the quantitative performance analysis in terms of SO, TC, and DSP. Fig. 4.1~Fig. 4.4 show analytical results in Table 4.2 in terms of SO, TC, and DSP, where n, m, k are defined as in Section 4.2.

4.3.1 Performance Evaluation

4.3.1.1 SO Analysis

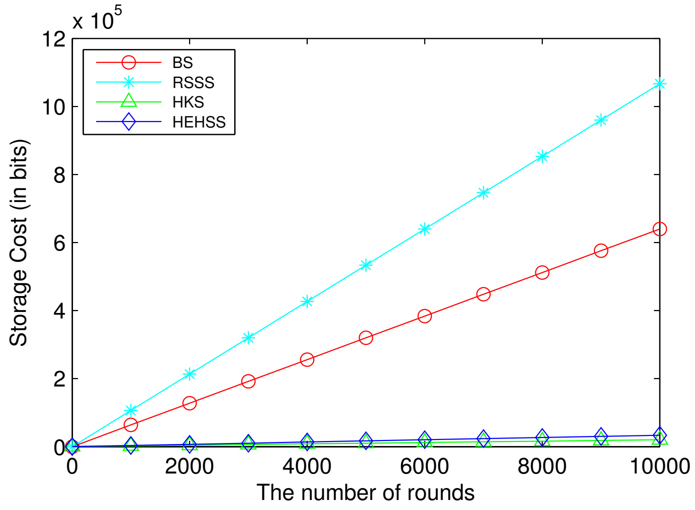


Figure 4.1: The SO comparison between proposed schemes for $n = 10, m = k = 6$, and $|Enc(d_i^r)| = 64$ bits.

In $RSSS$, sensor sends n data shares to n neighbors, which costs $SO_{RSSS} = \frac{n}{k}\sum_{r=1}^{R}|Enc(d_i^r)|$. In HKS, the sum of encrypted data and the sum of encrypted data squares are stored, which significantly reduce the storage cost, e.g., $SO = |\sum_{r=1}^{R} Enc(d_i^r)| + |\sum_{r=1}^{R} Enc(d_i^r)^2|$. Since HKS cannot provide data reliability, $HEHSS$ improves the HKS by encoding the sum of encrypted data and then sending the encoded shares to neighbor nodes to increase DSP, leading to SO increased slightly.

Table 4.2: Quantitative performance analysis between proposed schemes, where $\lvert * \rvert$ means $\lceil log_2 * \rceil$.

Scheme	SO	TC	DSP
DATA-MOVING	$\sum_{r=1}^{R} \lvert (Enc(d_i^r) \rvert$	$\sum_{j=1}^{R} \sum_{r=1}^{j} \lvert Enc(d_i^r) \rvert + \sum_{r=1}^{R} \lvert Enc(d_i^r) \rvert$	$1 - Pr^{ds} - Pr^{md}$
RSSS	$\frac{n}{k} \sum_{r=1}^{R} \lvert Enc(d_i^r) \rvert$	$\frac{n+k}{k} \sum_{r=1}^{R} \lvert Enc(d_i^r) \rvert$	$\begin{cases} 1, n(1-Pr^{ds}-Pr^{md}) \geq k \\ 0, n(1-Pr^{ds}-Pr^{md}) < k \end{cases}$
HKS	$\lvert \sum_{r=1}^{R} Enc(d_i^r) \rvert + \lvert \sum_{r=1}^{R} Enc((d_i^r)^2) \rvert$	$\lvert \sum_{r=1}^{R} Enc(d_i^r) \rvert + \lvert \sum_{r=1}^{R} Enc((d_i^r)^2) \rvert$	$1 - Pr^{ds} - Pr^{md}$
HEHSS	$\frac{n}{m}(\lvert \sum_{r=1}^{R} Enc(d_i^r) \rvert + \lvert \sum_{r=1}^{R} Enc((d_i^r)^2) \rvert)$	$(n+m)(\lvert \sum_{r=1}^{R} Enc(d_i^r) \rvert + \lvert \sum_{r=1}^{R} Enc((d_i^r)^2) \rvert)$	$\begin{cases} 1, n(1-Pr^{ds}-Pr^{md}) \geq m \\ 0, n(1-Pr^{ds}-Pr^{md}) < m \end{cases}$

As shown in Fig. 4.1, the y-axis represents the storage overhead in the whole UWSN. We can observe that $RSSS$ causes higher SO than the other three schemes. Due to the property of homomorphic encryption and homomorphic secret sharing, HKS can store received data share in one data block of size $\lceil log_2 M \rceil$ bits, as shown in Table 4.1. Thus, HKS has the lowest SO among those schemes, and keeps the SO in a very low level. $HEHSS$ increases DSP and the cost of SO more than HKS.

4.3.1.2 TC Analysis

Compared with other three schemes, BS causes very large TC, when data are KEEP-MOVING at each round.

In $RSSS$, sensor node s_i needs to send n encoded data share $|pt_{i,t}^r|$ to selected neighbor nodes $s_j, s_j \in NB_i$. Due to the property of Reed-Solomon codes, the size of encoded share is $1/k$ of the original data. Thus, $|pt_{i,t}^r| = \frac{1}{k} log_2(d_i^r)$. In addition, $\mathcal{MS}$ accesses the UWSN periodically to retrieve sensed data in R rounds. To reconstruct a data packet, the mobile node only needs to retrieve k data shares. That is,

$$\begin{aligned} TC_{RSSS} &= (n+k) * \sum_{r=1}^{R} |pt_{i,t}^r| \\ &= \frac{n+k}{k} \sum_{r=1}^{R} |Enc(d_i^r)|, \;\; t \in [1, \cdots, n]. \end{aligned}$$

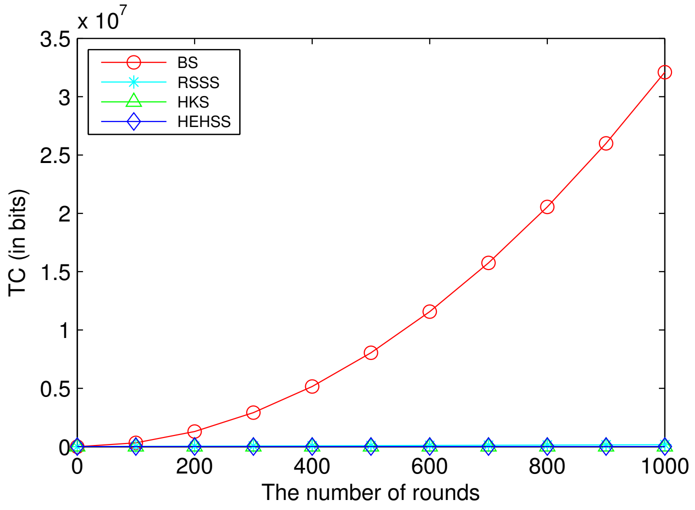


Figure 4.2: The TC comparison between proposed schemes, with $n = 10, m = k = 6$, and $|Enc(d_i^r)| = 64$ bits.

In $HEHSS$, sensor node s_i also needs to send n encrypted data share $|S^r_{i,t}|$ to selected neighbor nodes $s_j, s_j \in NB_i$. To retrieve data sensed in R round, $\mathcal{MS}$ only needs to retrieve compressed block from m neighbor nodes. Due to the property of (n,m) secret sharing, the size of encoded data share equals the size of original data share, $|S^r_{i,t}| = |Enc(d^r_i)|$. That is,

$$\begin{aligned} TC_{HEHSS} &= (n+m)|\sum_{r=1}^{R} S^r_{i,t}| \\ &= (n+m)(|\sum_{r=1}^{R} Enc(d^r_i)| + |\sum_{r=1}^{R} Enc((d^r_i)^2)|), \;\; t \in [1,\cdots,n]. \end{aligned}$$

As shown in Fig. 4.2, we observe that the TC for other three schemes is almost 0 when they compared with BS. Therefore in Fig. 4.3, we ignore the TC of BS, and plot only $RSSS$, HKS and $HEHSS$. We can observe that HKS and $HEHSS$ have the lowest TC for these three schemes.

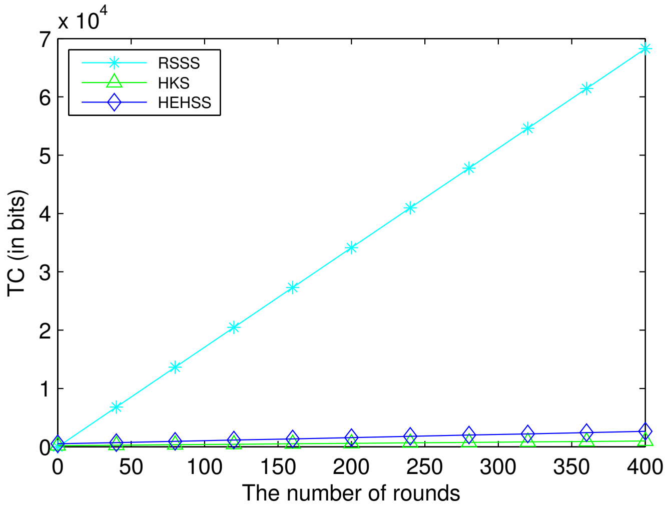


Figure 4.3: Comparison of TC for $RSSS$, HKS and $HEHSS$, when $n = 10, m = k = 6$, and $|Enc(d^r_i)| = 64$ bits.

4.3.1.3 DSP Analysis

For $RSSS$ and $HEHSS$, if $m = k$, both (n,k) Reed-Solomon codes and (n,m) secret sharing scheme need to get m(or k) data shares to reconstruct data. Thus, they have the same effect on DSP. As shown in Fig. 4.4, BS and HKS, which cannot provide data reliability, have the worst result where the DSP tends to 0 when $Pr^{del} + Pr^{md} \rightarrow 1$. The DSP of $RSSS$ and $HEHSS$

is still 1 even if the $Pr^{del} + Pr^{md}$ is up to 0.4. When $Pr^{del} + Pr^{md} > 0.4$, the DSP of $RSSS$ and $HEHSS$ tends to 0 because the number of survival data shares is not enough to reconstruct the original data.

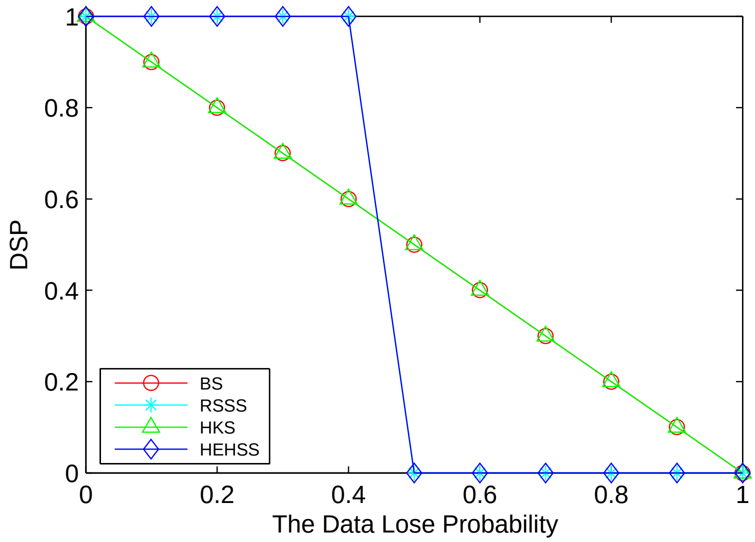


Figure 4.4: Comparison of DSP for $RSSS$, HKS and $HEHSS$, when $n = 10, m = k = 6$.

4.3.2 Security Analysis

For BS, secret may be revealed if nodes stored the original data or data replica are compromised by attackers . Compared with BS, $RSSS$ provides better security because attackers only get encoded data shares when a node is compromised. However, $\mathcal{ADV}$ who got more than k shares still can recover the original data. Once $\mathcal{ADV}$ obtains the secret key, it can decrypt all the data. Since key evolution is adopted in the HKS and $HEHSS$, $\mathcal{ADV}$ cannot derive the previous key from the current key it holds due to the one way property of hash function. Thus $\mathcal{ADV}$ cannot decrypt the data encrypted in previous rounds, which means that forward secrecy is guaranteed. Furthermore, the data generated in the previous rounds are stored as sum of encrypted data and sum of encrypted data squares. To decrypt them, the corresponding secret key $\sum_{r=1}^{R} \mathcal{K}_i^r$ is needed, which consists of all keys generated in each rounds. Since the previous keys cannot be derived from the current key, the corresponding key $\sum_{r=1}^{R} \mathcal{K}_i^r$ cannot be obtained by $\mathcal{ADV}$, meaning that backward secrecy is guaranteed. For (n, m) homomorphic secret sharing scheme, getting fewer than m shares cannot be used to reconstruct the secret. Therefore, even if some sensor nodes are compromised, $\mathcal{ADV}$ still cannot retrieve the data without having enough shares.

Finally, we summarize the comparison results between BS, $RSSS$, HKS, and $HEHSS$ in Table 4.3 in terms of forward secrecy, backward secrecy, DSP, SO, and TC. Table 4.3 shows the conclusion that $HEHSS$ has the best performance compared with other three schemes in terms of forward secrecy, backward secrecy, DSP, SO, and TC.

Table 4.3: Performance comparison results between different schemes.

Scheme	forward secrecy&backward secrecy	Reliability (DSP)	SO	TC
BS	No	Low	High	Very High
RSSS	No	High	High	High
HKS	Yes	Low	Low	Low
HEHSS	Yes	High	Low	Low

4.4 Summary

In this chapter, we have proposed a novel homomorphic encryption and homomorphic secret sharing based scheme for efficient and reliable distributed data storage suitable for UWSNs. Compared with the schemes proposed in Chapter 3, we demonstrated that backward secrecy of *historical* data can be achieved. Detailed analysis and numerical results demonstrate that our scheme accomplishes the goals of forward secrecy, backward secrecy, resilience to node compromises, reliability, and efficiency of storage and transmission.

Chapter 5

Optimized Data Distribution

The previous two chapters present data distribution schemes (i.e., ES and HEHSS) to provide data reliability. These schemes allow sensors to distribute generated data to a subset of their neighbor nodes, which would allow $\mathcal{MS}$ to retrieve data successfully even when certain amount of data are deleted or modified by $\mathcal{ADV}$. The next question we need to answer is how to select a suitable subset of nodes to store data.

In this chapter we further improve ES and HEHSS by using a constrained optimization data distribution scheme. Through extensive simulations, we demonstrate that our scheme achieves significant performance improvement over the benchmark scheme [109].

5.1 Optimized Data Distribution Scheme ($ODDS$)

In this section we try to describe how to select suitable sensor nodes to store data. Generally speaking, various metrics can be selected, such as battery level based, storage capability based, and security level based, etc. In this study we aim at improving backward secrecy and data reliability of the data distribution schemes proposed in the previous two chapters. The security level is chosen as our selection metric.

5.1.1 Problem Formulation

The basic idea of $ODDS$ is that sensor s_i selects top n security level neighbors in its neighbor set NB_i to distribute its data. Inspired by a routing path selection algorithm from [55], we assume each node, s_i, has a probability vector $\mathcal{PV}_i = \{P_{i,j}\}_{j=1}^{nb_i}$ that reflects the security levels of its neighbor nodes in NB_i, where $P_{i,j}$ is the probability that $s_{i,j}$, a neighbor node of s_i, is compromised in time interval T. Here, $P_{i,j}$ could be evaluated from the feedback of certain security monitoring software and/or assigned manually by the $\mathcal{MS}$ based on sensor's physical protection level, the location, or the role of the sensor. For example, the nodes buried under the ground have higher security level (lower $P_{i,j}$) than the nodes exposed, or the nodes deployed in enemy field would have lower security level (higher $P_{i,j}$). Without loss of generality, we further assume that $P_{i,1} \leq P_{i,2} \leq \cdots \leq P_{i,nb_i}$, meaning that the security levels are ordered from high to low.

Given a probability threshold value $PT_{i,}$ s_i can select t_i qualified neighbor nodes that have lower probability of being compromised than the threshold value PT_i, denoted as set $NB_{qlf\_i} = \{s_{i,1}, s_{i,2}, \cdots, s_{i,t_i}\}$, where $P_{i,1} \leq P_{i,2} \leq \cdots \leq P_{i,t_i} \leq PT_i$. Then, the data distribution scheme of s_i can be reformulated as a constrained optimization problem:

$$Minimize\ Pr_{recov}(m, n)$$
$$Subject\, to\ P_{i,j} \leq PT_i$$

where $Pr_{recov}(n, m)$ is the probability that the original data is compromised by an $\mathcal{ADV}$. Given a redundancy factor $\tau = \frac{n}{m}$ of the (n, m) Reed-Solomon codes, the data distribution scheme can be divided into two classes depending on the value of τ, i.e., $\tau = 1$ and $\tau > 1$, which causes the tradeoff between maximum security and data reliability. We address them as follows.

5.1.2 Maximum Security without Redundancy ($\tau = 1$)

To provide maximum security, in other words, minimize $Pr_{recov}(n, m)$, the data distribution scheme must force the $\mathcal{ADV}$ to compromise all the qualified data holders as $m = n$. In the data distribution scheme, s_i encodes data into $n = t_i$ shares and distributes them to the t_i qualified neighbor nodes from NB_{qlf_i}. The $Pr_{recov}(n, m)$ is thus equal to the probability that all t_i nodes are compromised,

$$Pr_{recov}(m, n) = \prod_{j=1}^{t} P_{i,j}\,. \tag{5.1}$$

We can observe that the higher the number of qualified neighbor nodes, the lower the $Pr_{recov}(n, m)$. However, too many t_i may cause large storage and communication overhead. Given a required security level λ_i, and considering storage overhead and communication overhead, s_i can choose the top $n = t'_i(t'_i \leq t_i)$ security level nodes, which satisfies $Pr_{recov}(n, m) = \prod_{j=1}^{t'_i} P_{i,j} \leq \lambda_i$ to distribute the data.

Discussion: When $\tau = 1$, *ODDS* is able to provide maximum security, but it is not resilient to node failure and message failure. If one node loses functionality, or one data share is not delivered, the original data cannot be recovered. Under certain conditions (to be explained in Section 5.3), it may cause low data reliability in case when data shares are distributed to low security level nodes. Moreover, in practical networks, sensors may stop working due to node crash, and messages cannot always be delivered. Therefore, it is necessary to add redundancy to provide data reliability.

5.1.3 Maximum Security with Redundancy ($\tau > 1$)

To provide data redundancy, we must have $m < n$, encoded by a (n, m) Reed-Solomon codes. When $m < n$, if α $(\alpha \leq n - m)$ data shares are corrupted or lost, the original data can still be recovered. Note that the higher the τ the more reliable the data scheme, but the easier for $\mathcal{ADV}$ to recover the data. The tradeoff is thus, given a required redundancy threshold, e.g., $\tau < 1 + \frac{2}{t_i}$ [1], how to distribute data shares among nodes that satisfy the required security level in order to obtain the maximum security while having the maximum data reliability. A sensor, s_i, splits data into $n = t_i$ shares and distributes them to t_i qualified neighbor nodes from NB_{qlf_i}. Considering that the data redundancy is upper-bounded by $\tau < 1 + \frac{2}{t_i}$, to maximize the data reliability, m can be chosen as

$$m > \frac{nt}{t+2}\,. \tag{5.2}$$

Thus, it is easy to see that *ENtext* can be recovered by the $\mathcal{ADV}$, only if the $\mathcal{ADV}$ compromised at least m nodes in $\{s_{i,1}, s_{i,2}, \cdots, s_{i,n}\}$, which has the probability

$$\prod_{j=1}^{m} P_{i,j} \leqslant Pr_{recov} \leqslant \prod_{j=n-m}^{n} P_{i,j}\,. \tag{5.3}$$

To reduce storage overhead and communication overhead, s_i can choose the top $n = t'_i(t'_i \leq t_i)$ security level nodes, which satisfies $Pr_{recov}(n, m) =$

[1] Here, the given redundancy threshold could be any value larger than 1.

$\prod_{j=1}^{t_i'} P_{i,j} \leq \lambda_i$ to distribute the data, for a given required security level λ_i.

5.1.4 A Numerical Example

We give a simple example to demonstrate how the node selection scheme works. For simplicity, we assume that a sensor, s_9, has 7 neighbor nodes, denoted as $NB_9 = \{s_{9,1}, s_{9,2}, \cdots, s_{9,7}\}$, with $\mathcal{PV}_9 = \{5\%, 5\%, 10\%, 10\%, 20\%, 30\%, 40\%\}$. Given a threshold value $PT_9 = 25\%$, it is easy to select the qualified nodes as set $NB_{qlf\_9} = \{s_{9,1}, s_{9,2}, \cdots, s_{9,5}\}$. At round 8, s_9 generates data d_9^8, encrypts it into $ENtext_9^8$, encodes $ENtext_9^8$ into $n = t_9 = 5$ shares, distributes these 5 shares to all the 5 nodes in NB_{qlf_9}. It then follows the steps below depending on τ.

1. $\tau = 1$. Since $m = n = 5$, it forces the $\mathcal{ADV}$ to compromise all the 5 nodes to recover the $ENtext_9^8$ with probability $Pr_{recov}(5,5) = \prod_{j=1}^{5} P_{i,j} = 5\% \cdot 5\% \cdot 10\% \cdot 10\% \cdot 20\% = 0.0005\%$. To compromise the BSe of s_9, the $\mathcal{ADV}$ has to recover $ENtext_9^8$. On the other hand, it has to compromise s_9 to get the key secret to decrypt the $ENtext_9^8$. Assuming $P_9 = 20\%$ is the probability of s_9 of being compromised by the $\mathcal{ADV}$, the probability of BSe of s_9 to be compromised is $Pr_{BSe\_comp} = P_9 {\cdot} Pr_{recov}(5,5) = 0.0001\%$.

2. $\tau > 1$. Based on Eq. (5.2), m should be chosen as $m = 4$ $(m > \frac{25}{7})$. If 1 $(n - m = 1)$ data share is corrupted or lost, the $ENtext_9^8$ can still be recovered. The $\mathcal{ADV}$ has to compromise at least 4 nodes to recover the $ENtext_9^8$ with probability $0.0025\% \leq Pr_{recov}(4,5) \leq 0.01\%$. Given $P_9 = 20\%$, the BSe of s_9 to be compromised is $0.0005\% \leq Pr_{BSe\_comp} \leq 0.002\%$.

5.2 Analysis

In this section, we analyze the security aspects of $ODDS$.

5.2.1 Security Analysis

Theorem 5.1. *Let PT_i be a probability threshold, τ be the redundancy factor and P_i be the probability of s_i to be compromised in time interval T. If* Conditions *1-3 of* Lemma *3.2 are satisfied then the probability Pr_{BaS_comp} that the*

$\mathcal{ADV}$ compromises the backward secrecy of s_i is as following:

$$\begin{cases} Pr_{\text{BaS\_}comp} = 0\,, & k < m \\ \prod_{j=1}^{m} P_{i,j}P_i \leqslant Pr_{\text{BaS\_}comp} \leqslant \prod_{j=n-m}^{n} P_{i,j}P_i\,, & k > m, \tau < 1, P_{i,j} \leq PT_i \\ Pr_{\text{BaS\_}comp} = \prod_{j=1}^{t} P_{i,j}P_i\,, & k > m, \tau = 1, P_{i,j} \leq PT_i\,. \end{cases}$$

Proof. Since an $\mathcal{ADV}$ can compromise backward secrecy only if the $\mathcal{ADV}$ can compromise s_i to get its secret key $\mathcal{K}_i^r$ and compromise at least its m neighbor nodes that store the data shares to recover the $ENtext_i^r$, the $\mathcal{ADV}$ thus can use $\mathcal{K}_i^r$ to decrypt the $ENtext_i^r$ to get the $PLtext_i^r$. As proved in *Lemma* 3.2, one can see that

$$Pr_{BaS\_comp} = Pr(C\,1) \cdot Pr(C\,2) \cdot Pr(C\,3)\,,$$

where *C1, C2* and *C3* refer to *Condition 1, 2* and *3*, respectively.

Case 1: $k < m$. That is $Pr(C\,2) = 0$. The $\mathcal{ADV}$ does not have the ability to compromise at least m sensors within $\mathcal{MS}$'s visiting interval. Thus, it cannot recover $ENtext_i^r$. Hence, it cannot compromise the backward secrecy, i.e., $Pr_{BaS\_comp} = 0$.

Case 2: $k > m, \tau < 1$, i.e., $Pr(C\,2) = 1$. The $\mathcal{ADV}$ has the ability to get enough data shares to recover $ENtext_i^r$. Given a threshold probability PT_i, the probability Pr_{recov} is computed in Eq. (5.3). Thus,

$$\begin{aligned} Pr_{BaS\_comp} &= Pr(C\,1) \cdot Pr(C\,2) \cdot Pr(C\,3) \\ &= Pr(C\,1) \cdot 1 \cdot Pr(C\,3) \\ &= P_i \cdot Pr_{recov}\,, \end{aligned}$$

that is

$$\prod_{j=1}^{m} P_{i,j}P_i \leq Pr_{BaS\_comp} \leq \prod_{j=n-m}^{n} P_{i,j}P_i\,.$$

Case 3: $k > m, \tau < 1$, i.e., $Pr(C\,2) = 1$. Similar to Case 2, given a PT_i, the probability Pr_{recov} is computed in Eq. (5.1). Thus,

$$\begin{aligned} Pr_{BaS\_comp} &= Pr(C\,1) \cdot Pr(C\,2) \cdot Pr(C\,3) \\ &= Pr(C\,1) \cdot 1 \cdot Pr(C\,3) \\ &= P_i \cdot Pr_{recov} \\ &= \prod_{j=1}^{t} P_{i,j} P_i\,. \end{aligned}$$

□

5.3 Performance Evaluation

In this section, we show a comparison between the results obtained through a custom-built simulator we developed using MATLAB.

Consider a UWSN where 200 nodes are randomly distributed in a 500m by 500m area. Each sensor node has a transmission range equaling to TR = 60m. The simulation results are averaged over 100 randomly deployed networks. As one snapshot of the network topology shown in Fig. 5.1, nodes are divided into four sets with different compromise probability P_i: 20% of nodes with probability P_i = 50%; 30% of nodes with P_i = 40%; 30% of nodes with P_i = 20%; and 20% of nodes with P_i = 10%, respectively. Table 5.1 shows the detailed simulation parameter configuration.

The probability threshold values of $ODDS$ are set as PT_i = 15%, 30% and 45%, respectively. The probability threshold value PT_i introduces that node with $P_i > PT_i$ is considered too risky to allocate data shares and would not be selected based on node selection scheme. The required security level is set as λ_i = 0.1%. Based on the specified PT_i, sensors are divided into two categories: qualified sensors and disqualified sensors. When PT_i = 15%, only 20% sensors are qualified. Whereas 50% sensors and 80% sensors are qualified when PT_i is specified to 30% and 45% respectively. We will analyze later the impact of PT_i based on the simulation results.

Since both [60] and [21] are operated in *ideal* network without node and message failure, we conduct simulations to compare data local storage schemes (i.e., NS, AS, HKS) and RSSS [109] with optimized data distribution scheme, $ODDS$ improved based on ES and HEHSS. In addition, since the authors in RSSS [109] did not address how to specify n and m, to compare with the

ODDS, we set the RSSS [109] having the same n and m with the *ODDS*. The simulation results are divided into two classes depending on the redundancy factor τ.

Table 5.1: Simulation parameter configuration.

Area	500m * 500m			
N	200 nodes			
Ratio	20%	30%	30%	20%
P_i	50%	40%	20%	10%
Transmission range	60m			

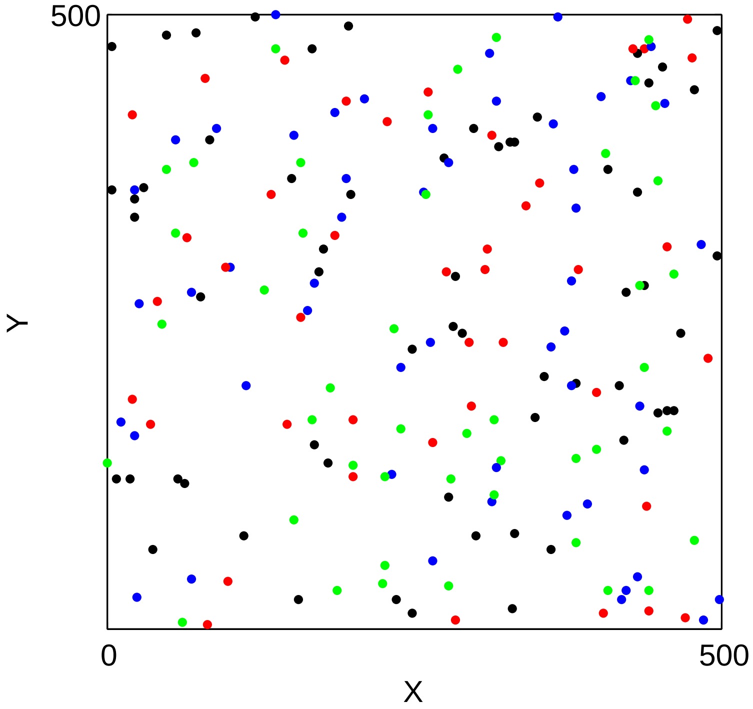


Figure 5.1: Network topology. 200 sensor nodes: 20% red nodes, 30% blue nodes, 30% black nodes, 20% green nodes. With different compromise probability: red = 50%, blue = 40%, black = 20%, green = 10%.

5.3.1 Maximum Security without Redundancy ($\tau = 1$)

Fig. 5.2 shows the simulation results of *ODDS* with maximum security without redundancy (discussed in Subsection 5.1.2).

5.3.1.1 Impacts of PT_i

The simulation results with respect to $PT_i = 15\%$, 30% and 45% are depicted as the first column, the second column and the third column of Fig. 5.2, respectively. The performances of NS, AS and HKS are almost the same no matter PT_i is specified. The reason is that NS, AS and HKS are not designed to distribute data based on PT_i.

The first row and the third row of Fig. 5.2 show the simulation results of the probability of BSe to be compromised Pr_{BSe_comp} in terms of node index and the number of neighbor nodes nb_i, respectively. We observe that the scheme in [109] and $ODDS$ have much lower Pr_{BSe_comp} than NS, AS and HKS. The reason is that data distribution makes $\mathcal{ADV}$ harder to compromise BSe. Moreover, we also observer that Pr_{BSe_comp} decreases as PT_i increases. This is because only 20% nodes can be selected as qualified nodes when $PT_i = 15\%$. Consequently each qualified node has to store more data shares than the qualified nodes in scenarios with $PT_i = 30\%$ and $PT_i = 45\%$. In other words, although data shares are distributed to low risk nodes with low PT_i, fewer qualified nodes result in that $\mathcal{ADV}$ obtains more data shares if one data holder node is compromised leading to higher Pr_{BSe_comp}. The simulation results show that $ODDS$ has the lowest Pr_{BSe_comp} among them, demonstrating optimized data distribution works.

The second row and the forth row of Fig. 5.2 depict the simulation results of the probability of data reliability with respect to node index and nb_i. The simulation results show that increasing PT_i leads to worse performance (probability of data reliability decreasing). This observation coincides with the discussion in Subsection 5.1.2 (maximum security without redundancy), meaning that the original data cannot be recovered if one data share is lost, and data reliability would decrease due to low security level of the data holder. This shows that adding data redundancy is important.

5.3.1.2 Impacts of nb_i

The number of neighbor nodes, nb_i, may be different from node to node. This is why the simulation results are shown as *waves* in the first row and the second row of Fig. 5.2. To further investigate the impact of nb_i on Pr_{BSe_comp} and the probability of data reliability, the simulation results are illustrated in the third row and the forth row of Fig. 5.2. We observe that the performance of Pr_{BSe_comp} gets better (Pr_{BSe_comp} decreasing) as nb_i increases. Whereas, the performance of the probability of data reliability becomes worse (the probability of data reliability decreasing) as nb_i increases. These observations also coincide with the discussion in Subsection 5.1.2, which is another evidence that confirms the discussion. We will show later that the probability of data reliability of $ODDS$ is improved significantly due to providing data redundancy.

5.3.2 Maximum Security with Redundancy ($\tau > 1$)

Providing data redundancy improves data reliability since $\mathcal{MS}$ is able to reconstruct the original data even it cannot obtain all the data shares. However, this advantage is also enjoyed by $\mathcal{ADV}$. Therefore, there is a tradeoff between data reliability and Pr_{BSe_comp}. In the following, we will investigate the impact of data redundancy on data reliability and Pr_{BSe_comp}.

As shown in Figs. 5.3 (A)-(C) and Figs. 5.3 (G)-(I), $ODDS$ has the best performance (the lowest Pr_{BSe_comp}) among all the schemes. Compared with the simulation results shown in Figs. 5.2 (A)-(C) and Figs. 5.2 (G)-(I), providing data redundancy slightly compromises Pr_{BSe_comp}. Whereas, as shown in Figs. 5.3 (D)-(F) and Figs. 5.3 (J)-(L), data reliability is improved significantly due to data redundancy. Again $ODDS$ has the highest probability of data reliability among all the schemes and the data reliability also increases as PT_i increases.

Figs. 5.3 (G)-(I) and Figs. 5.3 (J)-(L) also show the effect of nb_i with respect to Pr_{BSe_comp} and data reliability. We observe that increasing the number of neighbors also benefits to Pr_{BSe_comp} and data reliability. The number of neighbor nodes increases as data reliability increases and Pr_{BSe_comp} decreases. However, distributing data shares to more neighbor nodes also causes more communication overhead. Therefore, according to the results shown in Figs. 5.3 (G)-(I) and Figs. 5.3 (J)-(L), for a given security level λ_i, it is very easy to choose the average number of neighbors when a sensor network is deployed with uniformly distributed nodes. Moreover, as shown in Figs. 5.3 (J)-(L), we observe that the probability of data reliability of $ODDS$ is much higher than RSSS [109] especially when nb_i is small (see the curve in the circle). The reason is that RSSS [109] distributes data shares randomly no matter how low the security level of node is. This is another evidence to show that $ODDS$ has the best performance among NS, AS, HKS and RSSS [109].

Finally, Table 5.2 summarizes the comparison results of different schemes in terms of *FSe*, *BSe*, Resilient to Node Failure (*RNF*) and Resilient to Message Failure (*RMF*). The results demonstrate that the proposed scheme has the best performance among these studied schemes.

Table 5.2: Security and performance comparison result among existed work and enhanced schemes in terms of forward secrecy, backward secrecy, *RNF*, and *RMF*.

	forward secrecy	*backward secrecy*	**RNF**	**RMF**
DISH [60]	*Yes*	*Probabilistic*	*No*	*No*
POSH [21]	*Yes*	*Probabilistic*	*No*	*No*
RSSS *[109]*	*Partial*	*Probabilistic*	*Normal*	*Normal*
ODDS	*Yes*	*Enhanced Probabilistic*	*Strong*	*Normal*

5.4 Discussions

In this section, we describe coding scheme selection, and discuss the drawbacks of *ODDS* and possible improvements.

5.4.1 Coding Scheme Selection

Both (n, m) Reed-Solomon codes and (n, m) *secret sharing* scheme [79] can be used for encoding a data into n data shares, where any m $(m \leq n)$ data shares can be used for recovering the original data. The advantage of (n, m) *secret sharing* scheme is that it can provide excellent security of the secret in the sense of information-theoretic security, where any combination of fewer than m secret shares cannot reveal any information of the secret. The weak point of the scheme is that the size of the share is still the same as the original data. Compared with the secret sharing scheme, a (n, m) Reed-Solomon codes is a kind of erasure code that can encode a block of data into n fragments, and any m can be used to reconstruct the original block. Its advantage is that the size of its encoded block is $\frac{1}{m}$ of the original block. The weak point is Reed-Solomon codes is not so secure as the *secret sharing* scheme.

Multi-level secret sharing scheme [4], which distinguishes the importance of different shares with same share length, can distribute more *weight* shares to nodes with higher security levels. It can reduce communication overhead but increase computation overhead to sensors.

Thus, selecting suitable coding schemes needs to consider the tradeoff between communication cost, storage cost and computation cost depending on practical application scenarios.

5.4.2 Drawbacks and Possible Solutions

There are two drawbacks of the scheme proposed in this chapter:

1) Recall P_i is the probability that node s_i is compromised by the $\mathcal{ADV}$ and can be used for evaluation the security level of s_i (higher P_i corresponds to lower security level). The node selection priority of the proposed optimized data distribution scheme is based on P_i, so that a sensor s_i with the lowest P_i (the highest security level) would *attract* more data from its neighbors. In other words, all the neighbor nodes of s_i would prefer to distribute data to this node because it is the safest choice in their neighbor nodes. Hence, compared with other nodes, s_i has to use more storage memory to store the data distributed by its neighbor nodes. A straightforward solution is to replace low P_i (high security level) nodes with storage sensor node [63] (with large memory) in the network. We are currently working on the scheme that focuses on how to coordinate sensor nodes to support *long-lived* UWSNs.

2) Since the probability, P_i, is maintained in the probability vector of the neighbor nodes of s_i, the neighbor nodes evaluate the priority of s_i based on P_i. Thus, once a node is compromised by an $\mathcal{ADV}$, its security level should be degraded and should be known by its neighbors. Otherwise, the neighbor nodes would still distribute data to it based on its previous security level and the security level of data are undermined. *To solve this problem, a challenge is to dynamically estimate the security level of nodes in UWSNs.* Since a UWSN may consist of many sensors and there is no trusted third party inside, the challenge can convert to how to estimate reputation (security level) in distributed networks without a trusted third party. This observation indeed motivates our study on trust management in UWSNs in the next chapter.

5.5 Summary

In this chapter we proposed a constrained optimization data distribution scheme to maximize security level of data and optimize data reliability. As demonstrated in the chapter, the proposed scheme has low storage overhead, computational efficiency and it is especially suitable for UWSN applications. Compared with other existing schemes [60, 21, 109], our scheme does not rely on reliable nodes and communication channels, and is resilient to message failure and node failure with certain probabilities. Furthermore, through detailed security and efficiency analysis, we show that the proposed scheme can guarantee forward secrecy and maximize probabilistic backward secrecy while providing maximum data reliability. Finally, the simulation results demonstrate that, compared with other schemes, the proposed scheme is more robust to support forward secrecy, backward secrecy, and data reliability.

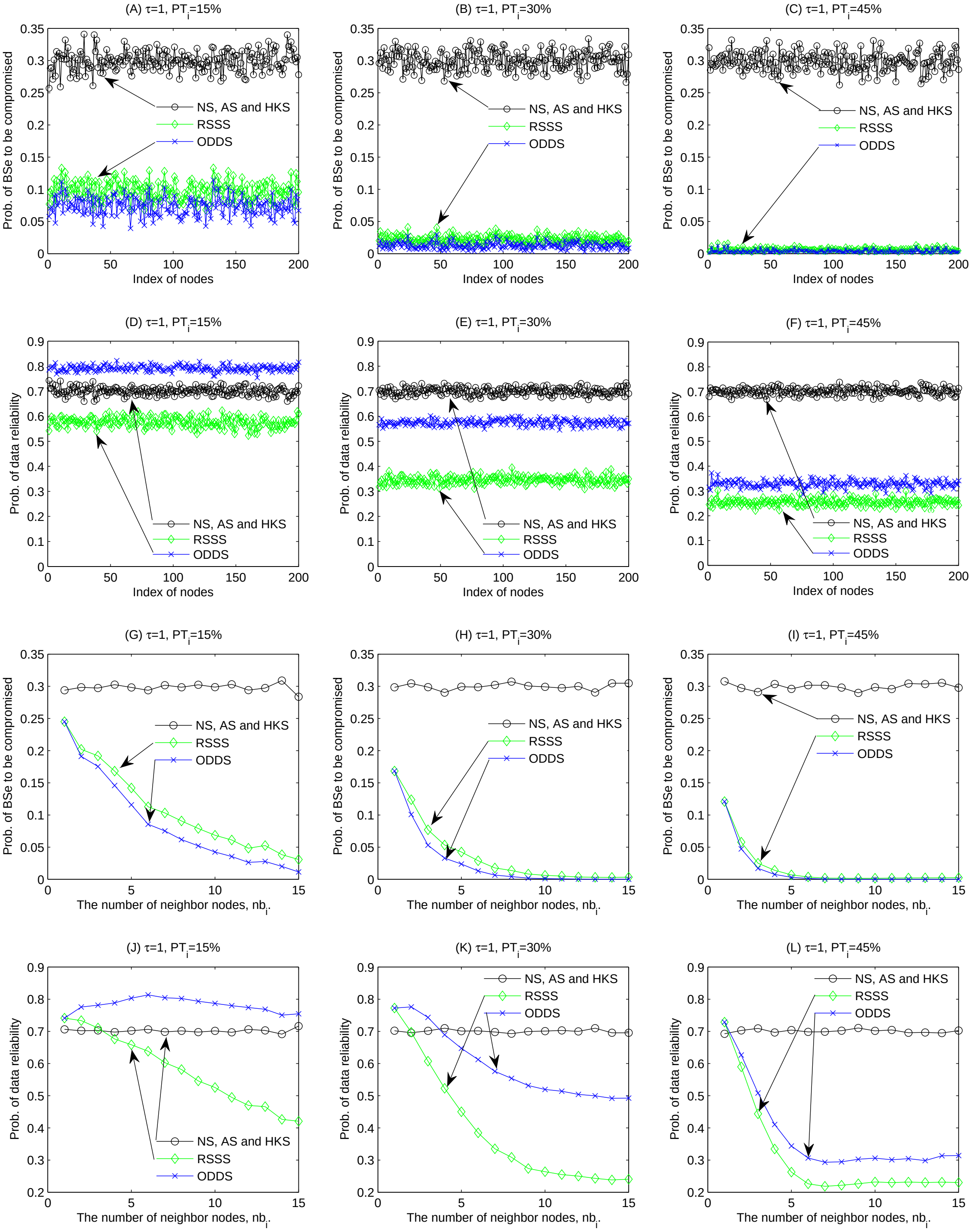


Figure 5.2: The comparison results of different schemes in terms of probability of backward secrecy to be compromised, Pr_{BSe_comp}, and probability of data reliability when $\tau = 1$ and probability threshold value $PT_i = 15\%$, $PT_i = 30\%$ and $PT_i = 45\%$, respectively.

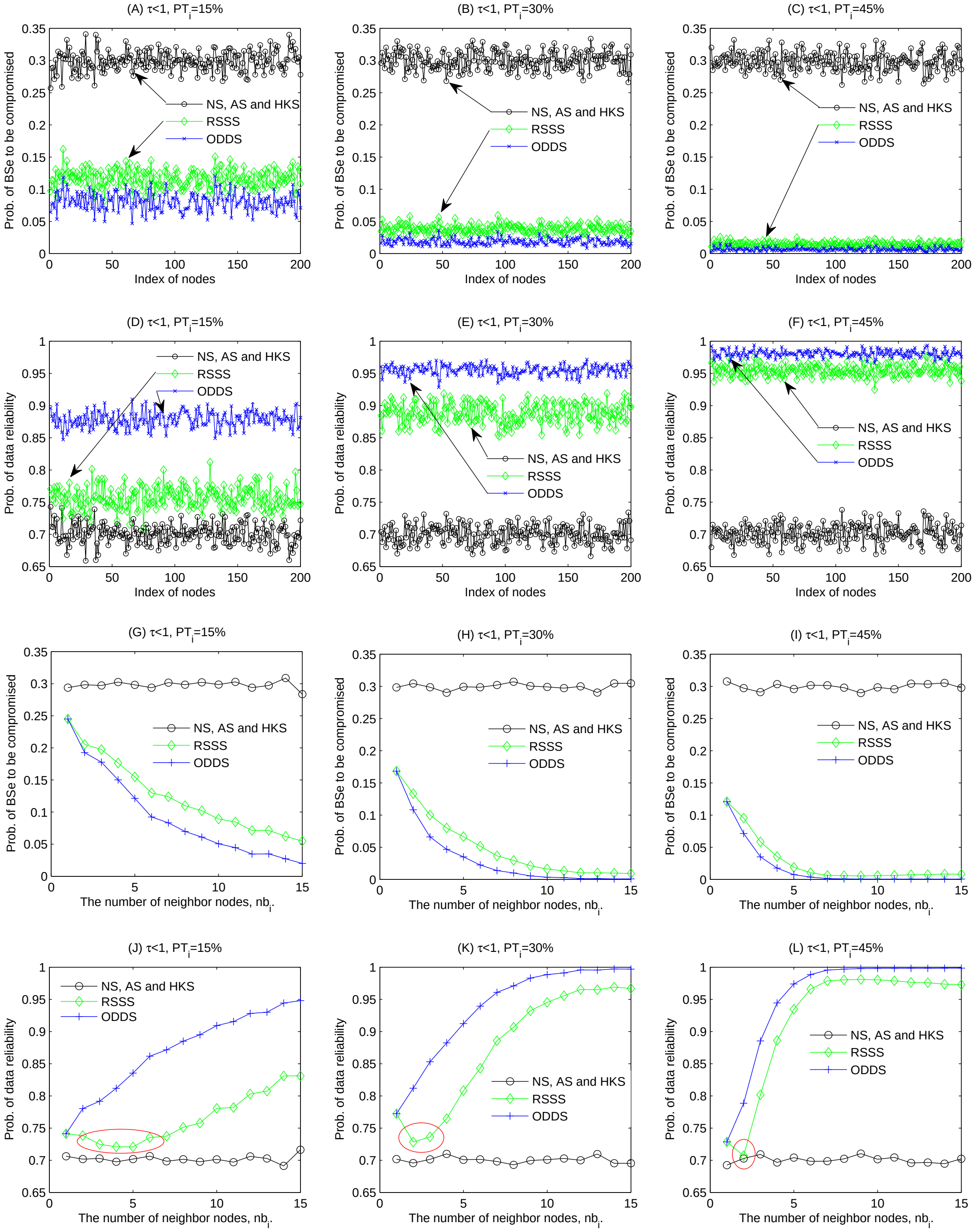


Figure 5.3: The comparison results of different schemes in terms of probability of backward secrecy to be compromised, Pr_{BSe_comp}, and probability of data reliability when $\tau < 1$ and probability threshold value $PT_i = 15\%$, $PT_i = 30\%$ and $PT_i = 45\%$, respectively.

Chapter 6

Trust Management

In the previous chapter we proposed a constrained optimization data distribution scheme, in which sensors send data to their neighbor nodes based on the security level of those neighbor nodes. In this chapter we develop a mechanism to show how the security level of sensors can be evaluated based on their trustworthiness.

In UWSNs, given the absence of the trusted third party, i.e., sink or base station, for storing trust-related data, we have to decide where and how to store trust related information in a secure and reliable way. As communication channels between sensors are generally noisy and unstable, another question is how to deal with such uncertainty. Furthermore, it requires to build a trust management scheme suitable for UWSNs that is efficient, robust and scalable with respect to trust-related data storage and trustworthiness calculations.

6.1 Related Work

Before presenting our own mechanism, we review the existing trust management schemes in WSNs, Ad hoc, and P2P networks.

6.1.1 Trust Management in WSNs

Several solutions have been recently proposed for trust management in WSNs. In [48] the authors considered a TIBFIT protocol to diagnose and to mask arbitrary node failures in an event-driven WSN. Each node is assigned a trust index to track and report events correctly. Cluster heads analyze the event reports using the trust index. In [35], the authors proposed a Bayesian trust management framework where each node maintains reputation metrics to assess past behavior of other nodes and to predict their future behavior. The authors in [113] proposed iTrust, an integrated trust framework for WSNs. A group of monitor nodes is responsible for evaluating neighbor nodes based on their behavior. A trust aware routing for WSNs was proposed in [76]. The protocol exploits prior routing patterns and link quality to determine efficient routes. In [41], the authors proposed a trust-based routing scheme that selects a forwarding path based on the trust requirement of a packet and the trust level of neighbor nodes.

6.1.2 Trust Management in Ad hoc Networks

More trust management studies were conducted in the field of ad hoc networks [12, 85, 73, 67]. The authors in [12] proposed a reputation system based on Bayesian estimation of misbehavior in mobile ad hoc networks. The work in [85] introduced an information theoretic framework to measure trust and to model trust evolution. They proposed four axioms to address trust relationships through third parties. A data-centric framework for trust establishment was proposed in [73]. Here, trust is based on data rather than on the data generating nodes. In [67], the authors proposed a distributed trust scheme based on distributed public key certificate management for mobile ad hoc networks. Each user of the network is permitted to issue public key certificates and to perform authentication using certificate chains without trusted authorities.

6.1.3 Trust Management in P2P Networks

A Peer-Trust model based on a weighted sum of five feedback parameters was proposed in [111]. The model takes advantage of public key infrastructure and trust propagation to secure remote scores and to prevent peer abuses. PowerTrust [119], a robust and scalable P2P reputation scheme, was proposed to leverage the power-law feedback factors. PowerTrust dynamically selects a number of the most reputable nodes, namely, power nodes, based on a distributed ranking mechanism. In addition, it uses a look-ahead random work strategy, leveraging the power nodes to improve global reputation accuracy and aggregation speed. In [108], the authors developed Credence, a decentralized object reputation and ranking system for P2P networks. Credence allows peers to discover trustworthy peers when direct observations or interactions through a flow-based trust computation are not possible.

Trust management solutions developed for traditional WSNs rely on the presence of an online trusted third party, e.g., to store and distribute trust data [48, 41, 76, 35, 113]. They cannot be applied directly to UWSNs due to the absence of the sink (or the base station).

Most schemes, e.g., [12, 85, 73, 67, 111, 119, 108], proposed for P2P and ad hoc networks are not suitable for UWSNs for the following reasons. First, UWSNs are more constrained with respect to computation, communication and power capabilities than P2P and ad hoc networks. Schemes designed for P2P and ad hoc networks based on public key cryptography are not suitable for UWSNs. Second, the number of nodes in ad hoc networks is lower than in UWSNs. A UWSN is likely to have thousands of sensors. P2P networks may have more nodes than UWSNs, but the nodes in P2P networks do not have the same computational and energy constraints as in UWSNs. Finally, sensor nodes provide their services for a lifetime, until their energy is depleted, while P2P nodes enter and exit the networks randomly.

6.2 Network Scenario, Security Model and Design Goals

6.2.1 Network Scenario

In this chapter we continue using the network model defined in Subsection 3.2.1. Additionally, we assume that a sensor, $s_j \in \mathbb{S}$, is located at point p_j and has the transmission range ϕ. Thus s_j at point p_j can communicate with s_m at point p_m if $\mathcal{D}(p_j, p_m) \leq \phi \, (j, m \in \{1, ..., N\})$, where $\mathcal{D}(p_j, p_m)$ is the distance between p_j and p_m. Each sensor $s_j \in \mathbb{S}$ has nb_j neighbors (we say that s_m is one of s_j's neighbors if $\mathcal{D}(p_j, p_m) \leq \phi$), as shown as yellow (light shadow) points in the circle in Fig. 6.1, which compose a set of neighbors of s_j, $NB_j = \{s | s \in \mathbb{S} \text{ and } \mathcal{D}(p_j, p) \leq \phi\}$. s_j's neighbors, $s_{j,i} \in NB_j$ ($i \in \{1, ..., nb_j\}$), are assumed to have their own trust opinions T_i^j regarding s_j's trustworthiness Υ^j [1], and are referred to as *trust producers* of s_j. The nodes storing trust-related data are referred to as *trust managers* TM_j^r

[1] In the context of this chapter, trust opinion T is one sensor's conclusion about the trust level of another sensor; trustworthiness Υ is the weighted-sum of trust opinions over time and across all involved sensors.

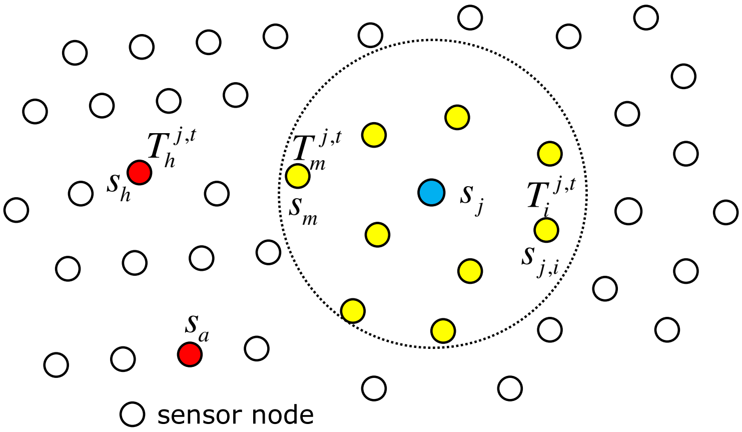


Figure 6.1: An example of network topology.

($r \in \{1, ..., \alpha\}$) of s_j, where α is the number of trust managers in the network. Meanwhile sensors that want to know other sensors' trustworthiness are referred to as *trust consumers*. The relationship between trust producer, trust manager and trust consumer is characterized in Fig. 6.2.

We assume that time is divided into equal time intervals and that sensors maintain loosely synchronized clocks. At time interval t, s_j's neighbor $s_{j,i}$ generates trust opinion $T_i^{j,t}$ regarding s_j. Note that trust consumers can be anywhere in the network but trust producers are only within the transmission rang of the corresponding sensor. Furthermore, there is an $\mathcal{MS}$ visiting the network at either fixed or irregular intervals to collect data from sensors.

6.2.2 Security Model

The UWSNs can be attacked in many ways. In this chapter, we focus on an adversary $\mathcal{ADV}$ launching attacks against trust data [2]. We divide the attacks into two categories: *trust eraser* and *trust pollution* attacks.

The effect of the trust eraser attack (denoted as $\mathcal{ADV}\_Del$) is that trust data stored in sensors are lost and cannot be retrieved by trust consumers. For instance, $\mathcal{ADV}$ could try to compromise sensors and to erase the trust data stored in them. Moreover, when sensors are nonfunctional (e.g., due to energy depletion, natural disasters, etc.) their stored trust data is lost.

In case of trust pollution attack, $\mathcal{ADV}$ does not delete the trust data but rather *pollute* them. We consider the following pollution strategies:

[2] In this chapter trust data means trust related data, such as opinions, trust measurement, etc., which are used by trust management scheme to make trust aware decisions.

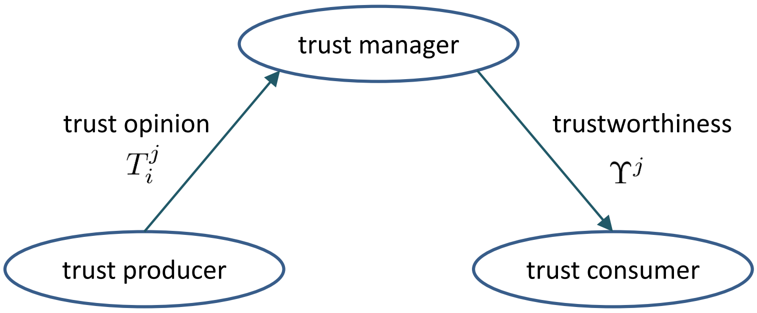


Figure 6.2: The relationship between trust producer, trust manager and trust consumer.

- Environmental effect ($\mathcal{ADV}\_Noise$). Since sensors' trust opinions are generated based on sensors' previous behavior, they may generate some noise due to environmental effects.

- Homogeneous attack ($\mathcal{ADV}\_Homo$). Given a sensor s_j, $\mathcal{ADV}$ tries to either increase s_j trustworthiness Υ^j or decrease it. To do so, in each time interval $\mathcal{ADV}$ can compromise a subset of sensors in order to find the trust managers and to modify the trust data thereby generating false trust opinions.

- Hybrid attack, denoted as $\mathcal{ADV}\_Hbd$. This attack is more severe since $\mathcal{ADV}$ can launch both increasing and decreasing Υ attacks.

For clarity, we assume that the number k of compromised sensors in each time interval is fixed. We referred to this number as the *compromising capability*. The compromised sensor can occur anywhere in the network.

6.2.3 Design Goals

To design an efficient, robust and scalable trust management scheme in UWSNs, the following design goals are targeted.

1) *Robustness*. The scheme is still functional, even though certain sensor nodes totally lose the functionality due to the depletion of battery power or physical corruption (e.g., smash, melt or corrode). Moreover, the trust-related data stored in the system should remain available to queries even if certain sensor nodes fail. That is, the system should be robust against $\mathcal{ADV}\_Del$.

2) *Resilience*. The generated trustworthiness Υ should be close to its real value as much as possible even though $\mathcal{ADV}$ tries to inject false trust opinions

to pollute it. In other words, the system should be resilient to $\mathcal{ADV}\_Noise$, $\mathcal{ADV}\_Homo$ and $\mathcal{ADV}\_Hbd$.

3) *Scalability.* The scheme should apply to a UWSN consisting of a large number of sensor nodes.

4) *Efficiency.* The designed trust management scheme should be efficient in terms of both communication cost and storage cost.

5) *Consistency.* Both trust opinions generated by trust producers and trust queries from trust consumers should be routed correctly to trust managers where the trust-related data are stored.

6.2.4 Performance Metrics

The following metrics are defined to evaluate the performance of our scheme.

- $Pr[survival]_t^j$ is defined as the probability that at least one trust managers of s_j survive to time interval t.
- *Communication Cost*: The communication cost consists of two shares: the cost to send generated trust opinions to trust managers; and the cost of querying and retrieving trustworthiness stored in trust managers. $\mathcal{C}_j$ is defined as communication cost regarding s_j. Since trust value queries and answers are short messages, for the sake of simplicity, we assume that sending and receiving a trust value message across each hop have the same cost, and approximate communication costs as $O(N)$ message transmissions for broadcast and $O(\sqrt{N})$ for point-to-point routing [72].
- *Storage Cost*: $\mathcal{S}_j$ is the storage cost of storing the trust opinions and trustworthiness of s_j, in the whole UWSN.

6.3 Preliminaries

6.3.1 Information Collection on Sensor Behavior:

Generally, there are two common approaches to evaluate trust in WSNs. First, *direct trust* where sensor s_j's behavior can be directly observed by its neighbors $\{s_{j,i}\}_{i=1}^{nb_j}$. Second, *indirect trust* where $s_{j,i}$ receives recommendations from other sensors about trustworthiness of s_j. In this chapter we only focus on direct trust.

The information collected based on entity's previous behavior [75] is one of the most important aspects of trust management solutions. Which kind of information about sensors' behavior is collected and analyzed varies from application to application. For example, a watchdog mechanism that monitors the behavior of neighboring nodes is proposed in [35] where node reputation and trust evolution are represented as a Bayesian formulation. Node capture attack [3], where nodes are removed from the network for a non-trivial amount of time, can be detected by their neighbors, since a node may appear and disappear from the network under normal conditions. One-shot probing is proposed in [87] to identify misbehaving nodes. Furthermore, the authors in [102] address the trust inference problem as a shortest path problem on a weighted directed graph, and utilized the theory of semirings for trust evaluation operations.

Investigating how to collect and evaluate sensor pre-behavior is beyond the scope in this study. Nodes may use the analyzing and scoring sensor trust approaches (e.g., [35, 3, 87, 102]) as a function of trust opinions. That is, in a time interval t, s_j's neighbors, $\{s_{j,i}\}_{i=1}^{nb_j}$, can generate trust opinions $T_i^{j,t}$ ($i \in \{1, ..., nb_j\}$) regarding s_j, by monitoring s_j's previous behavior.

6.3.2 Subjective Logic

In UWSNs, monitoring sensor behavior based on their previous communication record introduces uncertainty information since communication channels between sensors are unstable and noisy. To describe this uncertainty information, we adopt a subjective logic approach [45] in this chapter. In the subjective logic approach, the term *opinion* is defined to represent an opinion about trustworthiness. The definition of opinion is as follows.

Definition 6.1. *An* opinion *is a triplet, $T = \{B, D, U\}$, where $B, D, U \in [0, 1]$ and $B + D + U = 1$, and that B, D, and U correspond to* belief, disbelief, *and* uncertainty *respectively.*

A trust value can be thus defined based on these three parameters. For instance, a trust value associated with distrust could be represented as opinion $T_1 = \{0.0, 0.93, 0.07\}$, whereas a trust value associated with trust could be expressed as opinion $T_2 = \{0.88, 0.0, 0.12\}$.

Definition 6.2. *Let s_X, s_Y, and s_Z be three sensors. Then $T_Y^X = \{B_Y^X, D_Y^X, U_Y^X\}$*

and $T_Z^X = \{B_Z^X, D_Z^X, U_Z^X\}$ *denote the opinions of* s_Y *and* s_Z *about the trustworthiness of* s_X*. Then the combined consensus opinion is defined as follows:*

$$T_{Y,Z}^X = T_Y^X \oplus T_Z^X = \{B_{Y,Z}^X, D_{Y,Z}^X, U_{Y,Z}^X\}$$

where

$$B_{Y,Z}^X = (B_Y^X U_Z^X + B_Z^X U_Y^X)/(U_Y^X + U_Z^X - U_Y^X U_Z^X),$$

$$D_{Y,Z}^X = (D_Y^X U_Z^X + D_Z^X U_Y^X)/(U_Y^X + U_Z^X - U_Y^X U_Z^X),$$

and

$$U_{Y,Z}^X = (U_Y^X U_Z^X)/(U_Y^X + U_Z^X - U_Y^X U_Z^X).$$

The trust value expressed as subjective opinions instead of one simple trust level provides more flexible trust model of the real world. Hereby, according to Definition 6.2, the consensus of trust opinions generated by sensors $\{s_{j,i}\}_{i=1}^{nb_j}$ in time interval t about sensor s_j is

$$T_1^{j,t} \oplus \cdots \oplus T_i^{j,t} \oplus \cdots \oplus T_{nb_j}^{j,t} \quad = \quad T_{1,\cdots,i,\cdots,nb_j}^{j,t} \quad . \tag{6.1}$$

Definition 6.3. *Let* s_X *and* s_Y *be two sensors. Then* $\{T_Y^{X,t_1}, ..., T_Y^{X,t_n}\}$ *denote the opinion of* s_Y *about the trustworthiness of* s_X *on time intervals* $\{t_1, ...t_n\}$ *respectively, where* $T_Y^{X,t_n} = \{B_Y^{X,t_n}, D_Y^{X,t_n}, U_Y^{X,t_n}\}$*. Then* s_Y*'s opinion about the trustworthiness of* s_X *on* $t_1 \cup \cdots \cup t_n$ *is defined as*

$$T_Y^{X,t_1\cup\cdots\cup t_n} \quad = \quad \{B_Y^{X,t_1\cup\cdots\cup t_n}, D_Y^{X,t_1\cup\cdots\cup t_n}, U_Y^{X,t_1\cup\cdots\cup t_n}\} \tag{6.2}$$

where

$$B_Y^{X,t_1\cup\cdots\cup t_n} = \frac{1}{n}\big(B_Y^{X,t_1} + \cdots + B_Y^{X,t_n}\big),$$

$$D_Y^{X,t_1\cup\cdots\cup t_n} = \frac{1}{n}\big(D_Y^{X,t_1} + \cdots + D_Y^{X,t_n}\big),$$

and

$$U_Y^{X,t_1\cup\cdots\cup t_n} = \frac{1}{n}\big(U_Y^{X,t_1} + \cdots + U_Y^{X,t_n}\big).$$

According to Definitions 6.2 and 6.3, we define trustworthiness Υ^j in terms of sensor consensus to combine trust opinions generated by sensors $\{s_{j,i}\}_{i=1}^{nb_j}$ in time interval $\{t\}_{t=t_1}^{t_n}$ as

$$\Upsilon^j = T_{1,\cdots,i,\cdots,nb_j}^{j,t_1\cup\cdots\cup t_n} \ . \tag{6.3}$$

The Υ^j can be calculated with respect to sensor consensus or time as follows:

- With respect to *sensor consensus*:

$$\begin{aligned}\Upsilon^j &= T_{1,\cdots,i,\cdots,n_j}^{j,t_1\cup\cdots\cup t_n} \\ &= T_1^{j,t_1\cup\cdots\cup t_n} \oplus \cdots \oplus T_i^{j,t_1\cup\cdots\cup t_n} \oplus \cdots \oplus T_{n_j}^{j,t_1\cup\cdots\cup t_n}.\end{aligned}$$

- With respect to *time*:

$$\begin{aligned}\Upsilon^j &= T_{1,\cdots,i,\cdots,nb_j}^{j,t_1\cup\cdots\cup t_n} \\ &= \{B_{1,\cdots,i,\cdots,nb_j}^{j,t_1\cup\cdots\cup t_n}, D_{1,\cdots,i,\cdots,nb_j}^{j,t_1\cup\cdots\cup t_n}, U_{1,\cdots,i,\cdots,nb_j}^{j,t_1\cup\cdots\cup t_n}\},\end{aligned}$$

where

$$B_{1,\cdots,i,\cdots,nb_j}^{j,t_1\cup\cdots\cup t_n} = \frac{1}{n}\big(B_{1,\cdots,i,\cdots,nb_j}^{j,t_1} + \cdots + B_{1,\cdots,i,\cdots,nb_j}^{j,t_n}\big),$$

$$D_{1,\cdots,i,\cdots,nb_j}^{j,t_1\cup\cdots\cup t_n} = \frac{1}{n}\big(D_{1,\cdots,i,\cdots,nb_j}^{j,t_1} + \cdots + D_{1,\cdots,i,\cdots,nb_j}^{j,t_n}\big),$$

and

$$U_{1,\cdots,i,\cdots,nb_j}^{j,t_1\cup\cdots\cup t_n} = \frac{1}{n}\big(U_{1,\cdots,i,\cdots,nb_j}^{j,t_1} + \cdots + U_{1,\cdots,i,\cdots,nb_j}^{j,t_n}\big).$$

Remark: Definition 6.3 defines that each trust opinion has the same impact with respect to time. Actually, the desired scheme should be time-aware. For instance, the newer trust opinions may have higher impact on the trustworthiness, and past trust opinions should be also taken into account. However, we leave the time-aware trust calculation as our future work. We refer interested readers to [45] for more details on subjective logic and to [66] for an example of application of subjective logic in WSNs.

6.4 Efficient and Robust Storage of Trust Data

In traditional WSNs, a trusted third party, e.g., base station, is used to keep and calculate received trust opinions. The queries of sensors' trustworthiness are also sent to and answered by the base station. However, as there is no on-site base station in UWSNs, trust opinions of sensors need to be stored in sensors instead. Therefore, once sensor $s_{j,i}$ generates an opinion $T_i^{j,t}$ at time interval t, it either stores $T_i^{j,t}$ locally or sends $T_i^{j,t}$ to other nodes.

Next we consider three trust-related data storage schemes without involving the base station. First we consider two naive schemes and discuss their shortcomings. Then we propose an advanced scheme to improve the naive schemes.

6.4.1 Naive Scheme I (SI) - Trust Data Local Storage

The basic idea of the SI is to keep generated trust opinions locally, i.e., $s_{j,i}$ generates $T_i^{j,t}$ and then stores $T_i^{j,t}$ in its own memory. In other words, $s_{j,i}$ is not only one of s_j's trust producers but also one of s_j's trust managers.

The SI has three components: initialization, trust opinion local storage, and trust opinion querying and calculation, as explained below:

(1) Trust opinion local storage.

In every time interval, each sensor generates trust opinions about its neighbor nodes, combines it with previous trust opinions according to Eq. (6.2) and stores it locally. For instance, $s_{j,i}$ generates T_i^{j,t_1} , T_i^{j,t_2} and T_i^{j,t_3} at t_1, t_2 and t_3, respectively, and stores the combined trust opinion in its memory as

$$T_i^j \quad = \quad T_i^{j,t_0 \cup t_1 \cup t_2 \cup t_3} \quad .$$

(2) Trust opinion querying and calculation.

Consider the example in Fig. 6.1. Assume that sensor s_a intends to estimate the trustworthiness Υ^j of another sensor, s_j. It broadcasts a trust opinion request, $ASK(T^j)$, to ask sensors to collect opinions of other sensors about s_j. Here, we assume a suitable broadcast authentication protocol, e.g., multilevel μTESLA [51] for secure and reliable transmission of such broadcast values. If there is no direct relationship between two sensors (e.g., s_h and s_j), they keep most uncertain opinion about each other's trusts, i.e., $T_h^j = T_j^h = \{0, 0, 1\}$. Upon receiving $ASK(T^j)$, each sensor sends feedback messages, $ANS(T^j)$, to s_a if they have direct relationship with s_j. Otherwise they just drop $ASK(T^j)$. Next, s_a combines received sensors' opinions using consensus operator (Eq. (6.1)) to compute s_j's trustworthiness Υ^j, and keeps the results as Υ^j.

Proposition 6.4. *In the Naive Scheme I, the probability that at least one trust* *manager node stays as not compromised for up to time interval t is*

$$\begin{cases} Pr[survival]_t^j = 1, & k * t < nb_j \\ Pr[survival]_t^j = 0, & k * t \geq nb_j \quad , \end{cases} \tag{6.4}$$

where nb_j is the number of neighbor nodes and k is the compromising capability of $\mathcal{ADV}$ as defined in Subsection 6.2.2.

Proof. In SI, each sensor s_j has nb_j trust managers and nb_j trust producers in its transmission range. It is easy for $\mathcal{ADV}$ to find the trust managers in the transmission range of s_j. In each time interval $\mathcal{ADV}$ compromises k sensors (trust managers) in s_j's transmission range. Up to t-th time interval, $k * t$ sensors (trust managers) are compromised. Therefore, $k * t \geq nb_j$ means that all the trust managers are compromised, i.e., $Pr[survival]_t^j = 0$; otherwise $Pr[survival]_t^j = 1$. □

6.4.1.1 Communication Cost

Queries $ASK(T^j)$ are broadcast to all nodes at a cost of $O(N)$. Responses $ANS(T^j)$ are sent back to the trust consumer at a cost of $O(\sqrt{N})$ each. For t time intervals, the cost becomes $\mathcal{C}_j = O(tN) + O(tnb_j\sqrt{N}) = O(t(N + nb_j\sqrt{N}))$, where N is the number of sensors in the network.

6.4.1.2 Storage Cost

For a sensor, s_j, each neighbor in its transmission range generates one trust opinion per time interval. As the generated trust opinions are combined over time, the storage cost for each neighbor is $O(1)$. There are nb_j neighbors that need to store the trust opinions regarding s_j at a cost of $O(nb_j)$. That is, $\mathcal{S}_j = O(nb_j)$.

Discussion. According to Proposition 6.4, $\mathcal{ADV}$ can compromise all the trust managers in a short time ($\geq \frac{nb_j}{k}$). After that, both pre-compromised and post-compromised trust-related data can be modified by $\mathcal{ADV}$ once the trust managers are compromised. Therefore, we need to hide trust managers from $\mathcal{ADV}$. Next we propose Naive Scheme II that supports a distributed trust-related data storage.

6.4.2 Naive Scheme II (SII) - Distributed Trust Data Storage

In order to tackle the shortcomings of the SI, we should ensure that 1) a sensor s_j's trust producer and trust manager are not the same node; 2) $\mathcal{ADV}$ cannot easily find trust manager nodes; and 3) the desired scheme is resilient to node failures.

Keeping these desired properties in mind, a straightforward solution is that, for each node, specifying a designated node (which is not one of its direct neighbors) as its trust manager node to store its trust-related data. The procedures of the SII scheme is defined as follows:

(1) System initialization.

To provide trust-related data redundancy, in the beginning, each sensor, s_j, is associated with α randomly selected trust managers, $\{TM_j^r\}_{r=1}^{\alpha}$. Trust producers of s_j store the ID of $\{TM_j^r\}_{r=1}^{\alpha}$ since they need to send the generated trust opinions to $\{TM_j^r\}_{r=1}^{\alpha}$. Trust consumers store $\{s_j, \{TM_j^r\}_{r=1}^{\alpha}\}$ in their local memory so that trust consumers are able to retrieve s_j's trust-related data from $\{TM_j^r\}_{r=1}^{\alpha}$.

(2) Trust opinion distributed storage.

After generating trust opinions about s_j, the trust producers of s_j send them to $\{TM_j^r\}_{r=1}^{\alpha}$. Note that, in every time interval, TM_j^r receives nb_j trust opinions $\{T_i^{j,t}\}_{i=1}^{nb_j}$ from $s_{j,i} \in NB_j$ $(i \in [1, nb_j])$. After receiving $\{T_i^{j,t}\}_{i=1}^{nb_j}$, TM_j^r first removes outlier trust opinions as noise (to be discussed in Section 6.6). Then it calculates the trustworthiness $\Upsilon_r^{j,t}$ with the received trust opinions according to Eq. (6.3) where $\Upsilon_r^{j,t}$ is the trustworthiness of s_j stored in TM_j^r in time interval t.

(3) Trustworthiness query and calculation.

Trust consumers send $ASK(T^j)$ to $\{TM_j^r\}_{r=1}^{\alpha}$ to retrieve trustworthiness $\{\Upsilon_r^j\}_{r=1}^{\alpha}$ from s_j's trust manager nodes. Upon receiving α trustworthiness, trust consumers remove outliers using the similarity threshold functions defined in Section 6.6 and compute the expected value of the rest of Υ^j as s_j's trustworthiness.

Proposition 6.5. *In Naive Scheme II, the probability that at least one trust* *manager node is not compromised up to time interval t is*

$$Pr[survival]_t \;=\; 1-\Big(1-\prod_{t=1}^{t}\Big(1-\frac{k}{N-(t-1)k}\Big)\Big)^{\alpha}\;. \tag{6.5}$$

Proof. Let $E^r_{t'} = 1$ denote the event of r-th trust manager compromised by $\mathcal{ADV}$ at time interval t', and $E^r_{t'} = 0$ denote the event of r-th trust manager survival within the time interval t'. The probability of no trust manager nodes surviving up to t is

$$\begin{aligned}
&Pr[E^1_t = 1 \cap \cdots \cap E^r_t = 1 \cap \cdots \cap E^{\alpha}_t = 1]\\
&\quad = Pr[E^1_t = 1] * \cdots * Pr[E^r_t = 1] * \cdots * Pr[E^{\alpha}_t = 1]\\
&\quad = Pr[E^r_t = 1]^{\alpha}
\end{aligned}$$

Thus the probability of at least one trust manager node surviving up to t is $Pr[survival]_t = 1 - Pr[E^r_t = 1]^{\alpha}$.

The probability that r-th trust manager survives up to t is

$$\begin{aligned}
Pr[E^r_t = 0] \;&=\; \Big(1-\frac{k}{N}\Big)\Big(1-\frac{k}{N-k}\Big)\Big(1-\frac{k}{N-2k}\Big)\cdots\\
&\quad \cdots\Big(1-\frac{k}{N-(t-1)k}\Big)\\
&=\; \prod_{t=1}^{t}\Big(1-\frac{k}{N-(t-1)k}\Big)\;.
\end{aligned}$$

Thus, we have

$$\begin{aligned}
Pr[survival]_t \;&=\; 1 - Pr[E^r_t = 1]^{\alpha}\\
&=\; 1-(1-Pr[E^r_t = 0])^{\alpha}\\
&=\; 1-\Big(1-\prod_{t=1}^{t}\Big(1-\frac{k}{N-(t-1)k}\Big)\Big)^{\alpha}\;.
\end{aligned}$$

□

6.4.2.1 Communication Cost

Once a trust opinion is generated, it is sent and stored at the corresponding trust managers within the network. The communication cost to store the trust opinion is $O(\sqrt{N})$. Since nb_j trust producers need to send trust opinions to

α trust managers at each round, the total cost is $O(tnb_j\alpha\sqrt{N})$. Queries are sent to the designated node (trust manager), which also returns a response, both causing communication cost $O(\sqrt{N})$. That is, $\mathcal{C}_j = O(tnb_j\alpha\sqrt{N}) + O(t\alpha\sqrt{N}) = O(tnb_j\alpha\sqrt{N})$.

6.4.2.2 Storage Cost

For a sensor, s_j, all its trust producers (its neighbors) have to know that a designated sensor, TM_j, is the corresponding trust manager node, i.e., all sensors $s_{j,i} \in NB_j\,(i \in [1, nb_j])$, have to store the ID of TM_j at a cost of $O(\alpha nb_j)$. In addition, any node in the network could be s_j's trust consumer, consequently every node has to know which node is s_j's trust manager. That is, every node in the network has to store the ID of s_j and the ID of its corresponding trust manager TM_j, which causes $O(N)$ storage cost. Thus, $\mathcal{S}_j = O(\alpha(N + nb_j))$.

Discussion. In order to compare SII with SI, numerical results obtained in MATLAB using Eq. (6.4) and Eq. (6.5) are illustrated in Fig. 6.3.

Impact of the number of sensors, N: Fig. 6.3 (A) shows the impact of N on $Pr[survival]_t$, where compromising capability $k = 5$ and the upbound round of sensor surviving $t = 150$. We can see that SII has much better performance than SI in terms of $Pr[survival]_t$, and that $Pr[survival]_t$ of SII increases as either N or the number of trust manager nodes, α, increases. There are two reasons. First, increasing N decreases the probability of $\mathcal{ADV}\_Del$ finding the trust manager nodes. Second, increasing α increases trust data redundancy, forcing $\mathcal{ADV}\_Del$ to compromise more trust manager nodes to erase the data.

Impact of the compromising capability, k: Fig. 6.3 (B) shows the impact of k on $Pr[survival]_t$, where $N = 10000$ and the upbound time interval of sensor survival $t = 150$. We can see that $Pr[survival]_t$ decreases as k increases. The reason is that higher k means that $\mathcal{ADV}\_Del$ can compromise more sensor per time interval, increasing the probability of find the trust manager nodes. In contrast, increasing α increases the trust data redundancy, resulting in higher $Pr[survival]_t$. Finally, we observe that SII has much higher $Pr[survival]_t$ than SI.

Impact of the upbound time interval of sensor survival, t: Fig. 6.3 (C) shows the impact of t on $Pr[survival]_t$, where $N = 10000$ and $k = 5$. We observe that t has no impact on $Pr[survival]_t$ of SI, and that $Pr[survival]_t$ decreases as t increases. Again, SII has higher $Pr[survival]_t$ than that of SI.

As discussed above, these results clearly demonstrate that SII is much more resilient to $\mathcal{ADV}\_Del$ than SI. However, as discussed above, the storage cost of SII is proportional to $\alpha(N + nb_j)$, causing huge storage cost especially in large-scale UWSNs (see Figs. 6.7 (A) and (B)). This limitation motivates us to design scalable scheme that can reduce storage cost caused by distributed storage while keeping the same $Pr[survival]_t$ with SII.

6.4.3 Advanced Scheme (AS)

To reduce the large storage cost in SII, we propose AS. The idea of AS is motivated by GHT [72] where nodes can *put* and *get* data based on their data type, i.e., $Put(DataType, DataValue)$ and $Get(DataType)$, thereby supporting a hash-table-like interface. Since each sensor's ID is unique in the network, trust producers are able to *put* trust opinions to trust managers based on sensor ID, i.e., $Put(s_j, T_i^{j,t})$, and trust consumers are able to *get* trustworthiness from trust managers using the same sensor ID, i.e., $Get(s_j)$. In other words, trust opinions are pushed, and stored at the same trust manager node. Meanwhile it enables trust consumers to pull trustworthiness from the trust manager nodes consistently. Neither trust producers nor trust consumers need to store IDs of trust manager node, reducing storage cost significantly. Furthermore, the scheme should be robust against node failures. That is, the scheme should be resilient to $\mathcal{ADV}\_Del$. Thus, trust opinions are pushed to α ($\alpha > 1$) trust manager nodes, whereas trust consumers pull trustworthiness from α trust managers. To do so, we modify the original basic operations of GHT from

$$\begin{cases} Put(DataType, DataValue) \\ Get(DataType) \end{cases}$$

to

$$\begin{cases} Put(s_j, T_i^{j,t_l}, r) \\ Get(s_j, r) \end{cases} \quad \forall r \in [1, \alpha].$$

$Put(s_j, T_i^{j,t}, \alpha)$ is the function where trust producer $s_{j,i}$ is able to *put* its trust opinion $T_i^{j,t}$ regarding s_j to the r-th trust manager node of s_j, where α is the number of trust manager nodes specified by $\mathcal{MS}$ when the network is deployed. Whereas, $Get(s_j, r)$ is the function where trust consumers are able

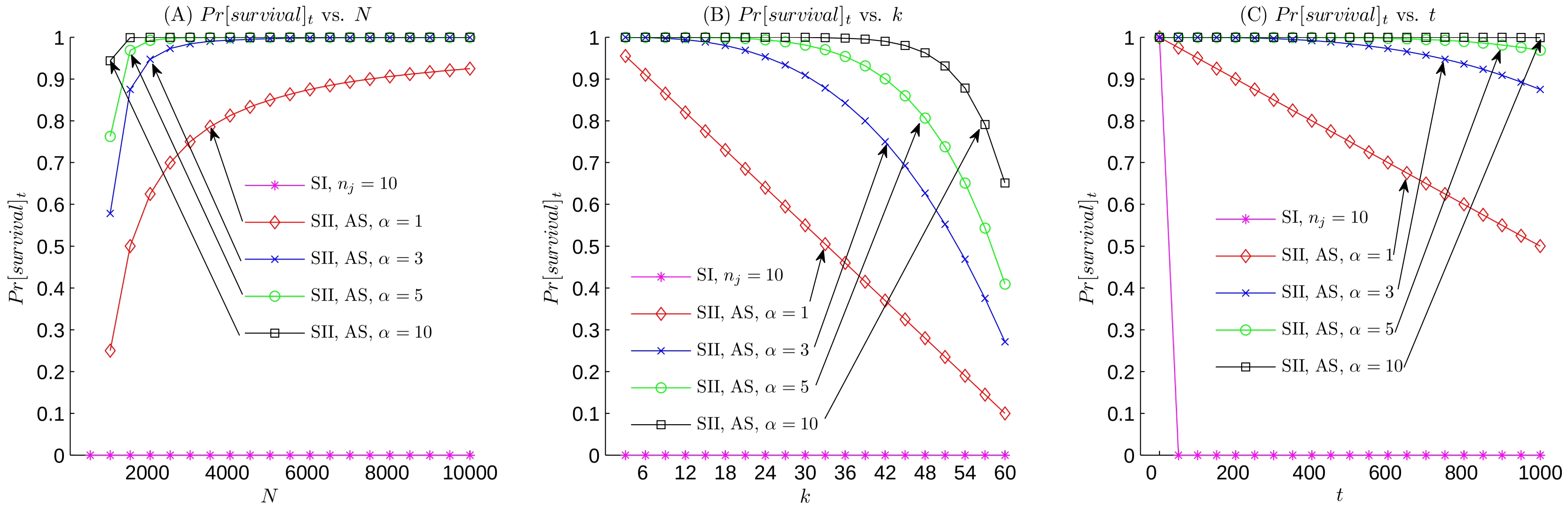


Figure 6.3: Comparison of SI, SII and AS using Eq. (6.4) and Eq. (6.5) with default setting: $N = 10000$, $k = 5$ and $t = 150$.

to *get* s_j's trustworthiness Υ^j from the r-th trust manager node of s_j. That is, each node has α trust manager nodes to store its trust opinions from its neighbors for data redundancy. In AS, trust opinions regarding a sensor, s_j, is hashed by the sensor's ID s_j to a geographical location. The node closest to the hashed geographical location is referred to as the trust manager node where data are sent to and retrieved from. As an example, as shown in Fig. 6.4, a sensor's ID s_j is hashed to $\alpha = 3$ random geographical locations in the sensor network by using a secure hash function $\mathcal{L}_j^r = h^r(s_j) = h(h^{r-1}(s_j))$ [3] $(\forall r \in \{1, ..., \alpha\})$; trust producers (e.g., $s_{j,i}$ and b_m) and trust consumers (e.g., s_a) can send trust opinions and trust query requests to $\mathcal{L}_j^r$ using GPSR [46]; and the closest node to the location $\mathcal{L}_j^r$, namely trust manager (see $\{TM_j^r\}_{r=1}^3$ in Fig. 6.4), can receive the trust opinions and trust query requests. The AS takes advantages of GHT and GPSR working as follows:

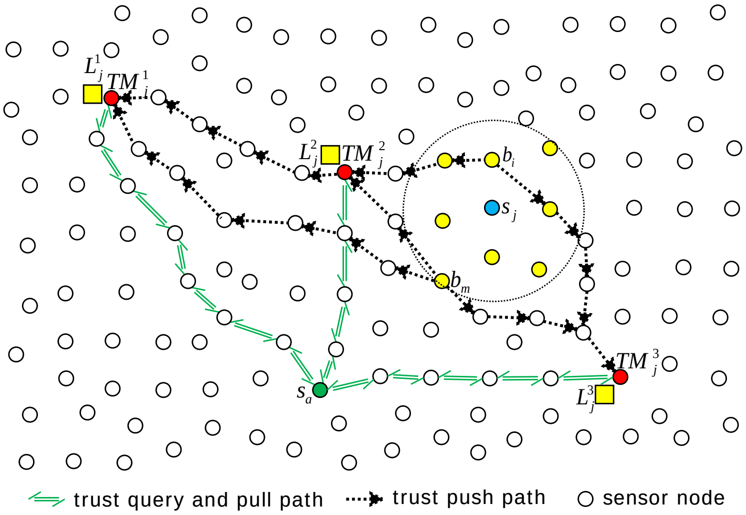


Figure 6.4: A simple example of GHT techniques on UWSNs with $\alpha = 3$.

(1) System initialization.

Each node is preloaded with a secure hash function, denoted as h, and the redundancy factor α specified by $\mathcal{MS}$ depending on application scenarios. All nodes know their own locations, and the locations of the nodes which are a single hop away from them.

(2) Trust opinion storage based on GHT.

In time interval t, after $T_i^{j,t}$ is generated, $s_{j,i}$ uses the function $Put(s_j, T_i^{j,t}, r)$ to *put* $T_i^{j,t}$ to α trust manager nodes. In other words, $s_{j,i}$ performs $h^r(s_j)$ to

[3] Note that $\mathcal{L}_j^r$ is not the location of s_j but the location closest to the r-th trust manager node of s_j.

obtain $\mathcal{L}_j^1, ..., \mathcal{L}_j^\alpha$, and then sends $T_i^{j,t}$ to locations $\mathcal{L}_j^1, ..., \mathcal{L}_j^\alpha$ using GPSR, respectively. The closest node to location $\mathcal{L}_j^r$, denoted as TM_j^r, finally receives the trust opinion $T_i^{j,t}$ and is called the r-th trust manager node of s_j.

(3) Trust opinion querying and calculation.

A trust consumer node, e.g., s_h, intends to know the trustworthiness of s_j. It uses the function $Get(s_j, r)$ $\forall r \in [1, \alpha]$ to *get* trustworthiness $\{\Upsilon_r^j\}_{r=1}^\alpha$ from s_j's α trust manager nodes. Similar as the *put* process, s_h performs $h^r(s_j)$ to obtain $\mathcal{L}_j^1, \cdots, \mathcal{L}_j^\alpha$, and sends $ASK(T_*^j)$ to locations $\mathcal{L}_j^1, \cdots, \mathcal{L}_j^\alpha$ using GPSR. The closest nodes to $\mathcal{L}_j^1, \cdots, \mathcal{L}_j^\alpha$, i.e., trust manager nodes $\{TM_j^r\}_{r=1}^\alpha$, finally receive $ASK(T_*^j)$ and then send $\{\Upsilon_r^j\}_{r=1}^\alpha$ to s_h.

Proposition 6.6. *Naive Scheme II and the Advanced Scheme have the same $Pr[survival]_t$. That is*

$$Pr[survival]_t \quad = \quad 1 - \left(1 - \prod_{t=1}^{t}\left(1 - \frac{k}{N-(t-1)k}\right)\right)^{\alpha} \quad .$$

Proof. The same as for Proposition 6.5. The numerical results are shown in Fig. 6.3. □

6.5 Efficiency and Robustness Evaluation

In this section we conduct a set of simulations in MATLAB to show that AS has the strongest performance among these three schemes in terms of efficiency and robustness. We consider a UWSN where 10000 nodes are randomly distributed in a 3000 × 3000 units area. The other parameters are set as follows. Each sensor has transmission range $\phi = 150$ units. $\mathcal{ADV}$ has compromising capability $k = 25$. The number of trust managers nodes $\alpha = 3$. The simulation results are averaged over 20 randomly deployed networks and are explained below.

Figs. 6.5 (A), (B), and (C) show the simulation results of t in terms of α, k, and ϕ. Fig. 6.5 (A) shows the impact of α on t, where $k = 25$ and $\phi = 150$. We can see that increasing α improves the performance of t of SII and AS, but has no effect on that of SI. This is because, in SII and AS, higher α forces $\mathcal{ADV}\_Del$ to compromise more trust manager nodes to erase trust data. In SI, however, all generated trust opinions are stored locally, which is not related to α. Fig. 6.5 (B) shows the impact of k on t, where $\alpha = 3$ and $\phi = 150$. We observe that increasing k decreases the performance of t of all three schemes. The reason is that increasing k enables $\mathcal{ADV}\_Del$ compromising more sensor per time interval, increasing the probability of finding trust manager nodes. Fig. 6.5 (C) shows that ϕ has no impact on SII and AS in terms of t but slightly increases the performance of t in SI. As explained above, Figs. 6.5 (A), (B), and (C) clearly demonstrate that SII and AS have better performance than SI does with respect to t.

Figs. 6.6 (A), (B), and (C) show the simulation results of communication cost $\mathcal{C}$ in terms of α, k, and ϕ, respectively. We observe that trust data distributed storage provides resilience to $\mathcal{ADV}\_Del$, and increases t, but causes larger communication cost. The reason is that trust producers in SII and AS send trust opinions to α trust manager nodes for adding trust data redundancy. In addition, increasing ϕ enables that more sensors can monitor their neighbor's behavior and then generate trust opinions. Thus more trust opinions have to sent to trust manager nodes. We can see that there is a tradeoff between trust data reliability and communication cost. Therefore, given a desired data reliability, we can select suitable values of α and ϕ for decreasing $\mathcal{C}$. For instance, as shown in Figs. 6.6 (B) and (C), the communication cost is acceptable if $\alpha \leq 3$ and $\phi \leq 120$.

Fig. 6.7 (A) shows the impact of α on $\mathcal{S}$, where $\phi = 150$. We can see that SII has very large storage cost $\mathcal{S}$, and that $\mathcal{S}$ obviously increases as α increases. This coincides with the analysis in Subsubsection 6.4.2.2. Whereas, AS and SI have very low storage cost, and α almost has no impact on $\mathcal{S}$. Fig. 6.7 (B) shows the impact of ϕ on $\mathcal{S}$, where $\alpha = 3$. We observe that ϕ has no impact on AS in terms of $\mathcal{S}$ but slightly increases $\mathcal{S}$ of SI and SII, and that AS and SI have very low storage cost. Finally, as discussed above, Figs. 6.7 (A) and (B) both show that AS has the lowest storage cost among them.

Discussion. We have the following observations.

(1) Robustness. As there are α trust mangers in the network, in each time interval, generated trust opinions $\{T_i^{j,t}\}_{i=1}^{nb_j}$ are routed to and stored in these α trust managers. Therefore, the trustworthiness is available to be retrieved even up to $\alpha - 1$ trust managers lose function totally. That is, the SII and AS are resilient to $\mathcal{ADV}\_Del$, as shown in Figs. 6.5 (A)(B)(C).

(2) Efficiency. SII and AS have the same communication cost. In addition, AS requires much less storage cost than SII does. Unlike SII, given a sensor s_j, trust producers do not need to store the ID of corresponding trust managers

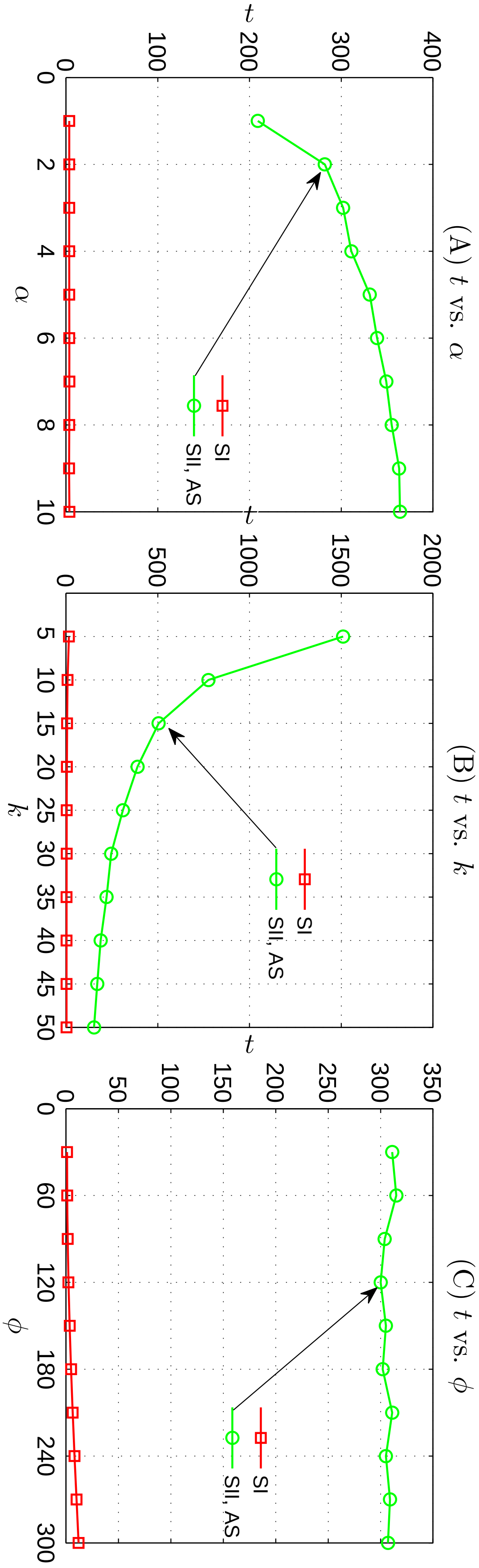
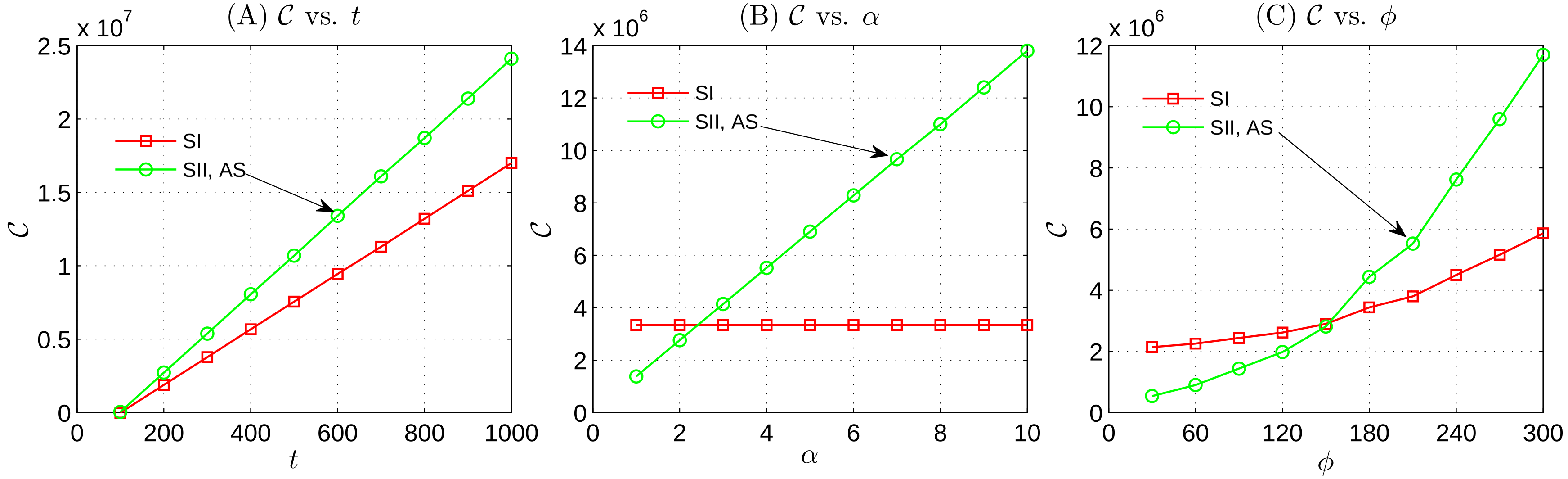


Figure 6.5: Simulation results: $\alpha/k/\phi$ vs. t.



(A) $\mathcal{C}$ vs. t

(B) $\mathcal{C}$ vs. α

(C) $\mathcal{C}$ vs. ϕ

Figure 6.6: Simulation results: $t/\alpha/\phi$ vs. $\mathcal{C}$.

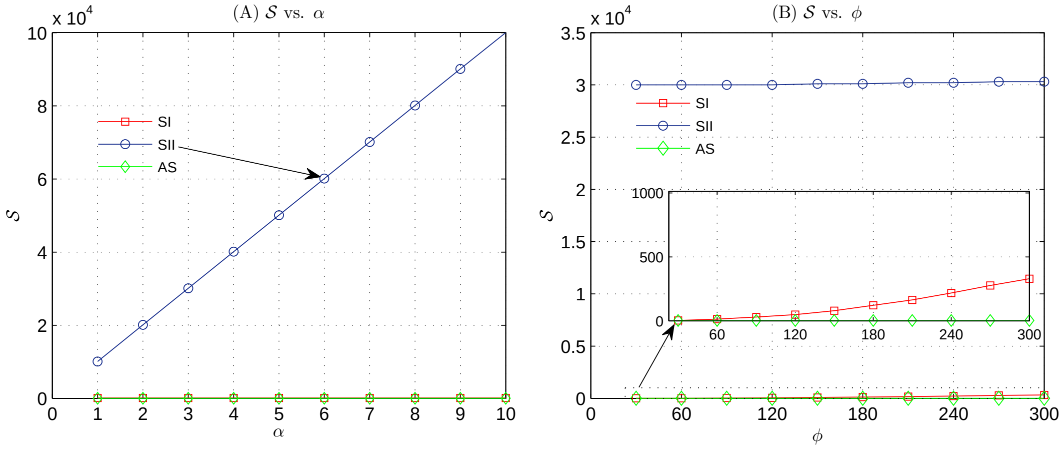


Figure 6.7: Simulation results: α/ϕ vs. $\mathcal{S}$.

of s_j due to the *put* and *get* function of GHT. For α trust managers, the total storage cost is $O(\alpha(N + nb_j))$ for SII and $O(\alpha)$ for AS. Thus, AS is more efficient than SII.

(3) Scalability. As discussed above, the storage cost of AS has no relation with the number of the sensors, N. That is, increasing N does not increase storage cost of AS. Moreover, Subsubsection 6.4.2.1 demonstrates that the communication cost of AS is proportional to $\sqrt{N}$. For example, when the number of sensors N increases 10 times from 1000 to 10000, the communication cost increases merely 3 times. Therefore AS is scalable.

(4) Consistency. For a given sensor, s_j, all generated trust opinions $\{T_i^{j,t}\}_{i=1}^{nb_j}$ are routed to $\{\mathcal{L}_j^r\}_{r=1}^{\alpha}$, respectively. The nodes closest to $\{\mathcal{L}_j^r\}_{r=1}^{\alpha}$ receive $\{T_i^{j,t}\}_{i=1}^{nb_j}$ and store them in their local memories, where $\mathcal{L}_j^r = h^r(s_j)$. As sensor ID is unique in the network, for a given sensor s_j, the generated hash values $\{\mathcal{L}_j^r\}_{r=1}^{\alpha}$ are also unique in the network due to the one-way property of the hash function. As a consequence, all the trust producers and trust consumers are able to find the exact trust managers to store and query trust-related data.

6.6 Trustworthiness Generation

Through the simulations and discussions in the previous section we have demonstrated that AS significantly reduces storage cost caused by distributed data storage and provides resilience to $\mathcal{ADV}\_Del$. In this section we continue to investigate the impacts of the performance of the proposed schemes regarding trustworthiness pollution attacks (i.e., $\mathcal{ADV}\_Noise$, $\mathcal{ADV}\_Homo$ and

$\mathcal{ADV}\_Hbd$) defined in Subsection 6.2.2.

Initially, for each sensor s_j, trust opinion $T_i^{j,0}$ could be set by $\mathcal{MS}$ based on such information as the physical protection, the location, or the role of the nodes. For example, a node, s_j, buried under the ground has higher $\{T_i^{j,0}\}_{i=1}^{nb_j}$ than the exposed ones. After the trust opinions generated, a trust producer $s_{j,i}$ is able to update T_i^j using time conjunction according to Eq. (6.2), and trust consumers can calculate the trustworthiness Υ^j using sensor consensus according to Eq. (6.1).

Next we conduct a set of simulations to study the trust resilience of the proposed schemes with respect to trustworthiness generation over pollution attacks. The uncompromised sensor nodes generate correct trust opinions $cT^j = \{cB^j, cD^j, cU^j\}$ whereas the compromised sensors generate false trust opinions $fT^j = \{fB^j, fD^j, fU^j\}$, where cB and fB denote *correct believe* and *false believe* respectively, so do cD, cU, fD and fU in our simulations. These trust opinion values are followed by normal distribution, i.e., $B \sim \mathcal{N}(E(B), \sigma^2)$, where $E(B)$ is the expected value of B and σ is the standard deviation of B. In order to compare the impact of false trust opinions fT, in the simulations, fT are generated after 20th time interval so that we can observe if there is any difference before 20th time interval and after 20th time interval on the trustworthiness of a node.

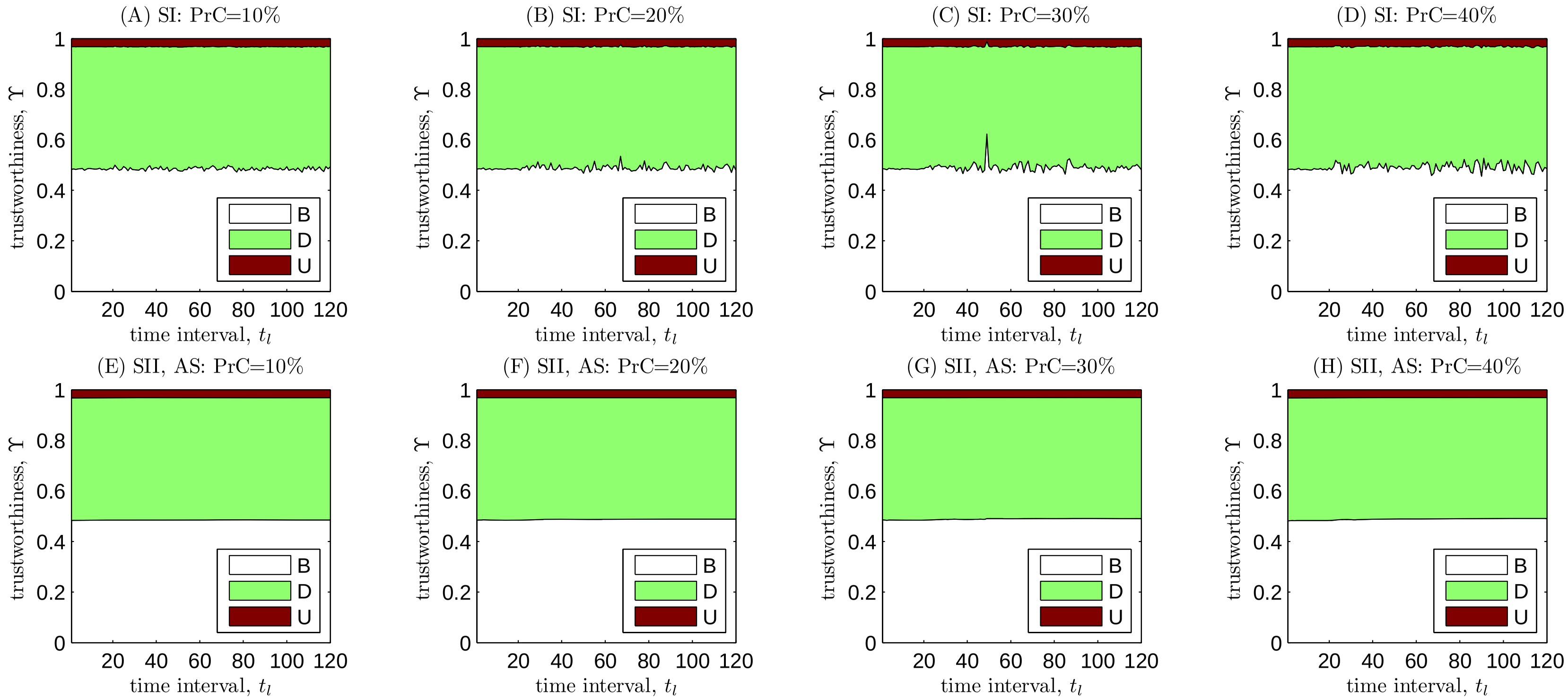
Let $d(B_i^j, E(B_m^j)) = \sqrt{(B_i^j - E(B_m^j))^2}$ denote the Euclidean distance between B_i^j and its expected value $E(B_m^j)$ where $j \in \{1, ..., N\}$ and $i, m \in \{1, ..., nb_j\}$, and so as $d(D_i^j, E(D_m^j)$ and $d(U_i^j, E(U_m^j))$.

6.6.1 Trust Consensus only Approach (TC-ONLY)

In this subsection we look at TC-ONLY, which we used in Section 6.4, as our baseline. This approach generates trustworthiness based on trust consensus only (Definition 6.2 and Definition 6.3). Through simulation results, we first show that it is resilient to $\mathcal{ADV}\_Noise$, and then demonstrate its shortcomings with respect to $\mathcal{ADV}\_Homo$ and $\mathcal{ADV}\_Hbd$.

6.6.1.1 Trust Resilience against $\mathcal{ADV}\_Noise$

To evaluate the performance of the proposed scheme with respect to environment effect ($\mathcal{ADV}\_Noise$), i.e., $d(a_i^j, E(a_m^j)) \approx 0$ ($a \in \{B, D, U\}$), we set the correct trust opinions as $cT = \{0.3, 0.3, 0.4\}$ and $\sigma_c = 0.01$, where


(A) SI: PrC=10%

(B) SI: PrC=20%

(C) SI: PrC=30%

(D) SI: PrC=40%

(E) SII, AS: PrC=10%

(F) SII, AS: PrC=20%

(G) SII, AS: PrC=30%

(H) SII, AS: PrC=40%

Figure 6.8: Consensus is enough, $cT = \{0.3, 0.3, 0.4\}$, $\sigma_c = 0.01$, $fT = \{0.3, 0.3, 0.4\}$, $\sigma_f = 0.1$.

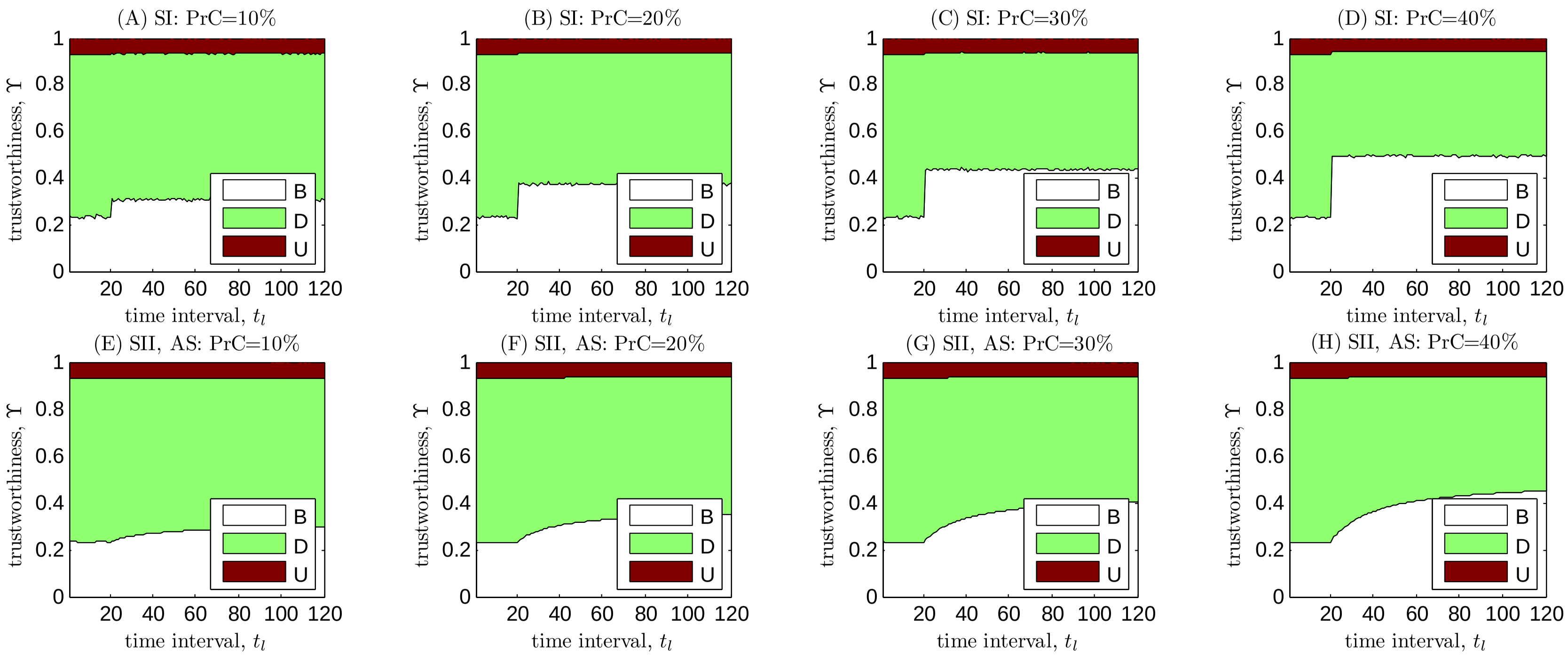


Figure 6.9: An example of $\mathcal{ADV}$ tries to increase Υ^j, $cT = \{0.1, 0.3, 0.6\}$, $fT = \{0.4, 0.1, 0.5\}$, $\sigma_c = \sigma_f = 0.01$.

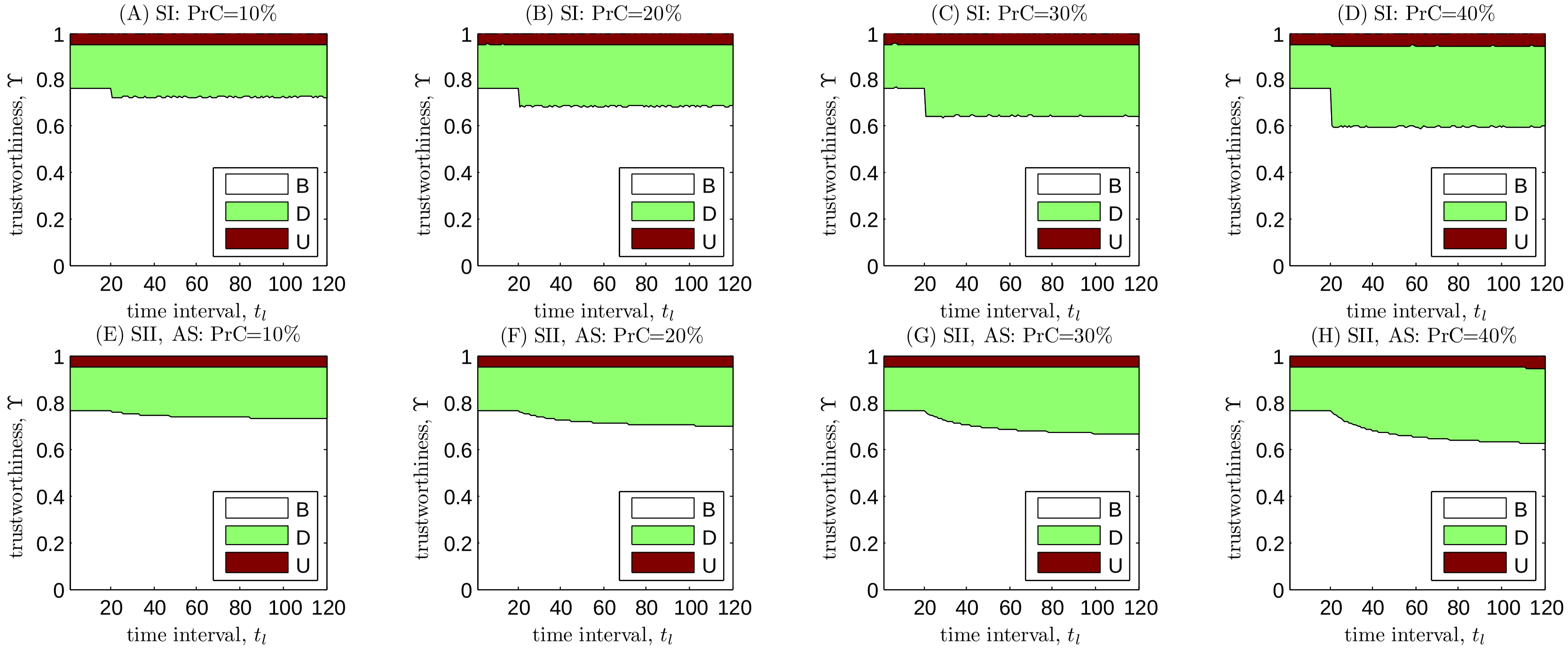


Figure 6.10: An example of $\mathcal{ADV}$ tries to decrease Υ^j, $cT = \{0.4, 0.1, 0.5\}$, $fT = \{0.1, 0.3, 0.6\}$, $\sigma_c = \sigma_f = 0.01$.

$cB, cD \sim \mathcal{N}(0.3, 0.001)$, and $cU = 1 - cB - cD$. In order to monitor environment effect, we set a certain percentage PrC of sensors generate trust opinions with larger $\sigma_f = 0.1$.

Fig. 6.8 shows the simulation results of SI, SII and AS when different values of PrC (from 10% to 40%) are specified. These figures are plotted by MATLAB *area* function displaying three elements B, D and U in white, green (light shadow), and red (black) respectively. The first row of Fig. 6.8 is the simulation results of SI, we can see that after 20th time interval the obtained trustworthiness Υ starts to become unstable. In addition, increasing the percentage PrC of anomalous sensors makes Υ more unstable. The second row of Fig. 6.8 is the simulation results of SII and AS, we observe that Υ is very smooth no matter how PrC is selected, and that the anomalous trust opinions have almost no influence on Υ (slightly increase Υ when $PrC = 40\%$, see Fig. 6.8(H)) when SII and AS are employed. This is because trust consensus makes Υ smooth. Comparing the first row of Fig. 6.8 and the second row of Fig. 6.8, it is easy to see that SII and AS are more resilient against $\mathcal{ADV}\_Noise$ than SI. Therefore, trust consensus makes good performance against $\mathcal{ADV}\_Noise$, i.e., $d(a_i^j, E(a_m^j)) \approx 0$ ($a \in \{B, D, U\}$).

6.6.1.2 Trust Resilience against $\mathcal{ADV}\_Homo$

Since $\mathcal{ADV}\_Homo$ tries to either increase Υ or decrease Υ monotonously, we conduct two sets of simulations. In the first set simulation, $\mathcal{ADV}\_Homo$ is assumed to generate false trust opinions fT to increase Υ. In contrast, $\mathcal{ADV}\_Homo$ is assumed to decrease Υ in the second set simulation.

Simulation one. To increase Υ, the most effective way for $\mathcal{ADV}\_Homo$ is to increase B and to decrease D simultaneously. That is, generating opinions fT which satisfy $E(cB) < E(fB)$ and $E(cD) > E(fD)$. We select a special case when $cT = \{0.1, 0.3, 0.6\}$, $fT = \{0.4, 0.1, 0.5\}$ and $\sigma_c = \sigma_f = 0.01$. The simulation results are as shown in Fig. 6.9. Again, the simulation results of SI are plotted in the first row, and that of SII and AS are plotted in the second row. We observe that results of SII and AS are smoother than that of SI. After 20th time interval, Υ increases as $\mathcal{ADV}\_Homo$ expected, meaning that the trust consensus cannot tolerate $\mathcal{ADV}\_Homo$. In addition, when PrC increases, Υ starts to increase. The reason is that more sensors generate false trust opinions, increasing the impact of false trust opinion fT in trustworthiness Υ.

Simulation two. In contrast, to decrease Υ, the most effective way for $\mathcal{ADV}\_Homo$ is to decrease B and to increase D simultaneously. That is, $E(cB) > E(fB)$ and $E(cD) < E(fD)$. We set $cT = \{0.4, 0.1, 0.5\}$, $fT = \{0.1, 0.3, 0.6\}$, $\sigma_c = \sigma_f = 0.01$ in the set of simulations. As shown in Fig. 6.10, the results also illustrate that the trust consensus cannot tolerate $\mathcal{ADV}\_Homo$. After 20th time interval, Υ starts to decrease immediately. The same as simulation one, increasing PrC gives more influence on Υ, and SII and AS have better performance than SI.

Through the simulation results conducted above, we conclude that TC-ONLY cannot tolerate $\mathcal{ADV}\_Homo$.

6.6.1.3 Trust Resilience against $\mathcal{ADV}\_Hbd$

As $\mathcal{ADV}\_Hbd$ is free to launch both increasing Υ attack and decreasing Υ attack, we claim a scheme can tolerate $\mathcal{ADV}\_Hbd$ if and only if the scheme is able to get good results against both increasing Υ attack and decreasing Υ attack. However, as shown in Fig. 6.9 and Fig. 6.10, the trust consensus can get good results in neither increasing Υ attack nor decreasing Υ attack. Therefore, the trust consensus cannot tolerate $\mathcal{ADV}\_Hbd$.

Discussion. From the simulation results shown above, it is easy to see that TC-ONLY is not enough for trustworthiness calculation. It can only tolerate $\mathcal{ADV}\_Noise$ caused by environment effect. The reason is that using trust consensus for trust calculation decreases uncertainty U and makes Υ stable. However, it does not show good performance against $\mathcal{ADV}\_Homo$ and $\mathcal{ADV}\_Hbd$. The reason is that both correct trust opinions cT and false trust opinions fT are taken into trustworthiness calculation as input, resulting in the polluted trustworthiness Υ. To solve this problem, a straightforward solution is to reduce the effect of fT as much as possible. Thus we propose the next scheme that is able to remove false trust opinions.

6.6.2 Trust Consensus with One Parameter Similarity Threshold Function (ONE-PARA)

As compromised trust producers may send false trust opinions to trust managers to *pollute* trustworthiness, we use ONE-PARA to remove outliers. A one parameter similarity threshold function is defined as

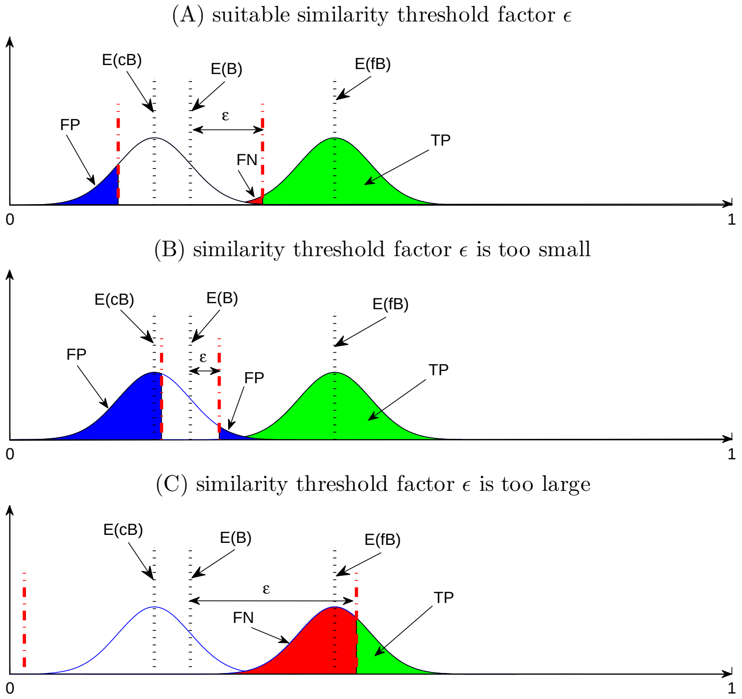
$$ST(T_i^j) = \frac{\sqrt{(B_i^j - E(B_m^j))^2}}{B_i^j E(B_m^j)} \ . \tag{6.6}$$

Any T_i^j is considered as an outlier if $ST(T_i^j) > \epsilon$ where ϵ is a similarity threshold factor (e.g., $\epsilon = 0.1$).

The similarity threshold function is expected to take out false trust opinions as much as possible. It is also desirable that false positives (trust opinions are considered as false trust opinions even though they are correct ones), and that false negatives (trust opinions are considered as correct trust opinions even though they are false ones) are as fewer as possible. In other words, we prefer to increase true positive as much as possible while keeping very low false positives and false negatives. However, as the idea of the threshold function is based on how *far* B_i^j is away from its expected value $E(B_m^j)$, and $E(B_m^j)$ is the average value of both cB and fB, the selection of ϵ is a dilemma. As shown in Fig. 6.11, decreasing ϵ increases true positives, however, also it increases false positives. Fig. 6.11 (A) shows that most part of fB and a small part of cB are considered as outliers, and a small part of false trust opinions are considered as correct trust opinions (false negative), if suitable similarity threshold factor ϵ is specified. When ϵ is selected too small, as shown in Fig. 6.11 (B), all the false trust opinions are considered as outliers, however, more than half of the correct trust opinions are also considered as outliers. In contrast, when ϵ is specified too large, as shown in Fig. 6.11, more than half of the false trust opinions are considered as correct trust opinions. In addition, the more false trust opinions are considered as correct trust opinions the more $E(B_m^j)$ closes to $E(fB)$. Fig. 6.12 shows the receiver operating characteristic curve with $cT = \{0.1, 0.3, 0.6\}$, $fT = \{0.4, 0.1, 0.5\}$, $\sigma_c = \sigma_f = 0.01$ where similarity threshold factor ϵ is specified as 0.2, 0.4, 0.6 and 0.8 respectively. We observe that true positives increases as false positives increases, and that the receiver operating characteristic curves are the same when $\epsilon = 0.6$ and 0.8. That is the reason why Figs. 6.13 (D)(I) and (E)(J) have the same performance.

6.6.2.1 Trust Resilience against $\mathcal{ADV}\_Noise$

Since ONE-PARA is an improved version of TC-ONLY, it works well against $\mathcal{ADV}\_Noise$ and can further filter outliers.


(A) suitable similarity threshold factor ϵ

(B) similarity threshold factor ϵ is too small

(C) similarity threshold factor ϵ is too large

Figure 6.11: An example of similarity threshold factor selection in terms of false positive (FP), ture positive (TP) and false negative (FN).

6.6.2.2 Trust Resilience against $\mathcal{ADV}\_Homo$

As demonstrate above, TC-ONLY cannot tolerate $\mathcal{ADV}\_Homo$. To compared with the simulations results of TC-ONLY, we conduct two more sets of other simulations using the same parameters as the last two sets of simulations of TC-ONLY when the percentage of compromised sensors are fixed as $PrC = 20\%$.

The simulation results are as shown in Fig. 6.13 and Fig. 6.14. The first row of them are the simulation results for SI, and the second row of those figures are simulation results for SII and AS. Again, we can see that the generated Υ in SII and AS is more stable than Υ in SI. The first column of Fig. 6.13 and Fig. 6.14 is the simulation results when the threshold function is not applied (TC-ONLY). The rest of columns of those figures are the simulation results when the threshold function (ONE-PARA) is applied where the similarity threshold factor ϵ is various from 0.2 to 0.8. It is easy to conclude that TC-ONLY (see the first column of Fig. 6.13 and Fig. 6.14) cannot tolerate $\mathcal{ADV}\_Homo$ and $\mathcal{ADV}\_Hbd$ as expected. Let us take a closer look at the simulation results of SII and AS (the second column to the fifth column of Fig. 6.13 and Fig. 6.14).

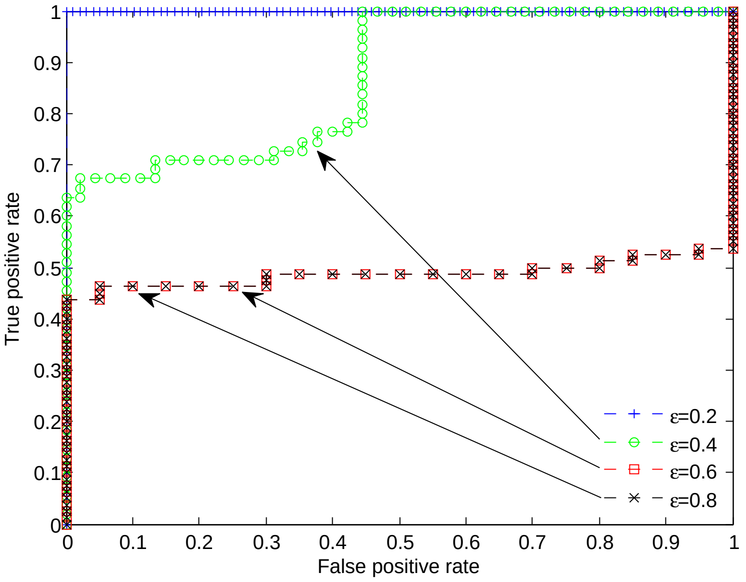


Figure 6.12: Receiver operating characteristic curve with $cT = \{0.1, 0.3, 0.6\}$, $fT = \{0.4, 0.1, 0.5\}$, $\sigma_c = \sigma_f = 0.01$ where similarity threshold factor ϵ are specified as 0.2, 0.4, 0.6 and 0.8 respectively.

- Suitable ϵ. We observe that ONE-PARA works well (see Figs. 6.13 (I)(J) and Figs. 6.14 (H)(I)) when a suitable ϵ is selected.
- ϵ is too small. We also illustrate that uncertainty U increases while believe B and disbelieve D decrease in the second column of Fig. 6.13 and Fig. 6.14 which is coincident with the discussion above.
- ϵ is too large. As shown in Fig. 6.11 (C), false negative increases as ϵ increases. That is, more false trust opinions are considered as correct trust opinions, resulting in that disbelieve D increases and believe B decreases, as shown in Figs. 6.14 (E)(J).

Discussion. As shown in Fig. 6.13 and Fig. 6.14, ONE-PARA works well in some special cases (see Table 6.1). However, when $d(B_i^j, E(B_m^j)) \approx 0$ and $d(D_i^j, E(D_m^j))$ is high, ONE-PARA cannot distinguish the difference of disbelieve D, such as correct trust opinion $cT = \{0.1, 0.1, 0.8\}$ and false trust opinion $fT = \{0.1, 0.8, 0.1\}$, or $cT = \{0.1, 0.8, 0.1\}$ and $fT = \{0.1, 0.1, 0.8\}$. As shown in Fig. 6.15, when $cB = fB = 0.1$, $cD = 0.1$ and $fD = 0.8$, the differences of disbelieve D cannot be detected by ONE-PARA, but it causes very large consequence. We observe that there is no difference between the simulation results with ONE-PARA and without it. $\mathcal{ADV}$'s attack pollutes the

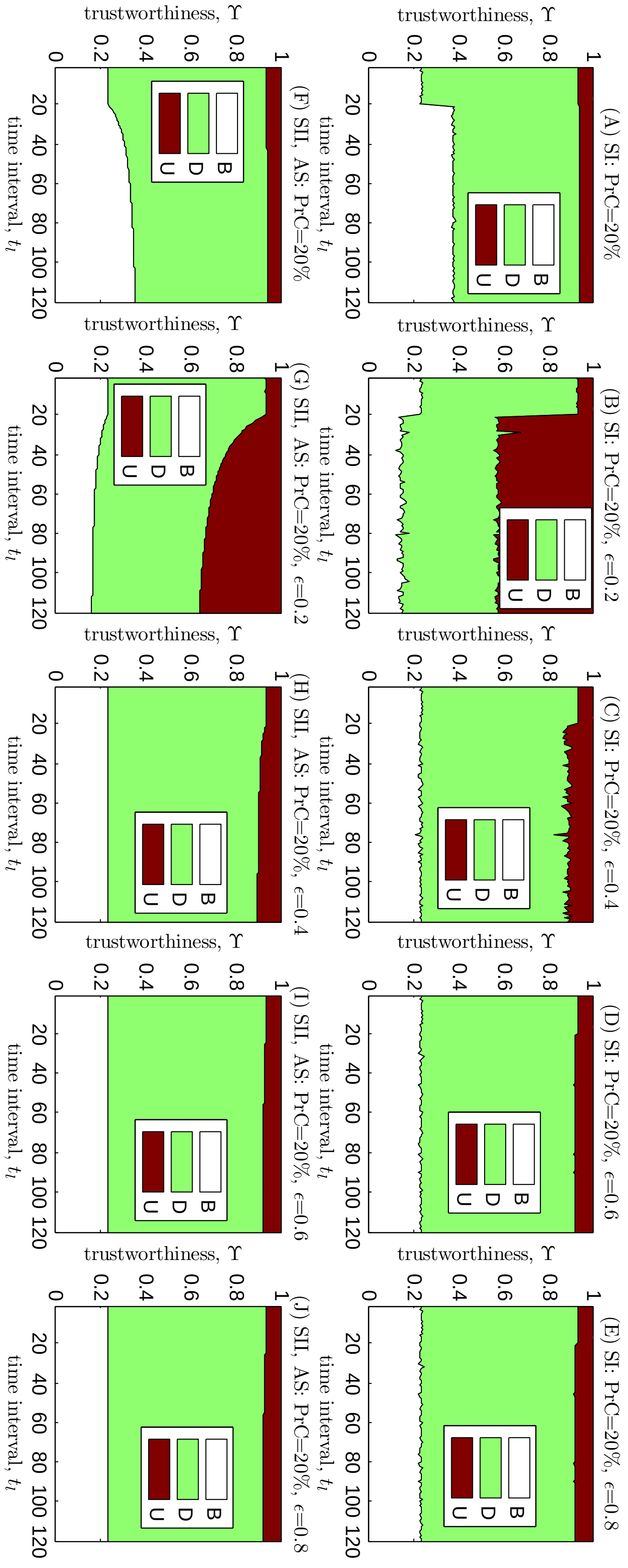


Figure 6.13: An example of $\mathcal{ADV}$ tries to increase Υ^j, $cT = \{0.1, 0.3, 0.6\}$, $fT = \{0.4, 0.1, 0.5\}$, $\sigma_c = \sigma_f = 0.01$.

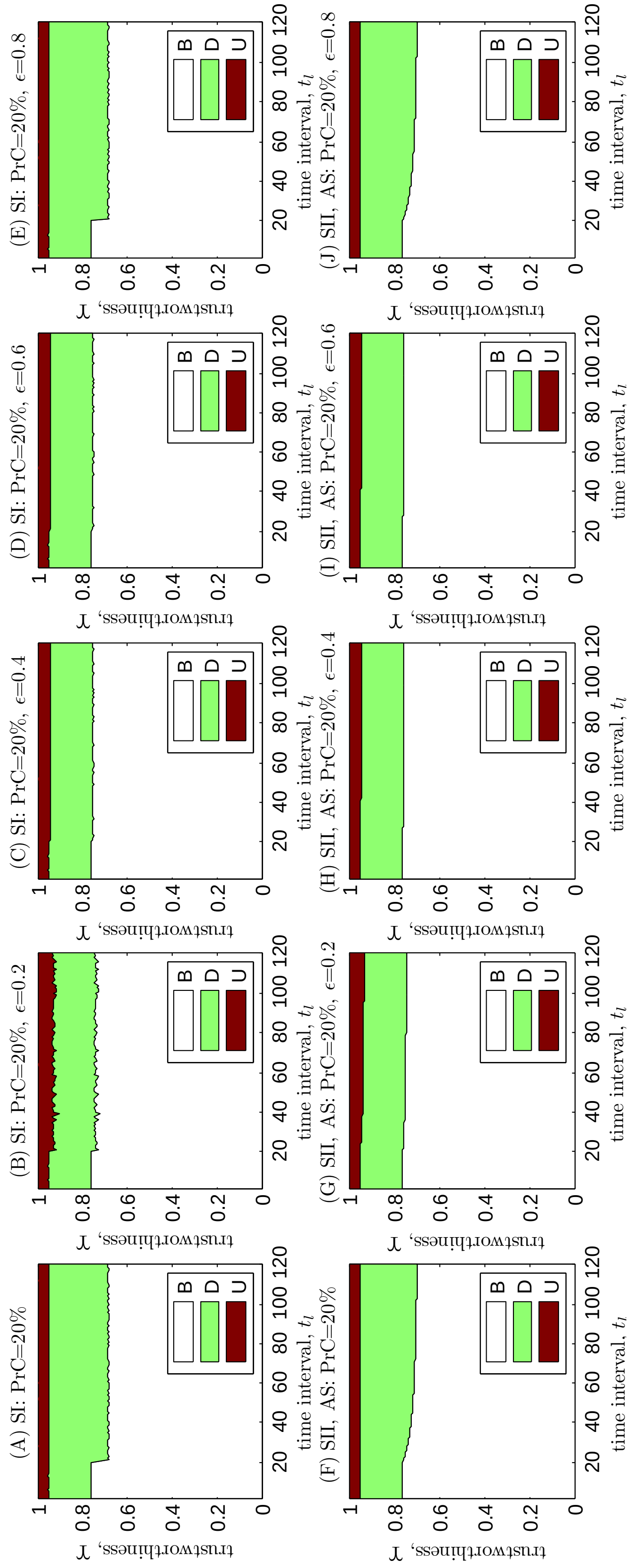


Figure 6.14: An example of $\mathcal{ADV}$ tries to decrease Υ^j, $cT = \{0.4, 0.1, 0.5\}$, $fT = \{0.1, 0.3, 0.6\}$, $\sigma_c = \sigma_f = 0.01$.

trustworthiness severely for all schemes. As shown in Fig. 6.16, we also observe that $\mathcal{ADV}$'s attack is tolerated by the consensus when $d(B_i^j, E(B_m^j)) \approx 0$ and $E(cB) > E(fB)$. Furthermore, ONE-PARA can only specify one parameter among the three elements B, D and U so that it cannot tolerate $\mathcal{ADV}\_Hbd$. The reason is that the most efficient way for $\mathcal{ADV}$ to increase Υ is to increase B and to decrease D simultaneously, whereas to decrease Υ is to decrease B and to increase D simultaneously. The threshold functions based on one parameter cannot control two parameters. To solve the pollution caused by $\mathcal{ADV}\_Hbd$, and by the special case when $d(B_i^j, E(B_m^j)) \approx 0$ and $E(cB) < E(fB)$, we propose the next scheme.

6.6.3 Trust Consensus with Three Parameter Similarity Threshold Function (T-PARA)

As ONE-PARA cannot identify the difference between $cT = \{0.1, 0.1, 0.8\}$ and false trust opinion $fT = \{0.1, 0.8, 0.1\}$. We propose a three parameters similarity threshold function as follows,

$$ST(T_i^j) = \frac{\sqrt{(B_i^j - E(B_m^j))^2 + (D_i^j - E(D_m^j))^2 + (U_i^j - E(U_m^j))^2}}{B_i^j E(B_m^j) + D_i^j E(D_m^j) + U_i^j E(U_m^j)} \; . \qquad (6.7)$$

6.6.3.1 Trust Resilience against $\mathcal{ADV}\_Noise$

Again, since T-PARA is an improved version of TC-ONLY, the simulation results are better than the ones obtained from TC-ONLY, even though the simulation results with respect to $\mathcal{ADV}\_Noise$ are not shown here.

6.6.3.2 Trust Resilience against $\mathcal{ADV}\_Homo$ and $\mathcal{ADV}\_Hbd$

We use the same simulation parameters as ONE-PARA simulations. Fig. 6.17 and Fig. 6.18 show the simulation results. Again, the simulation results based on TC-ONLY are plotted in the first column of them. We further observe that the simulation results of T-PARA are much better than the simulation results of TC-ONLY when suitable similarity threshold factor ϵ is specified (see Figs. 6.17 (H)(I)(J) and Figs. 6.18 (H)(I)(J)). Moreover, we observe that T-PARA works well when $\epsilon = 0.4, 0.6$ and 0.8 (see Figs. 6.17 (H)(I)(J)), while the simulation results is very poor in terms of ONE-PARA (see Fig. 6.15).

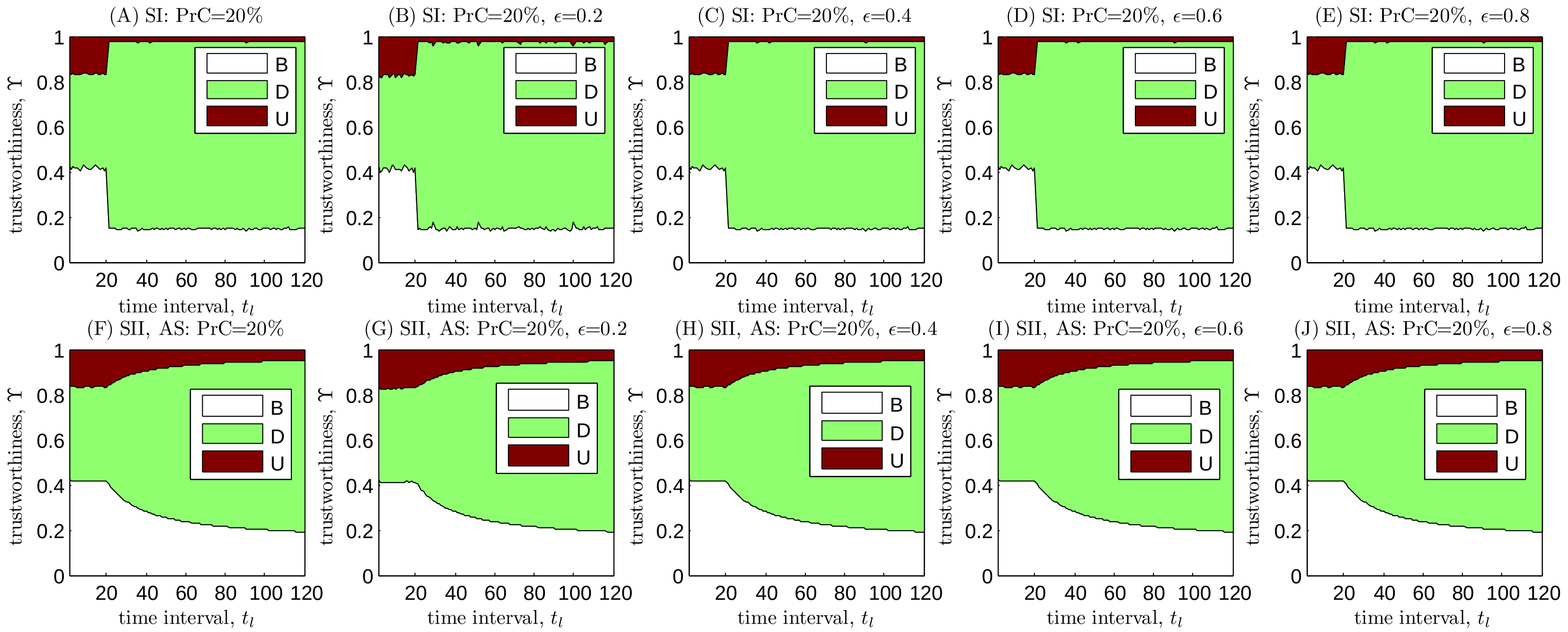


Figure 6.15: An example of one parameter threshold function does not work, when $cT = \{0.1, 0.1, 0.8\}$, $fT = \{0.1, 0.8, 0.1\}$, $\sigma_c = \sigma_f = 0.01$.

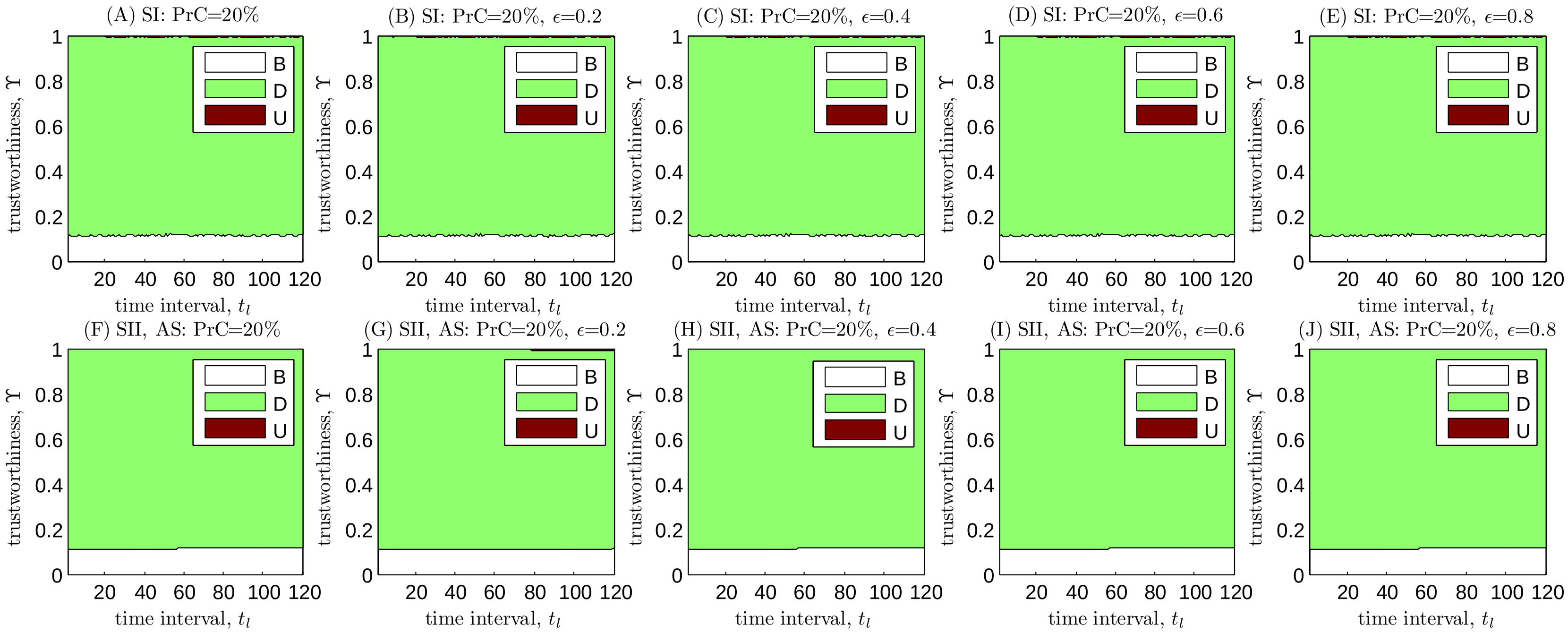


Figure 6.16: An example of one parameter threshold function works even it cannot identity the difference between $cT = \{0.1, 0.8, 0.1\}$ and $fT = \{0.1, 0.1, 0.8\}$, $\sigma_c = \sigma_f = 0.01$.

Figs. 6.17 (B)(G) and Figs. 6.18 (B)(G) depict the impact of ϵ when it is specified too small.

Since T-PARA works well (see Figs. 6.17 (H)(I)(J) and Figs. 6.18 (H)(I)(J)) against both $\mathcal{ADV}\_Homo$ increasing Υ attack and $\mathcal{ADV}\_Homo$ decreasing Υ attack. Therefore, it is resilient to $\mathcal{ADV}\_Hbd$.

6.6.4 Three Parameters with Weighted Factors (T-PARA-WF)

Compared with Eq. (6.7) having equal weight to all three parameters, we define a more flexible threshold function to prevent $\mathcal{ADV}$ from pollution attacks. An improved version of Eq. (6.7) is proposed as follows,

$$ST(T_i^j) = \frac{\sqrt{x^2(B_i^j - E(B_m^j))^2 + y^2(D_i^j - E(D_m^j))^2 + z^2(U_i^j - E(U_m^j))^2}}{xB_i^j E(B_m^j) + yD_i^j E(U_m^j) + zU_i^j E(U_m^j)}\ , \tag{6.8}$$

where $xB + yD + zU = 1$.

We add three weighted factors x, y and z into Eq. (6.7), enabling T-PARA-WF with flexible ability. It can be adjusted depending on different scenarios. For example, to prevent $\mathcal{ADV}$ from illegally increasing trustworthiness, we can increase the weight of B in Eq. (6.8), i.e., increase x. In contrast, we can increase y to prevent $\mathcal{ADV}$ from illegally decreasing trustworthiness. For preventing $\mathcal{ADV}$ from $\mathcal{ADV}\_Hbd$, i.e., $\mathcal{ADV}$ generates false trust opinions both to increase trustworthiness and to decrease trustworthiness, we can increase z for increasing more weight on uncertainty U.

Note that Eq. (6.6) and Eq. (6.7) are special cases of Eq. (6.8) when $x = 1$, $y = z = 0$ and $x = y = z = 1$, respectively. Furthermore, we have the following observations.

- If $\mathcal{ADV}\_Homo$ intends to increase Υ, ONE-PARA as a function of B is better than T-PARA since B is the only weight factor in it. That is, $x = 1$, $y = z = 0$, meaning that D and U do not have any weight.

- In contrast, if $\mathcal{ADV}\_Homo$ intends to decrease Υ, ONE-PARA as a function of D is better than T-PARA since D is the only weight factor in it.

- T-PARA is resilient to $\mathcal{ADV}\_Noise$, $\mathcal{ADV}\_Homo$ and $\mathcal{ADV}\_Hbd$.

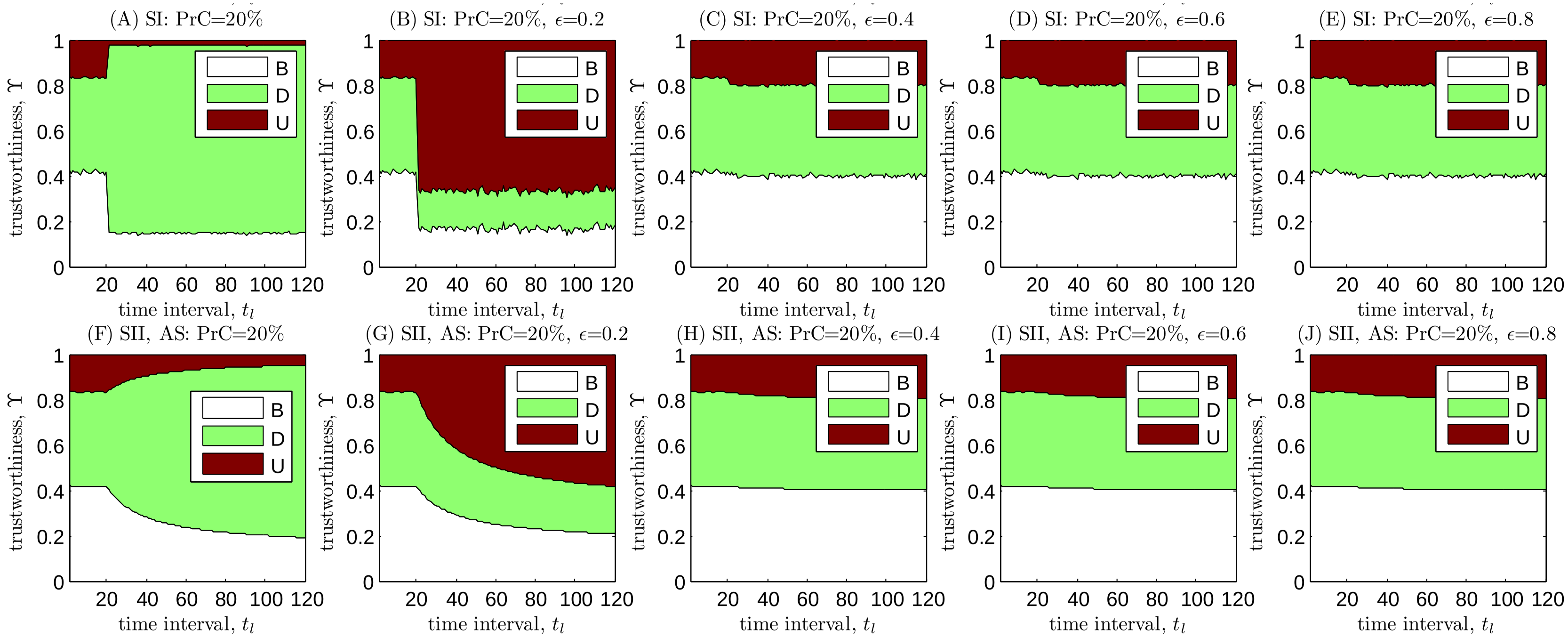


Figure 6.17: An example of three parameters threshold function works while trust consensus only approach and one parameter threshold function does not work well where $cT = \{0.1, 0.1, 0.8\}$, $fT = \{0.1, 0.8, 0.1\}$, $\sigma_c = \sigma_f = 0.01$.

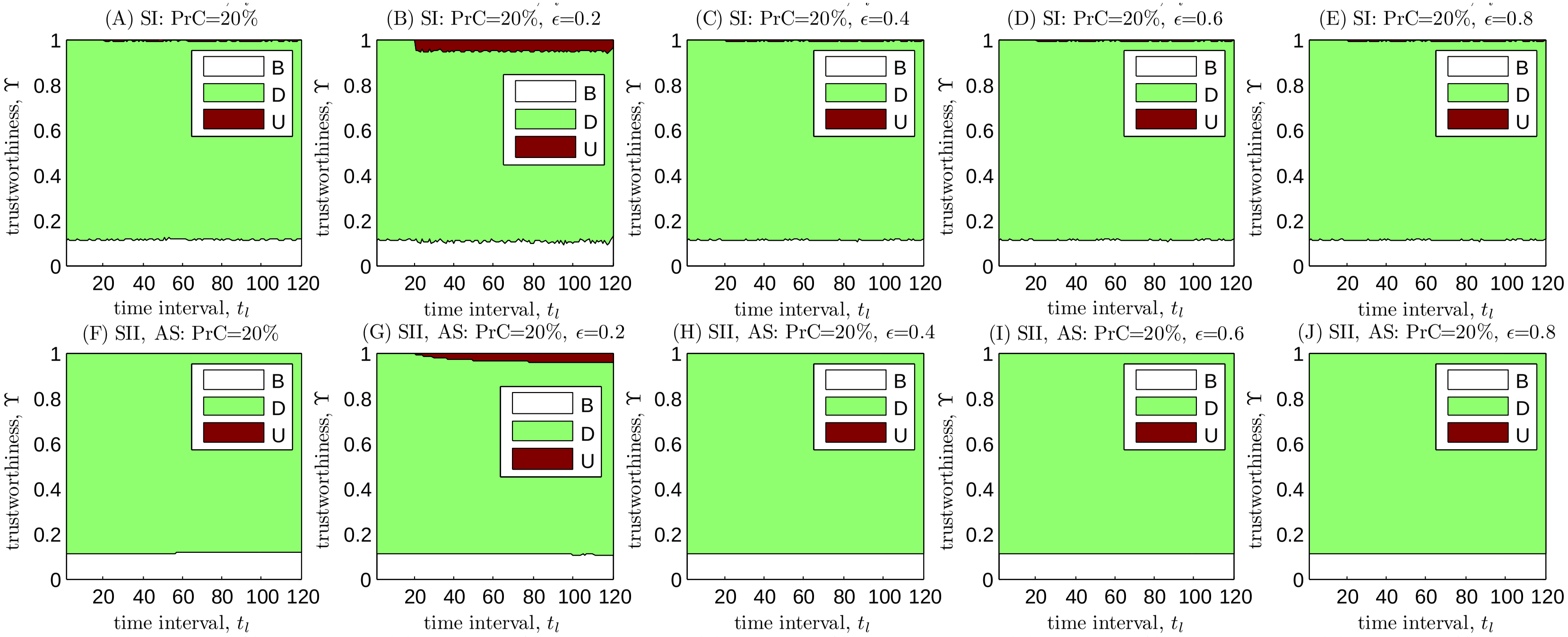


Figure 6.18: An example of trust consensus only approach works well while three parameters threshold function slightly compromised the result when ϵ is selected too small. $cT = \{0.1, 0.8, 0.1\}$, $fT = \{0.1, 0.1, 0.8\}$, $\sigma_c = \sigma_f = 0.01$.

Table 6.1: Impacts of consensus and threshold functions with respect to $d(B_i^j, E(B_m^j))$, $d(D_i^j, E(D_m^j))$ and $d(U_i^j, E(U_m^j))$.

					Threshold Function			
$d(B_i^j, E(B_m^j))$		$d(D_i^j, E(D_m^j))$		$d(U_i^j, E(U_m^j))$	TC-ONLY	ONE-PARA	T-PARA	T-PARA-WF
						(Eq. (Eq. (6.6)))	(Eq. (Eq. (6.7)))	(Eq. (Eq. (6.8)))
≈ 0		≈ 0		≈ 0	good	good	good	good
≈ 0		high	$E(cD) > E(fD)$	-	good	good	good	good
≈ 0			$E(cD) < E(fD)$	-	not enough	not enough	good	good
high	$E(cB) > E(fB)$	-		-	not enough	good	good	good
	$E(cB) < E(fB)$	-		-	not enough	good	good	good

Table 6.2: $\mathcal{ADV}$'s pollution attack strategies and their countermeasures.

	Countermeasures			
$\mathcal{ADV}$'s strategy	TC-ONLY	ONE-PARA	T-PARA	T-PARA-WF
$\mathcal{ADV}\_Noise$	OK	Good	Good	Good
$\mathcal{ADV}\_Homo$: increase Υ	NO	Good	OK	Good
$\mathcal{ADV}\_Homo$: decrease Υ	NO	Good	OK	Good
$\mathcal{ADV}\_Hbd$	NO	NO	Good	Good

- T-PARA-WF is a more flexible way to prevent $\mathcal{ADV}$ from various attacks. The selection of x, y and z is depending on scenarios.

A conclusion of countermeasures against $\mathcal{ADV}$'s pollution attack strategies are summarized in Table 6.2.

6.7 Summary

In this chapter, we have proposed a family of efficient and robust trust management schemes for the first time in UWSNs. The AS utilized distributed trust-related data storage to provide trust-related data reliability and took advantage of both GHT and GPSR to find storage nodes and to route trust-related data to them. In addition, a set of similarity threshold functions are proposed to remove outliers. As the simulation results shown and discussed above, we demonstrated that SII and AS with T-PARA are resilient to $\mathcal{ADV}\_Noise$, $\mathcal{ADV}\_Inva$, $\mathcal{ADV}\_Homo$ and $\mathcal{ADV}\_Hbd$. Moreover, the simulation results showed in Section 6.5 demonstrated that AS requires much less storage cost than SII. Therefore, AS with similarity threshold functions significantly reduced storage cost caused by distributed trust-related data storage, and provided resilience to node invalidation and $\mathcal{ADV}$'s pollution attacks. As our future work, we intend to investigate time-aware trustworthiness generation for UWSNs under different trust models, e.g., indirect trust.

Chapter 7

Sensor Capture Resistance

The previous chapters discussed intrusion-resilient data security when a sensor is captured and compromised. In this chapter we investigate 1) how to detect a captured sensor, and 2) how to reduce the effects of compromised sensors. We propose a node capture resistance and key refreshing scheme based on the Chinese remainder theorem (CRT) for mobile UWSNs. The scheme is able to provide forward security, backward security and collusion resistance for diminishing the effects of capture attacks.

7.1 Related Work

The security issues in Mobile UWSNs (MUWSN) have received little attention so far in general. A few related studies regarding replica node attacks [40] and intrusion-resilience [25] can be found in the literature, but none of them have considered node capture attacks. The first work on detecting mobile node capture attacks is [19], which proposed a scheme depending on cooperation of honest sensors to detect possible sensor captures. However, it did not consider how to provide forward security, backward security, and resistance to collusion attack.

Group key management schemes can be classified into two categories: distributed group key management schemes and centralized group key management schemes. Distributed group key management schemes relaying on Diffie-Hellman (DH) [83, 47, 9] and discrete logarithm [104, 16] are considered to be too expensive in computation for WSNs. Centralized group key management schemes [38, 37, 106] cannot satisfy the overall requirements of forward security, backward security and resistance to collusion attack needed for Mobile

WSNs. A set of improved centralized group key management schemes, e.g., Logical Key Hierarchy (LKH) [107, 110], Efficient Large-group Key (ELK) [68], One-way Function Trees (OFT) [80], One-way Function Chain Tree (OFCT) [13], which are based on certain hierarchical structure, require $O(log\ n)$ keys to be computed, received and stored by each group member, where n is the number of group members. It causes large overhead for sensors with limited resources. The secure broadcasting protocol, Secure Lock [18], requires $O(n)$ encryptions for each real data broadcast. The authors in [118] improved Secure Lock [18] where only one encryption is needed for each real data broadcast. Our work in this study extends [118] by further providing authentication and integrity for distributing group keys.

7.2 Models and Assumptions

7.2.1 Network Model

Consider an MUWSN that consists of $\mathcal{N}$ mobile sensors (denoted $\mathbb{S}$) and one $\mathcal{MS}$ (the number of $\mathcal{MS}$ could be more than one, but for the sake of simplicity we assume only one $\mathcal{MS}$ in the network). All the sensors are partitioned into M groups denoted as $G_1, \cdots, G_k, \cdots, G_M$. Each group consists of N_k sensors denoted as $G_k = \{s_{k,1}, \cdots, s_{k,i}, \cdots, s_{k,N_k}\}$, and an initial group key $\mathcal{GK}_k$ is selected for each group, $k \in [1, M]$. Thus $G_1 \cup \cdots \cup G_M = \mathbb{S}$ and $\sum_{k=1}^{M} N_k = \mathcal{N}$. $s_{k,i}$ is a sensor in group G_k, and is loaded with the corresponding group key $\mathcal{GK}_k$ and a secret key $\mathcal{K}_{k,i}$, which are set up before the deployment of the MUWSN, where $\mathcal{K}_{k,i}$ is an integer over $GF(2^q)$, q is the length of keys, and $gcd(\mathcal{K}_{k,i}, \mathcal{K}_{k,j}) = 1, i \neq j$. The $\mathcal{MS}$ stores all secret keys of sensors, denoted as $\{\mathbb{K}_1, \cdots, \mathbb{K}_k, \cdots \mathbb{K}_M\}$, where $\mathbb{K}_k$ is a set of secret keys of G_k and $\mathbb{K}_k = \{\mathcal{K}_{k,1}, \cdots, \mathcal{K}_{k,i}, \cdots, \mathcal{K}_{k,N_k}\}$. Each sensor is able to perform modular and XOR operations. $\mathcal{MS}$ is not constrained by available computational capability and energy. This is a reasonable assumption in many monitoring applications. For example, we can deploy more than one unmanned submarines where one is on duty and the rest can be on the way or recharge batteries. In addition, the $\mathcal{MS}$ is considered as a trusted party that cannot be compromised.

7.2.2 Adversary Model

We assume that a mobile adversary $\mathcal{M}\_\mathcal{ADV}$ is able to capture one node at a time and remove it from the network, and that the only way to access (or modify) the memory of a sensor is first to remove the sensor from the network and then to temper with it. Here, we define time interval $\mathcal{CP}$ as $\mathcal{M}\_\mathcal{ADV}$'s compromise power, meaning that $\mathcal{M}\_\mathcal{ADV}$ is able to capture one node, compromise it to learn its secret key, and re-deploy it back to the network within the time interval of $\mathcal{CP}$. Moreover, based on the secret key obtained from captured sensor nodes, $\mathcal{M}\_\mathcal{ADV}$ is able to launch the following attacks to the network:

- Surveillance hole attack: $\mathcal{M}\_\mathcal{ADV}$ re-programmes the captured sensor and redeploys it into the network to make an unsurveilled hole so that $\mathcal{M}\_\mathcal{ADV}$ can go through the surveillance area undiscovered.
- Communication analysis attack: $\mathcal{M}\_\mathcal{ADV}$ is able to record the communication within the network and try to decrypt the communication record.
- Node replica attack: $\mathcal{M}\_\mathcal{ADV}$ is able to generate a large number of replica nodes and spread them throughout the network to mix up the network communication.

7.2.3 Security Requirements

As all the sensors are divided into several groups, group communication is encrypted and thus protected by group keys. Security requirements are thus converted to how to update group key for all group members except the revoked sensor, and further prevent the group communication from suffering attacks (see Subsection 7.2.2) launched by $\mathcal{M}\_\mathcal{ADV}$ with the help of key material from the captured sensor. A group key $\mathcal{GK}_k$ can be updated m times each with a key as $\mathcal{GK}_{k,t}$, $t \in [1, m]$. Denote the set of group keys as $\mathbb{GK}_k = \{\mathcal{GK}_{k,1}, \cdots, \mathcal{GK}_{k,t}, \cdots, \mathcal{GK}_{k,m}\}$. We now define security requirements.

Definition 7.1. (**Group Key Security**) Group key security is provided if for any sensor $s_{k,i} \in G_k$, it is efficient to obtain $\mathcal{GK}_{k,t}$ based on its secret key $\mathcal{K}_{k,i}$ and broadcast value $\mathcal{X}$. For any sensor $s_{k',i'} \notin G_k$, it holds that:

$$|Pr\{\mathcal{GK}_{k',t'}|\mathcal{X}, \mathcal{K}_{k',t'}\} - Pr\{\mathcal{GK}_{k',t'}\}| < \epsilon(q),$$

where $\epsilon(q)$ is a negligible polynomial related to security parameter q. q usually is the length of the security parameter.

Definition 7.2. (**Backward Security**) Backward security is provided if for any set $R_{k,t'} \subset G_k$, where $R_{k,t'}$ is a set of revoked sensors from G_k before session t', it is computationally infeasible for sensors from $R_{k,t'}$ working together to get any information about $\mathcal{GK}_{k,t}\,(t \geq t')$, even when the group keys $\{\mathcal{GK}_{k,1}, \cdots, \mathcal{GK}_{k,t'-1}\}$ are available.

Definition 7.3. (**Forward Security**) Forward security is provided if for any set $\mathcal{A}_{k,t'} \subset \mathbb{S}$, where $\mathcal{A}_{k,t'}$ is a set of added sensors after session t', it is computationally infeasible for sensors from $\mathcal{A}_{k,t'}$ working together to obtain any knowledge about $\mathcal{GK}_{k,t}$ $(t < t')$, even when a set of group keys $\{\mathcal{GK}_{k,t'}, \mathcal{GK}_{k,t'+1}, \cdots, \mathcal{GK}_{k,m}\}$ after session t' are available.

Definition 7.4. (**Collusion Resistance**) Let $E \subseteq R_{k,t'}$ be a collusion of sensors revoked before session t' and let $F \subseteq \mathcal{A}_{k,t''}$ be a set of colluding sensors added after session t''. For any $t \in T = [t', t'')$, sensors in $E \cup F$ cannot gain any information about $\mathcal{GK}_{k,t}$ even by sharing their knowledge.

7.3 SCARKER: Sensor Capture Resistance and Key Refreshing

In this section, we present our SCARKER scheme. Before giving the detailed description, we first present the general idea of the sensor capture detection scheme based on [19].

7.3.1 General Idea

The basic idea of the captured sensor detection scheme [19] is that a sensor can raise an alert message if it *detects* that another sensor has disappeared since each sensor is aware of the existence of other sensors in the network. The detection scheme can be further divided into two classes: Simple Distributed Detection (SDD), in which a captured sensor is detected using information local to the sensors; and Cooperative Distributed Detection (CDD), which exploits local sensor cooperation to improve the detection performance.

Let us consider an example, as addressed in Section 7.2, where each sensor $s_{k,i}$ shares an initial secret key $\mathcal{K}_{k,i}$ and a group key $\mathcal{GK}_k$ with $\mathcal{MS}$. In the beginning, none of those keys are known to $\mathcal{M}\_\mathcal{ADV}$. Let $s_{k,a}, s_{k,b}, s_{k,c} \in G_k$ $(a \neq b \neq c)$ and these three sensors are responsible for tracking each other. With SDD, if sensor $s_{k,a}$ has observed a transmission originated by sensor $s_{k,b}$ (a *meeting* occurred), but $s_{k,a}$ does not re-meet sensor $s_{k,b}$ again for a given time interval λ_k[1] $(\lambda_k < \mathcal{CP})$, then $s_{k,a}$ can deduce that $s_{k,b}$ has probably been captured by $\mathcal{M}\_\mathcal{ADV}$, and consequently sends an alert message to $\mathcal{MS}$, where Pr_{re-m} denotes sensor re-meeting probability. To improve the performance of SDD, $s_{k,a}$ in CDD can take the advantage of other sensors' *meeting* information with $s_{k,b}$ to reduce false alert probability. That is, $s_{k,a}$ and $s_{k,c}$ can exchange *meet* time information T_b of $s_{k,b}$ when they meet, where $s_{k,a}$ and $s_{k,c}$ can compare T_b when they met $s_{k,b}$ last time and update the time which is more recent. Indeed, CDD can reduce false revocation since if $s_{k,a}$ does not re-meet $s_{k,b}$ within λ_k (even though $\overline{Pr_{re-m}} = 1 - Pr_{re-m} \approx 0$, it still happens *especially* in practical MUWSN), $s_{k,a}$ can use T_b from other sensors as an evidence that $s_{k,b}$ is still a legal group member.

Upon receiving γ alert messages from sensors (we will discuss later the selection of γ), $\mathcal{MS}$ will broadcast a revoke-sensor message (see Subsection 7.3.4) in order to revoke sensor which is considered as a captured one. If, N_k, the number of sensors in a group decreases due to sensor revoking, it makes sensors more difficult to meet each other (λ_k increases) and it may also increase false alert messages. As a consequence, $\mathcal{MS}$ needs to add new sensors into the group to guarantee network connectivity. When $N_k < N_T$, $\mathcal{MS}$ will broadcast an add-sensor message (described in Subsection 7.3.5 and Subsection 7.3.6) to compute new group key for including new sensor into the group, where N_T is a pre-specified threshold and can be *specified* as the number of sensors in G_k when Pr_{re-m} approaches to 1.

Discussion: The proposed scheme may have *false positive*s. That is, $s_{k,b}$ may be revoked from G_k by mistake, while $s_{k,b}$ is an uncaptured node. Indeed, false positives exist if $\mathcal{MS}$ decides to revoke $s_{k,b}$ as long as $\mathcal{MS}$ receives only one alert message regarding $s_{k,b}$. To reduce false positives, $\mathcal{MS}$ can decide to

[1] To specify time interval λ_k, the authors in [19] conducted a simulation based on the random way-point mobility model [11] to investigate the sensor re-meeting probability that two sensors re-meet again after λ_k seconds. Through simulation, λ_k can be specified as when the sensor re-meeting probability Pr_{re-m} approaches to 1. To guarantee $\lambda_k < \mathcal{CP}$, given a fixed area, increasing N_k will decrease λ_k.

revoke $s_{k,b}$ from G_k if up to γ ($\gamma \geq 1$) alert messages regarding $s_{k,b}$ are received. By doing so, the false positives decreases as γ increases. However, if $\mathcal{MS}$ tolerates $\gamma - 1$ alert messages, the proposed scheme would have *false negatives*, which occur when a captured node is considered, by mistake, as a normal one. As γ increases, $\mathcal{MS}$ ignores more alert messages, which may lead to false negatives. We observe that false negatives have more serious consequences than false positives because false positives only cause that the number of sensors decreases faster, whereas, false negatives give chances to $\mathcal{M}\_\mathcal{ADV}$ to launch various attacks as described in Subsection 7.2.2. It is therefore more realistic to tolerate a few false positives but have no false negatives. Hence, we set $\gamma = 1$ in our scheme. Furthermore, to keep zero false negative, $\mathcal{MS}$ needs to add sensors in G_k to guarantee $N_k > N_T$. Table 7.1 describes the influence of N_T on λ_k, the detection time, the number of false positives and the number of false negatives.

Table 7.1: Influence of N_T on λ_k, detection time, false positives and false negatives.

N_T	λ_k	Detection Time	False Positives	False Negatives
Increase ($\nearrow$)	$\searrow$	$\searrow$	$\searrow$	$\searrow$
Decrease ($\searrow$)	$\nearrow$	$\nearrow$	$\nearrow$	$\nearrow$

7.3.2 Broadcast Value Generation

In this subsection, the details of broadcast value generation algorithms are addressed.

Firstly, $\mathcal{MS}$ picks a random number $\mathcal{GK}'_k$ ($\mathcal{GK}'_k \neq \mathcal{GK}_k$) over $GF(2^q)$ as a new group key if $\mathcal{GK}'_k \oplus \mathcal{K}_{k,i} < \mathcal{K}_{k,i}$, and then computes $r_i (1 \leq i \leq N_k)$ using $\mathcal{GK}'_k$ and secret keys $\mathcal{K}_{k,i}$ shared with sensors as follows:

$$r_i \leftarrow \mathcal{GK}'_k \oplus \mathcal{K}_{k,i}, \tag{7.1}$$

where $r_i < \mathcal{K}_{k,i}$, and further constructs a congruence system using r_i as follows:

$$\begin{cases} \mathcal{X} \equiv & r_1 \,(\mathrm{mod}\, \mathcal{K}_{k,1}) \\ \mathcal{X} \equiv & r_2 \,(\mathrm{mod}\, \mathcal{K}_{k,2}) \\ & \vdots \\ \mathcal{X} \equiv & r_i \,(\mathrm{mod}\, \mathcal{K}_{k,i}) \\ & \vdots \\ \mathcal{X} \equiv & r_{N_k} \,(\mathrm{mod}\, \mathcal{K}_{k,N_k}). \end{cases} \tag{7.2}$$

Obviously, the above congruence system satisfies the requirement of a CRT system having a unique solution. For any broadcast value $\mathcal{X}$ obtained from congruence system (Eq. (7.2)), the group key, $\mathcal{GK}'_k$, can be obtained according to $\mathcal{GK}'_k = ((\mathcal{X} \bmod \mathcal{K}_{k,i}) \oplus \mathcal{K}_{k,i})$. The $\mathcal{MS}$ is thus able to solve the congruence system and find $\mathcal{X}$. Algorithm 3 and Algorithm 4 are given for new broadcast value generation and new group key generation, respectively.

7.3.3 Membership Adjustment Command

The following commands need to be implemented in our scheme:

- Add sensor: a new sensor is added into the group.
- Revoke sensor: a sensor is revoked from the group.
- Key update: the group key is updated.
- Merge: merge two groups of sensors.

The following subsections introduce the three operations that implement the above commands. Note that key update is a special case of revoke command protocol, where the group key is updated without any sensor revoked from the group.

7.3.4 Sensor Revocation

Let us consider a group of sensors $G_k = \{s_{k,1}, s_{k,2}, \cdots, s_{k,N_k}\}$. When $\mathcal{MS}$ concludes that a sensor is captured and should be revoked from the group, it executes a CRT-based scheme to update the group key of each sensor in the

```
Algorithm 3: $\mathcal{X}\_GEN$
  /* Generate a new broadcast value and broadcast it.                    */
  INPUT   : $\mathbb{K}_k = \{\mathcal{K}_{k,1}, \mathcal{K}_{k,2}, \cdots, \mathcal{K}_{k,j}, \cdots, \mathcal{K}_{k,N_k}\}$, $\mathcal{GK}_k$
  OUTPUT: broadcast value $\mathcal{X}$
select $\mathcal{GK}'_k \in GF(2^p)$, where $\mathcal{GK}'_k \neq \mathcal{GK}_k$ and $\mathcal{GK}'_k \oplus \mathcal{K}_{k,i} < \mathcal{K}_{k,i}$.;
  /* Compute a new broadcast value $\mathcal{X}$ using CRT.                  */
for $\{i = 1;\ i \leq N_k;\ i++\}$ do
    $r_i \leftarrow \mathcal{GK}'_k \oplus \mathcal{K}_{k,i}$.;
end
for $\{i = 2;\ i < N_k;\ i++\}$ do
    $c_i \leftarrow 1$
    for $\{j = 1; j < i - 1; j++\}$ do
        $U \leftarrow \mathcal{K}_{k,j}^{-1} \bmod \mathcal{K}_{k,j}$
        $c_i \leftarrow U * c_i \bmod \mathcal{K}_{k,j}$
    end
end
$U \leftarrow r_1$
$\mathcal{X} \leftarrow U$
for $\{i = 2; i < N_k; i++\}$ do
    $U \leftarrow (r_i - x) * c_i \bmod \mathcal{K}_{k,i}$
    $\mathcal{X} \leftarrow \mathcal{X} + U * \Pi_{j=1}^{i-1} \mathcal{K}_{k,j}$
end
delete $\{r_1, r_2, \cdots, r_{N_k}\}$
$Seq\_no \leftarrow Seq\_no + 1$;
return $\{\mathcal{X}||Seq\_no||h(\mathcal{X}||Seq\_no)||h(\mathcal{GK}'_k)\}$
```

group. In the proposed scheme, each sensor of the group except the revoked sensor uses the broadcast value $\mathcal{X}$, computed by $\mathcal{MS}$, to compute a new group key $\mathcal{GK}_k$. We now present the protocol in details.

On the $\mathcal{MS}$ side: Based on secret keys $\mathbb{K}_k$ and a current group key $\mathcal{GK}_k$, $\mathcal{MS}$ generates a new broadcast value $\mathcal{X}$ according to Algorithm 3 and broadcasts it. Here, we assume a suitable broadcast authentication protocol, e.g., multilevel μTESLA [51] for secure and reliable transmission of such broadcast value $\mathcal{X}$. That is,

$$\mathcal{MS} \rightarrow * : \ \{\mathcal{X}||Seq\_no||h(\mathcal{X}||Seq\_no)||h(\mathcal{GK}'_k||Seq\_no)\},$$

where Seq_no is a nonce generated by $\mathcal{MS}$ (a nonce is an unpredictable bit string, usually used to achieve freshness), $h(\cdot)$ is a secure hash function, $\mathcal{GK}'_k$ is the group key and "||" is concatenation. Thus it is easy for sensor to check

Algorithm 4: $\mathcal{GK}\_GEN$

```
/* Compute a new group key based on broadcast value and
   secret key.                                              */
INPUT   : $\mathcal{K}_{k,i}$, $\{\mathcal{X}||Seq\_no||h(\mathcal{X}||Seq\_no)||h(\mathcal{GK}'_k||Seq\_no)\}$
OUTPUT: new group key $\mathcal{GK}'_k$
/* Compute the hash value of received $\mathcal{X}$ and $Seq\_no$.*/
$temp \leftarrow h(\mathcal{X}||Seq\_no)$;
/* Compare $temp$ with received $h(\mathcal{X}||Seq\_no)$.*/
if ($temp \overset{?}{=} h(\mathcal{X}||Seq\_no)$ received) then
    compute $\mathcal{GK}'_k \leftarrow ((\mathcal{X} \bmod \mathcal{K}_{k,i}) \oplus \mathcal{K}_{k,i})$
    if ($h(\mathcal{GK}'_k||Seq\_no) \overset{?}{=} h(\mathcal{GK}'_k||Seq\_no)$ received) then
        return $\mathcal{GK}'_k$;
    else
        /* This message is not for this sensor. */;
        Drop this message;
    end
else
    return received message is broken.;
end
```

the integrity of $\mathcal{X}$ (see Algorithm 4).

On the sensor side: Based on the received message and $\mathcal{K}_{k,i}$, each sensor $s_{k,i}$ computes a new group key $\mathcal{GK}'_k$ according to Algorithm 4.

If the captured sensor is $s_{k,i}$, then $\mathcal{MS}$ deletes its secret key $\mathcal{K}_{k,i}$ from $\mathbb{K}_k$. That is

$$\mathbb{K}'_k \leftarrow \mathbb{K}_k \setminus \{\mathcal{K}_{k,i}\}. \tag{7.3}$$

New broadcast value $\mathcal{X}'$ is thus obtained based on $\mathbb{K}'_k$ according to Algorithm 3. Then $\mathcal{MS}$ broadcasts $\mathcal{X}'$ to all the sensors of group G_k.

Upon receiving $\mathcal{X}'$, the sensors within group G_k will compute the new group key $\mathcal{GK}'_k$ according to Algorithm 4. Indeed, the revoked sensor $s_{k,i}$ can also get $\mathcal{X}'$, however, it is not able to compute $\mathcal{GK}'$ based on $\mathcal{X}'$ (we defer the detailed proof to Section 7.4 and a numeral example to Subsubsection 7.3.7.1).

7.3.5 Sensor Join Operation

Let $s_{k',i} \notin G_k$ be the new sensor to be added. $\mathcal{MS}$ adds its corresponding secret key $\mathcal{K}_{k',i}$ into $\mathbb{K}_k$. That is

$$\mathbb{K}'_k \leftarrow \mathbb{K}_k \cup \{\mathcal{K}_{k',i}\}.$$

Again, $\mathcal{MS}$ computes $\mathcal{X}'$ according to Algorithm 3 based on $\mathbb{K}'_k$, and broadcasts $\mathcal{X}'$ to $G'_k = G_k \cup \{s_{k',i}\}$. The sensors within G'_k are able to compute $\mathcal{GK}'_k$ based on $\mathcal{X}'$.

7.3.6 Group Merge Operation

As discussed in Subsection 7.3.1, a network may be required to add several sensors to improve network performance.

For instance, to add new sensors into group G_k when the number of members in a group is lower than the certain threshold, we can deploy a new sensor group, G_l, with group key $\mathcal{GK}_l$ and combine these two groups into one. After deploying the new sensor group G_l, $\mathcal{MS}$ combines two sets of secret keys ($\mathbb{K}_k$ and $\mathbb{K}_l$). That is

$$\begin{aligned} \mathbb{K}'_k &= \mathbb{K}_k \cup \mathbb{K}_l \\ &= \{\mathcal{K}_{k,1}, \cdots, \mathcal{K}_{k,N_k}, \mathcal{K}_{l,1}, \cdots, \mathcal{K}_{l,N_l}\}. \end{aligned} \tag{7.4}$$

$\mathcal{MS}$ is able to compute $\mathcal{X}'$ based on $\mathbb{K}'_k$ according to Algorithm 3 and broadcast it to G_k and G_l. Again, after receiving the broadcast message, the sensors within G_k and G_l will compute $\mathcal{GK}'$ according to Algorithm 4 respectively (see Subsubsection 7.3.7.2 for a numeral example).

7.3.7 Two Numeral Examples

Two simple examples are given below to demonstrate how the proposed scheme works.

7.3.7.1 Sensor revoke operation example

For simplicity, we assume that there are three sensors denoted as $s_{2,1}$, $s_{2,2}$, and $s_{2,3}$ in a group, G_2, with secret keys $\mathcal{K}_{2,1} = 17$, $\mathcal{K}_{2,2} = 19$, and $\mathcal{K}_{2,3} = 15$, respectively. Assume that the $\mathcal{MS}$ regards $s_{2,3}$ is a captured node. The $\mathcal{MS}$

can delete $\mathcal{K}_{2,3}$ from $\mathbb{K}_2$ according to Eq. (7.3), i.e., $\mathbb{K}_2' \leftarrow \mathbb{K}_2 \setminus \{\mathcal{K}_{2,3}\}$. The $\mathcal{MS}$ computes broadcast value $\mathcal{X} = 24$ and broadcasts it to $s_{2,1}$, $s_{2,2}$ and $s_{2,3}$ on input $\mathbb{K}_2'$ according to Algorithm 3.

Upon receiving the value, $\mathcal{X} = 24$, the sensors in G_2 can compute group key $\mathcal{GK}_2'$ according to Algorithm 4 as follows:

$$\begin{aligned} s_{2,1} : \mathcal{GK}_2' = 22 &\quad\leftarrow\quad ((24 \bmod 17) \oplus 17) \\ s_{2,2} : \mathcal{GK}_2' = 22 &\quad\leftarrow\quad ((24 \bmod 19) \oplus 19) \\ s_{2,3} : \mathcal{GK}_2' = 14 &\quad\leftarrow\quad ((24 \bmod 15) \oplus 15). \end{aligned}$$

We consider $s_{2,3}$ as revoked from G_2 since it cannot compute the new group key which is $\mathcal{GK}_2' = 22$.

7.3.7.2 Group merge operation example

Since sensor join operation is a special case of group merge operation where there is only one sensor added in the group $\mathbb{K}_l$, we use an example to illustrate how to add sensors in a group. We continue the example above. Since $s_{2,3}$ is regarded as a captured sensor and revoked afterwards, the $\mathcal{MS}$ needs to add new sensors to maintain network connectivity. Assume that a group G_5 ($s_{5,1}$ and $s_{5,2}$ with secret keys $\mathcal{K}_{5,1} = 29$ and $\mathcal{K}_{5,2} = 31$ respectively) is going to be added into G_2. The $\mathcal{MS}$ applies Algorithm 3 to $\mathbb{K}_2'' = \mathbb{K}_2' \cup \mathbb{K}_5 = \{\mathcal{K}_{2,1}, \mathcal{K}_{2,2}, \mathcal{K}_{5,1}, \mathcal{K}_{5,2}\}$ to obtain $\mathcal{X} = 149214$ and then broadcasts it to all sensors.

Upon receiving the value $\mathcal{X} = 149214$, the sensors compute group key $\mathcal{GK}_2''$ according to Algorithm 4 as follows:

$$\begin{aligned} s_{2,1} : \mathcal{GK}_2'' = 20 &\quad\leftarrow\quad ((149214 \bmod 17) \oplus 17) \\ s_{2,2} : \mathcal{GK}_2'' = 20 &\quad\leftarrow\quad ((149214 \bmod 19) \oplus 19) \\ s_{5,1} : \mathcal{GK}_2'' = 20 &\quad\leftarrow\quad ((149214 \bmod 29) \oplus 29) \\ s_{5,2} : \mathcal{GK}_2'' = 20 &\quad\leftarrow\quad ((149214 \bmod 31) \oplus 31). \end{aligned}$$

As a consequence, both G_2 and G_5 are able to get group key $\mathcal{GK}_2'' = 20$.

7.4 Security Analysis

In this section, we analyze the security aspects of SCARKER in terms of group key security, forward security, backward security and collusion attack. We will now show how our scheme confirms with all the security requirements specified in Definitions 7.1- 7.4.

Lemma 7.5. *$\mathcal{M}\_\mathcal{ADV}$ cannot derive $r_1, \cdots, r_{N_k}$ based on the broadcast value $\mathcal{X}$ without knowing $\mathcal{K}_{k,1}, \cdots, \mathcal{K}_{k,N_k}$.*

Proof. Since the broadcast value $\mathcal{X}$ can be rewritten as $\mathcal{X} = a\mathcal{K}_{k,i} + r_i$, where a is an integer and $1 \leq i \leq N_k$, $\mathcal{M}\_\mathcal{ADV}$ must guess $\mathcal{K}_{k,1}, \cdots, \mathcal{K}_{k,N_k}$ to derive possible $r_1, \cdots, r_{N_k}$. However, it is computationally infeasible to guess $r_1, \cdots, r_{N_k}$ because of $\mathcal{K}_{k,1}, \cdots, \mathcal{K}_{k,N_k}$ are secret even though $\mathcal{M}\_\mathcal{ADV}$ has the broadcast value $\mathcal{X}$. Furthermore, since broadcast value $\mathcal{X}$ varies from time to time, $\mathcal{M}\_\mathcal{ADV}$ cannot verify the correctness of guess even when $\mathcal{M}\_\mathcal{ADV}$ has guessed the correct $\mathcal{K}_{k,1}, \cdots, \mathcal{K}_{k,N_k}$. As a consequence, analyzing a set of broadcast values $\mathcal{X}$ without knowing $\mathcal{K}_{k,1}, \cdots, \mathcal{K}_{k,N_k}$ cannot give $r_1, \cdots, r_{N_k}$. □

Proposition 7.6. *The proposed scheme provides backward security.*

Proof. Let $\mathcal{R}_{k,t'} \subset G_k$, where $\mathcal{R}_{k,t'}$ is a set of revoked sensors from G_k before the current session t'. The knowledge of $\mathcal{R}_{k,t'}$ consists of a set of broadcast values $\mathcal{X}$, revoked sensors' secret keys $\mathcal{K}_{R_{k,t'}}$ and the previous group keys $\mathcal{GK}_{k,1}, \cdots, \mathcal{GK}_{k,t'-1}$ they obtained when they were in G_k. However, since the secret keys of $\mathcal{R}_{k,t'}$ are deleted from the new congruence system (in Eq. (7.3)), the group key $\mathcal{GK}_{k,t}$ $(t > t')$ is computed based on a new set of secret keys of sensors $G_k \setminus \mathcal{R}_{k,t'}$. Thus a sensor in $\mathcal{R}_{k,t'}$ can compute $\mathcal{GK}_{k,t}$ unless it is able to obtain the secret keys of $G_k \setminus \mathcal{R}_{k,t'}$. Moreover, according to Lemma 7.5 it cannot derive $\mathcal{GK}_{k,t}$ based on broadcast value $\mathcal{X}$ without knowing the secret keys of $G_k \setminus \mathcal{R}_{k,t'}$. As a result, it cannot compute $\mathcal{GK}_{k,t}$ to access the future group communication any more. Thus, the backward security is provided. □

Proposition 7.7. *The proposed scheme provides forward security.*

Proof. Let $\mathcal{A}_{k,t'} \not\subseteq G_k$, where $\mathcal{A}_{k,t'}$ denotes a set of new sensors added into G_k after session t'. The information of $\mathcal{A}_{k,t'}$ knows is a set of broadcast values $\mathcal{X}$, the newly added sensors' secret keys $\mathcal{K}_{\mathcal{A}_{k,t'}}$ and the group keys $\mathcal{GK}_{k,t'}, \cdots, \mathcal{GK}_{k,m}$ they obtained after joining the group. Since the new group key $\mathcal{GK}'_{k,t}$ of G'_k, where $G'_k = G_k \cup \mathcal{A}_{t'}$, is randomly selected and has no relation to any previous group key, the knowledge about the newly added sensors does not help to compute $\mathcal{GK}'_{k,t}$. The previous group key can be successfully derived if and only if $\mathcal{M}\_\mathcal{ADV}$ can solve the congruence system without knowing the corresponding secret keys of G_k, which is impossible according to Lemma 7.5. Therefore, the forward security is provided. □

Proposition 7.8. *The proposed scheme prevents group communication from collusion attack.*

Proof. Let $E \subseteq R_{k,t'}$ be a collusion of sensors revoked before session t' and let $F \subseteq \mathcal{A}_{k,t''}$ be a collusion of sensors added after session t'' where $R_{k,t'} \cup A_{k,t''} \neq G'_k$. Consider the worst case (i.e., $E = R_{k,t'}$ and $F = \mathcal{A}_{k,t''}$). The sensors of $E \cup F$ work together to launch collusion attack based on the knowledge they obtained when $R_{k,t'}$ was in the group before session t' and $\mathcal{A}_{k,t''}$ was in the group after session t''. The knowledge of $R_{k,t'}$ got is the group keys $\mathbb{GK}'_k = \{\mathcal{GK}_{k,1}, \cdots, \mathcal{GK}_{k,t'-1}\}$ before session t' and their secret keys $\mathcal{K}_{R_{k,t'}}$. Meanwhile, the information of $\mathcal{A}_{k,t''}$ consists of the group keys $\mathbb{GK}''_k = \{\mathcal{GK}_{k,t''}, \cdots, \mathcal{GK}_{k,m}\}$ and the sensors secret keys $\mathcal{K}_{\mathcal{A}_{k,t''}}$. Since broadcast values $\mathcal{X}$ are broadcast, they can be known by both $R_{k,t'}$ and $\mathcal{A}_{k,t''}$. The knowledge of $R_{k,t'} \cup \mathcal{A}_{k,t''}$ is represented by $\{\mathcal{K}_{R_{k,t'}} \cup \mathcal{K}_{\mathcal{A}_{k,t''}}, \mathcal{X}, \mathbb{GK}'_k, \mathbb{GK}''_k\}$. According to Eq. (2.1) in Chapter 2, $\mathcal{X}$ can be presented as

$$\mathcal{X} = r_1 Q_1 U_1 + r_2 Q_2 U_2 + \cdots + r_{N_k} Q_{N_k} U_{N_k} \pmod{K_k}, \tag{7.5}$$

where $K_k = \prod_{i=1}^{N_k} \mathcal{K}_{k,i}$, $K_k \cap (\mathcal{K}_{R_{k,t'}} \cup \mathcal{K}_{\mathcal{A}_{k,t''}}) = \emptyset$, $Q_i = \frac{K_k}{\mathcal{K}_{k,i}}$ and U_i is its multiplicative inverse modulo $\mathcal{K}_{k,i}$, i.e., $U_i = Q_i^{-1} \pmod{\mathcal{K}_{k,i}}$. Due to $K_k \cap (\mathcal{K}_{R_{k,t'}} \cup \mathcal{K}_{\mathcal{A}_{k,t''}}) = \emptyset$, the sensors in $R_{k,t'} \cup \mathcal{A}_{k,t''}$ would have to solve Eq. (7.5) containing at least k variables where $k \neq 1$ in order to guess $\mathcal{X}$. Furthermore, since the congruence system is updated after each group membership change and $R_{k,t'} \cup A_{k,t''} \neq G'_k$, the secret information held by sensors in $R_{k,t'} \cup A_{k,t''}$ will not give any knowledge about the new congruence system as long as $\mathcal{K}_{G'_k}$ is large enough, where $\mathcal{K}_{G'_k}$ is a set of secret keys of G'_k. Therefore, according to Lemma 7.5, the proposed scheme is collusion resistant. □

7.5 Performance Evaluation

In this section, the performance of SCARKER is evaluated from both the sensor perspective and the $\mathcal{MS}$ perspective.

1. **Sensor side**: We implement the SCARKER on a Sun SPOT [86] based sensor network testbed with 180 MHz 32 bit ARM920T core, 512K RAM and 4M flash (ROM). The experiment is iterated 100 times to reduce randomness.

2. $\mathcal{MS}$ **side**: The experimental results are obtained through a Maple [61] simulator we developed. The experiment is iterated 25 times and is conducted in an Acer laptop with Inter Celeron M processor 520 (1.6 GHz, 533 MHz, 1 MB L2 cache) and 1 G RAM for evaluating the computation cost of $\mathcal{MS}$.

The experimental results in terms of computation cost, storage cost and communication cost are given below.

7.5.1 Time and Energy Required for Computing New Group Key

Updating a new group key, each sensor needs to perform one modular operation $\mathcal{X} \bmod \mathcal{K}_{k,i}$ and one XOR operation. The total computation cost at each sensor is $Mod^q_{\mathcal{X}} + XOR^q$, where $Mod^q_{\mathcal{X}}$ denotes that $\mathcal{X}$ modulo a number with size q, and XOR^q denotes XOR operation with input size of q. Consequently we can see that the computation cost is a function of q and $\mathcal{X}$. As shown in Fig. 7.1, we observe that the length of $\mathcal{X}$ is affected by q and N_k. For $q = 24$ and $N_k = 30$, the length of $\mathcal{X}$ is 705 bits. Upon receiving a 705 bits $\mathcal{X}$, each sensor performs operation $Mod^q_{\mathcal{X}} + XOR^q$. We measure that the time for computing group key is 170 ms and its corresponding energy consumption is 64 mJ. Please refer to Table 7.2 for more experiment results of computation cost and energy consumption according to Algorithm 4 with respect to q and N_k. Our experiment results show that the computation cost is low and affordable by sensors.

On the $\mathcal{MS}$ side, the $\mathcal{MS}$ needs to compute Algorithm 3 caused by group member change. Fig. 7.2 shows the execution time of group key generation in terms of q and N_k. We can observe that the execution time increases linearly

Table 7.2: Sensor side: Experiment results in terms of the length of secret key $\mathcal{K}_{k,i}$ and the number of group members N_k.

	Computation cost (ms)				Energy consumption (mJ)			
N_k	10	20	30	40	10	20	30	40
$q = 24$(bit)	56	113	170	228	16.5	40	64	89
$q = 28$(bit)	129	267	407	546	49	105	165	225

with the increasing of q and N_k. When $q = 28$ and $N_k = 30$, the $\mathcal{MS}$ needs merely 48.8 ms to generate the broadcast value $\mathcal{X}$. This result indicates that the computation cost of the proposed scheme is lightweight, even when the updating of keys is frequent.

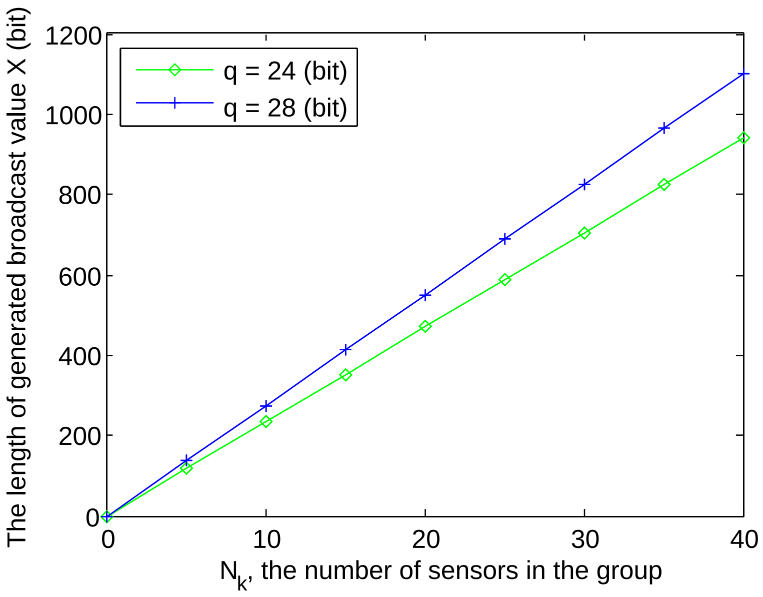


Figure 7.1: $\mathcal{MS}$ side: The length of generated broadcast value $\mathcal{X}$ in terms of the length of secret key $\mathcal{K}_{k,i}$.

7.5.2 Storage Cost

The group key storage cost can be divided into two shares. On the one hand, each sensor needs to spend q bits to store the group key; on the other hand, it also needs to prepare $log_2\mathcal{X}$ bits as buffer to store $\mathcal{X}$. Upon computing a new group key $\mathcal{GK}'$, the sensors just delete $\mathcal{X}$. Thus the storage cost of each sensor is $2q$ bits without buffering and $2q + log_2\mathcal{X}$ bits with buffering. Fig. 7.3 shows that the storage cost on each sensor with respect to q and N_k. When

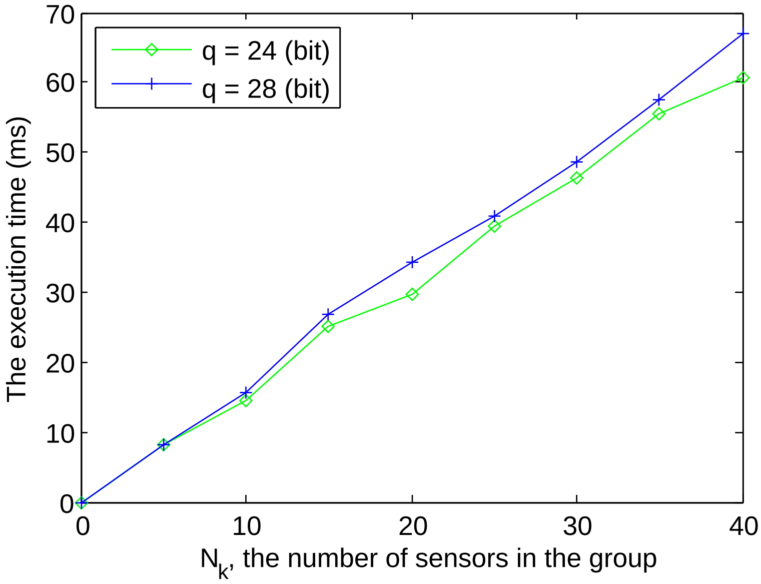


Figure 7.2: $\mathcal{MS}$ side: The execution time in terms of the length of secret key $\mathcal{K}_{k,i}$ and the number of group members N_k.

$N_k = 20$ and $q = 28$, the storage cost for each sensor is merely 56 bits without buffering and 606 bits with buffering. Therefore, the storage cost for each sensor is affordable.

The $\mathcal{MS}$ needs to store $\mathcal{N}$ pairwise key and M group key. The storage cost is $(\mathcal{N} + M)q$. When $\mathcal{N} = 400$, $M = 10$ and $q = 28$, the storage cost is 11480 bits. Thus the storage cost on $\mathcal{MS}$ is negligible.

7.5.3 Communication Cost

Upon updating a group key, the $\mathcal{MS}$ just broadcasts $\mathcal{X}$ to the sensors in its group. Since communication cost is variable depending on the transmission distance between the sensors and the $\mathcal{MS}$, we cannot measure the exact value of communication cost when sensors are mobile. However, it is easy to know that the length of broadcast value is merely $log\mathcal{X}$ bits. Thus we measure the communication cost of our scheme based on $log\mathcal{X}$. Fig. 7.1 shows the length of $\mathcal{X}$ with respect to N_k and q. For example, when $q = 24$ and $N_k = 30$, the length of the broadcast value $\mathcal{X}$ is 705 bits. Hence, the communication cost is affordable.

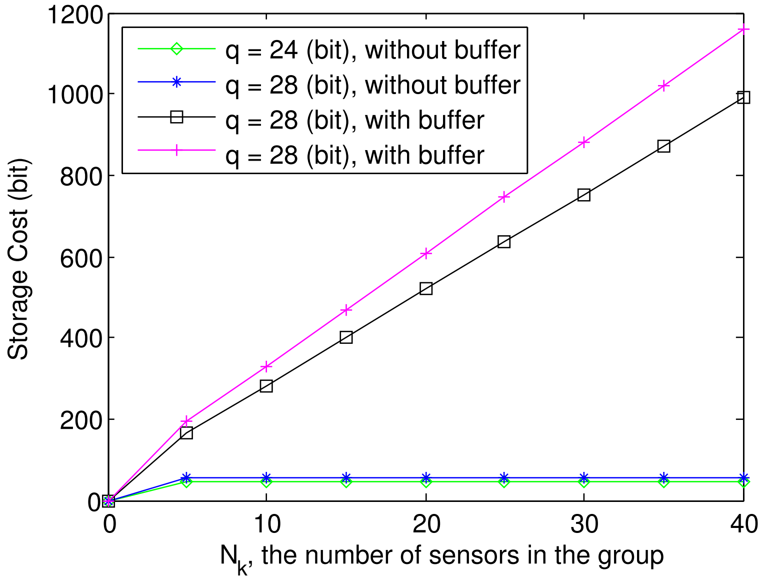


Figure 7.3: Sensor side: The storage cost in terms of the length of secret key $\mathcal{K}_{k,i}$ and the number of group members N_k.

7.6 Summary

In this chapter, we propose SCARKER to provide both forward security and backward security as well as collusion resilience in addition to node capture detection. We demonstrate that SCARKER is lightweight where each sensor only needs to perform one modular operation and one XOR operation for each group key update, significantly reducing the computation overhead of sensors. It is worth mentioning that the time to compute group key and the length of broadcast value (hence the communication cost) are proportional to the number of group members. Thus SCARKER is suitable for distributing secret group key to a group with a small group size. In case that the number of group members is large, we can divide the group into a number of subgroups with suitable size. Then broadcast values are thus constructed for each subgroup using CRT at acceptable expense.

Chapter 8

Conclusions and Future Work

This chapter summaries the contributions of the research work covered by this dissertation and suggests directions for future work.

8.1 Summary of the research and the scientific contributions

In this dissertation, we address emerging security issues in UWSNs considering the fact that a trusted third party does not exist in such networks. We investigate the challenges pertained to those problems, and propose to exploit key evolution as well as (n, m) Reed-Solomon codes based scheme (i.e., ES) to provide forward secrecy, probabilistic backward secrecy and data reliability in absence of reliable nodes and communication channels. Compared with existing schemes [23, 60, 21, 109], the ES does not rely on reliable nodes and communication channels, and can be resilient to message failure and node failure with certain probabilities.

However, as the ES only provides probabilities backward secrecy, $HEHSS$, a homomorphic encryption and homomorphic secret sharing based scheme is proposed to accomplish the goals of achieving forward secrecy, backward secrecy, resilience to node compromises, reliability, and efficient storage and transmission. Moreover, we demonstrate through the $HEHSS$ that backward secrecy of *historical* data can be achieved.

Furthermore, as an enhancement to the ES and the $HEHSS$, a constrained optimization scheme, $ODDS$, is proposed for data distribution to maximize security level of data and optimize data reliability. We show through MATLAB simulations that the optimal algorithm achieves both backward secrecy and data reliability and has significant improvement over the state of art techniques [60, 21, 109].

The *ODDS* is based on an assumption that sensors know their neighbor's security level. We use trustworthiness of sensors to denote their security level and propose an efficient, robust and scalable trust management scheme for UWSNs. Our approach, based on GHT and GPSR, reduces significantly storage cost caused by distributed trust data storage. We demonstrate the effectiveness of the scheme through detailed analysis and simulations using MATLAB.

In addition to intrusion-resilient schemes mentioned above, we also introduce SCARKER, a Sensor Capture Resistance and Key Refreshing scheme, to reduce the consequences when a sensor is captured and compromised. The SCARKER provides both forward security and backward security as well as collusion resilience in addition to node capture detection. We demonstrate that each sensor only needs to perform one modular operation and one XOR operation for each group key update, significantly reducing the computation overhead of sensors. Through implementing SCARKER in actual sensors, the obtained experimental results show that SCARKER is especially efficient and suitable for MUWSNs.

8.2 Future Work

Although many security mechanisms have been studied, there are still a number of open questions in UWSNs which need to be addressed, as outlined below. Refer to [96] for more details.

8.2.1 Long-lived UWSNs

Consider network topology of static sensors with an $\mathcal{MS}$. While there is no static sink available, an $\mathcal{MS}$ visits the static sensors with irregular and even unpredictable frequency. Consequently, each sensor must accumulate sensed data and have the ability to wait *long enough* until a specified signal sent by $\mathcal{MS}$ to offload data onto it. Since the memory size of sensor nodes is limited, no matter how the data are compressed, the memory would be full after *certain* phases. Thus $\mathcal{MS}$ has to access the network to offload data at a reasonable interval. Otherwise, newly collected data (or old data) would be lost. However, in realistic scenarios, $\mathcal{MS}$ may fail to visit the UWSN as planed for any unpredictable reasons (bad weather or blocked by adversaries, for instance). In addition, $\mathcal{MS}$ may be specified to prolong visit time interval for sending an $\mathcal{MS}$ to hostile environments (or unattended areas), something which is highly risky (or costly). Hence, to develop secure, long-lived UWSNs, we need to answer the following questions:

- How to reduce power consumption of security mechanism as little as possible.
- How to compress sensed data in a more efficient way?
- How to design power management scheme to prolong both battery lifetime and network lifetime?

8.2.2 SKC-based Distributed Data Access Control

Access control is a mechanism to prevent unauthorized use of resources, including the prevention of use of a resource in an unauthorized manner [82]. Generally, the data access strategy of WSNs can be divided into two ways, namely access control at the base station or access control at sensor nodes. In traditional WSNs, sensor nodes transmit generated data to a base station that users can access by querying through the base station. Access control strategy is adopted in the base station side rather than in the sensor nodes. Since the base station is not constrained by resources (computation, energy, memory, etc.), many access control policies [78, 77] can be applied. In UWSNs, as a large amount of sensed data are stored in individual sensor nodes, the data storage and access have to be protected by using encryption, so that the data can only be accessed by authorized users with the corresponding keys.

Existing literature [115, 117] addressing distributed data access control policy of UWSNs is based on PKC which is considered to be too expensive to implement in commodity sensors. The authors in [84] have proposed an SKC based distributed data storage and retrieval scheme, where access control policy is provided by sharing the symmetric key with authorized users based on perturbed polynomials [8]. It has, however, not been proven to be secure in [2]. Therefore how to design an SKC-based distributed data access control scheme in UWSNs remains as another open question.

8.2.3 Protecting $\mathcal{MS}$

To the best of our knowledge, currently there is no literature that addresses location privacy of $\mathcal{MS}$ in UWSNs. When an $\mathcal{MS}$ visits the static sensors to collect accumulated data, if the location of the $\mathcal{MS}$ is detected by an adversary, the adversary can easily capture or destroy $\mathcal{MS}$. Since an $\mathcal{MS}$ usually holds the network secret key and has higher privilege than sensors, once an $\mathcal{MS}$ is compromised, secret keys and privileges granted to it will be abused. Therefore, the open questions with this respect include:

- How to protect the location privacy of an $\mathcal{MS}$?
- How to avoid impersonating the behavior of a real $\mathcal{MS}$ in the event that it is captured (or comprised) by an adversary?
- How to design the optimal travel route to collect accumulated sensor data with least security risk to $\mathcal{MS}$?

Bibliography

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks (Elsevier)*, 38(4):393–422, Mar. 2002.

[2] M. Albrecht, C. Gentry, S. Halevi, and J. Katz. Attacking cryptographic schemes based on "perturbation polynomials". In *Proc. ACM Conf. on Computer and Communications Security (CCS '09)*, pages 1–10, Chicago, IL, USA, Nov. 2009.

[3] A. Becher, Z. Benenson, and M. Dornseif. Tampering with motes: real-world physical attacks on wireless sensor networks. In *Proc. Int. Conf. Security in Pervasive Computing (SPC '06)*, pages 104–118, York, UK,, Apr. 2006.

[4] M. Belenkiy. *Disjunctive multi-level secret sharing.* Cryptology ePrint Archive, Report 2008/018, 2008, http://eprint.iacr.org/.

[5] M. Bellare and S. Miner. A forward-secure digital signature scheme. In *Proc. Advances in Cryptology - CRYPTO '99*, pages 786–786, Santa Barbara, CA, USA, Aug. 1999.

[6] M. Bellare and B. Yee. Forward-security in private-key cryptography. In *Proc. CT-RSA '03*, pages 1–18, San Francisco, CA, USA, 2003.

[7] J. Benaloh. Secret sharing homomorphisms: keeping shares of a secret secret. In *Proc. Advances in Cryptology - CRYPTO '86*, pages 251–260, Santa Barbara, CA, USA, Aug. 1986.

[8] C. Blundo, A. De Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung. Perfectly-secure key distribution for dynamic conferences. In *Proc. Advances in Cryptology - CRYPTO '92*, volume 740, pages 471–486, Santa Barbara, CA, USA, Aug. 1992.

[9] E. Bresson, O. Chevassut, and D. Pointcheval. Provably secure authenticated group Diffie-Hellman key exchange. *ACM Trans. Inf. Syst. Secur. (TISSEC)*, 10(3):255–264, Jul. 2007.

[10] E. Brickell and Y. Yacobi. On privacy homomorphisms. In *Proc. Advances in Cryptology - EUROCRYPT '88*, volume 87, pages 117–125, Davos, Switzerland, May 1988.

[11] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proc. ACM MobiCom '98*, pages 85–97, Dallas, TX, USA, Oct. 1998.

[12] S. Buchegger and J. Le Boudec. A robust reputation system for mobile ad-hoc networks. In *Proc. Workshop on Economics of Peer-to-Peer Systems (P2PEcon'04)*, pages 1–11, Harvard University, Jun. 2004.

[13] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas. Multicast security: a taxonomy and some efficient constructions. In *Proc. IEEE INFOCOM '99*, volume 2, pages 708–716, New York, NY, USA, Mar. 1999.

[14] C. Castelluccia, E. Mykletun, and G. Tsudik. Efficient aggregation of encrypted data in wireless sensor networks. In *Proc. ACM/IEEE MobiQuitous '05*, pages 109–117, San Diego, CA, USA, 2005.

[15] C.-W. Chen and Y.-R. Tsai. Location privacy in unattended wireless sensor networks upon the requirement of data survivability. *IEEE J. Sel. Areas Commun. (JSAC)*, 29(7):1480 –1490, Aug. 2011.

[16] J. Cheng and C. Laih. Conference key agreement protocol with non-interactive fault-tolerance over broadcast network. *Int. J. Information Security*, 8(1):37–48, Jan. 2009.

[17] S. Chessa and P. Maestrini. Dependable and secure data storage and retrieval in mobile, wireless networks. In *Proc. IEEE Int. Conf. on Dependable Systems and Networks (DSN '03)*, pages 207–216, San Francisco, CA, USA, 2003.

[18] G. Chiou and W. Chen. Secure broadcasting using the secure lock. *IEEE Trans. Softw. Eng. (TSE)*, 15(8):929–934, Aug. 2002.

[19] M. Conti, R. Di Pietro, L. Mancini, and A. Mei. Emergent properties: detection of the node-capture attack in mobile wireless sensor networks. In *Proc. ACM Conf. on Wireless Network Security (WiSec '08)*, pages 214–219, Alexandria, VA, USA, Mar. 2008.

[20] K. Dantu, M. Rahimi, H. Shah, S. Babel, A. Dhariwal, and G. Sukhatme. Robomote: enabling mobility in sensor networks. In *Proc. IEEE Int.*

Symp. on Information Processing in Sensor Networks (IPSN '05), pages 404 – 409, Los Angeles, CA, USA, Apr. 2005.

[21] R. Di Pietro, D. Ma, C. Soriente, and G. Tsudik. POSH: proactive cooperative self-healing in unattended wireless sensor networks. In *Proc. IEEE Symp. on Reliable Distributed Systems (SRDS '08)*, pages 185–194, Napoli, Italy, Oct. 2008.

[22] R. Di Pietro, L. Mancini, C. Soriente, A. Spognardi, and G. Tsudik. Playing hide-and-seek with a focused mobile adversary in unattended wireless sensor networks. *Ad Hoc Networks (Elsevier)*, 7(8):1463–1475, Nov. 2009.

[23] R. Di Pietro, L. V. Mancini, C. Soriente, A. Spognardi, and G. Tsudik. Catch me (if you can): data survival in unattended sensor networks. In *Proc. IEEE PERCOM '08*, pages 185–194, Hong Kong, Mar. 2008.

[24] R. Di Pietro, L. V. Mancini, C. Soriente, A. Spognardi, and G. Tsudik. Data security in unattended wireless sensor networks. *IEEE Trans. on Computers (TC)*, 58(11):1500–1511, Nov. 2009.

[25] R. Di Pietro, G. Oligeri, C. Soriente, and G. Tsudik. Intrusion-resilience in mobile unattended WSNs. In *Proc. IEEE INFOCOM '10*, pages 1–9, San Diego, CA, USA, Mar. 2010.

[26] R. Di Pietro, G. Oligeri, C. Soriente, and G. Tsudik. Securing mobile unattended WSNs against a mobile adversary. In *Proc. IEEE Int. Symp. on Reliable Distributed Systems (SRDS'10)*, pages 11–20, New Delhi, India, Oct. 2010.

[27] R. Di Pietro, C. Soriente, A. Spognardi, and G. Tsudik. Collaborative authentication in unattended WSNs. In *Proc. ACM Conf. on Wireless Network Security (WiSec '09)*, pages 237–244, Zurich, Switzerland, Mar. 2009.

[28] R. Di Pietro and N. Verde. Epidemic data survivability in unattended wireless sensor networks. In *Proc. ACM Conf. on Wireless Network Security (WiSec '11)*, pages 11–22, Hamburg, Germany, Jun. 2011.

[29] R. Di Pietro and N. Verde. Introducing epidemic models for data survivability in unattended wireless sensor networks. In *Proc. IEEE WoWMoM '11*, pages 1–6, Lucca, Italy, Jun. 2011.

[30] Y. Diao, D. Ganesan, G. Mathur, and P. Shenoy. Rethinking data management for storage-centric sensor networks. In *Proc. Biennial Conf. on Innovative Data Systems Research (CIDR '07)*, Pacific Grove, CA, USA, 2007.

[31] Y. Dodis, M. Franklin, J. Katz, A. Miyaji, and M. Yung. Intrusion-resilient public-key encryption. In *Proc. Topics in Cryptology - CT-RSA '03*, pages 19–32, San Francisco, CA, USA, Apr. 2003.

[32] Y. Dodis, M. Franklin, J. Katz, A. Miyaji, and M. Yung. A generic construction for intrusion-resilient public-key encryption. In *Proc. Topics in Cryptology - CT-RSA '04*, pages 1997–1997, San Francisco, CA, USA, Feb. 2004.

[33] Y. Dodis, J. Katz, S. Xu, and M. Yung. Key-insulated public key cryptosystems. In *Proc. Advances in Cryptology - EUROCRYPT '02*, pages 65–82, Amsterdam, Netherlands, Apr. 2002.

[34] L. Eschenauer and V. D. Gligor. A key-management scheme for distributed sensor networks. In *Proc. ACM Conf. on Computer and Communications Security (CCS '02)*, pages 41–47, Washington, DC, USA, Nov. 2002.

[35] S. Ganeriwal, L. Balzano, and M. Srivastava. Reputation-based framework for high integrity sensor networks. *ACM Trans. Sen. Netw. (TOSN)*, 4(3):1–37, May 2008.

[36] V. Giruka, M. Singhal, J. Royalty, and S. Varanasi. Security in wireless sensor networks. *Wirel. Commun. Mob. Comput. (Wiley)*, 8(1):1–24, Jan. 2008.

[37] H. Harney, C. Muckenhirn, and T. Rivers. Group key management protocol (GKMP) architecture, Jul. 1997. IETF RFC 2094.

[38] H. Harney, C. Muckenhirn, and T. Rivers. Group key management protocol (GKMP) specification, Jul. 1997. IETF RFC 2093.

[39] W. He, X. Liu, H. Nguyen, K. Nahrstedt, and T. T. Abdelzaher. PDA: privacy-preserving data aggregation in wireless sensor networks. In *Proc. IEEE INFOCOM '07*, pages 2045–2053, Anchorage, AL, USA, May 2007.

[40] J.-W. Ho, M. Wright, and S. Das. Fast detection of replica node attacks in mobile sensor networks using sequential analysis. In *Proc. IEEE INFOCOM '09*, pages 1773 –1781, Rio de Janeiro, Brazil, Apr. 2009.

[41] K.-S. Hung, K.-S. Lui, and Y.-K. Kwok. A trust-based geographical routing scheme in sensor networks. In *Proc. IEEE WCNC '07*, pages 3123–3127, Hong Kong, Mar. 2007.

[42] G. Itkis and L. Reyzin. Forward-secure signatures with optimal signing and verifying. In *Proc. Advances in Cryptology - CRYPTO '01*, pages 332–354, Santa Barbara, CA, USA, Aug. 2001.

[43] G. Itkis and L. Reyzin. SiBIR: signer-base intrusion-resilient signatures. In *Proc. Advances in Cryptology - CRYPTO '02*, pages 499–514, Santa Barbara, CA, USA, Aug. 2002.

[44] J. Jaffe and C. Schurgers. Sensor networks of freely drifting autonomous underwater explorers. In *Proc. ACM Int. Workshop on Underwater Networks (WUWNet '06)*, pages 93–96, Los Angeles, CA, USA, Sep. 2006.

[45] A. Josang. An algebra for assessing trust in certification chains. In *Proc. Netw. Distribut. Syst. Secur. Symp. (NDSS '99)*, pages 1–10, San Diego, CA, USA, Feb. 1999.

[46] B. Karp and H. T. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In *Proc. ACM MobiCom '00*, pages 243–254, Boston, MA, USA, Aug. 2000.

[47] J. Katz and M. Yung. Scalable protocols for authenticated group key exchange. In *Proc. Advances in Cryptology - CRYPTO '03*, pages 110–125, Santa Barbara, CA, USA, Aug. 2003.

[48] M. Krasniewski, P. Varadharajan, B. Rabeler, S. Bagchi, and Y. Hu. TIBFIT: trust index based fault tolerance for arbitrary data faults in sensor networks. In *Proc. Int. Conf. Dependable Systems and Networks (DSN '05)*, pages 672–681, Yokohama, Japan, Jun. 2005.

[49] D. Liu and P. Ning. Efficient distribution of key chain commitments for broadcast authentication in distributed sensor networks. In *Proc. Netw. Distribut. Syst. Secur. Symp. (NDSS '03)*, volume 276, San Diego, CA, USA, Feb. 2003.

[50] D. Liu and P. Ning. Establishing pairwise keys in distributed sensor networks. In *Proc. ACM Conf. on Computer and Communications Security (CCS '03)*, pages 52–61, Washington, DC, USA, Oct. 2003.

[51] D. Liu and P. Ning. Multilevel μTESLA: broadcast authentication for distributed sensor networks. *ACM Trans. Embed. Comput. Syst. (TECS)*, 3(4):800–836, Nov. 2004.

[52] D. Liu, P. Ning, and W. Du. Group-based key predistribution for wireless sensor networks. *ACM Trans. Sen. Netw. (TOSN)*, 4:1–30, Apr. 2008.

[53] Z. Liu, J. Ma, Y. Park, and S. Xiang. Data security in unattended wireless sensor networks with mobile sinks. *Wirel. Commun. Mob. Comput. (Wiley)*, pages 1–16, Mar. 2011.

[54] J. Lopez, R. Roman, and C. Alcaraz. Analysis of security threats, requirements, technologies and standards in wireless sensor networks. *Foundations of Security Analysis and Design V*, 5705:289–338, Aug. 2009.

[55] W. Lou, W. Liu, and Y. Fang. SPREAD: enhancing data confidentiality in mobile ad hoc networks. In *Proc. IEEE INFOCOM '04*, pages 2404–2413, Hong Kong, Mar. 2004.

[56] J. Luo, D. Wang, and Q. Zhang. Double mobility: coverage of the sea surface with mobile sensor networks. In *Proc. IEEE INFOCOM '09*, pages 118 –126, Rio de Janeiro, Brazil, Apr. 2009.

[57] D. Ma. Practical forward secure sequential aggregate signatures. In *Proc. ACM Symp. on Information, Computer and Communications Security (ASIACCS '08)*, pages 341–352, Tokyo, Japan, Mar. 2008.

[58] D. Ma, C. Soriente, and G. Tsudik. New adversary and new threats: security in unattended sensor networks. *IEEE Netw.*, 23(2):43–48, Mar. 2009.

[59] D. Ma and G. Tsudik. Forward-secure sequential aggregate authentication. In *Proc. IEEE Symp. on Security and Privacy (S&P '07)*, pages 86–91, Oakland, CA, USA, May. 2007.

[60] D. Ma and G. Tsudik. DISH: Distributed Self-Healing. In *Proc. 10th Int. Symp. on Stabilization, Safety, and Security of Distributed Systems (SSS '08)*, pages 47–62, Detroit, MI, USA, Nov. 2008.

[61] $Maplesoft^{TM}$. Maple 13.

[62] G. Mathur, P. Desnoyers, D. Ganesan, and P. Shenoy. Capsule: an energy-optimized object storage system for memory-constrained sensor devices. In *Proc. ACM SenSys '06*, pages 195–208, Boulder, CO, USA, Oct.-Nov. 2006.

[63] G. Mathur, P. Desnoyers, D. Ganesan, and P. Shenoy. Ultra-low power data storage for sensor networks. In *Proc. ACM IPSN '06*, pages 374–381, Nashville, TN, USA, Apr. 2006.

[64] A. Mitra, A. Banerjee, W. Najjar, D. Zeinalipour-Yazti, D. Gunopulos, and V. Kalogeraki. High performance, low power sensor platforms featuring gigabyte scale storage. In *Proc. ACM/IEEE Mobiquitous '05*, pages 1–10, San Diego, CA, USA, Jul. 2005.

[65] V. Naik, A. Arora, S. Bapat, and M. Gouda. Whisper: local secret maintenance in sensor networks. *IEEE Distributed Systems Online*, 4:1–6, 2003.

[66] V. Oleshchuk and V. Zadorozhny. Trust-aware query processing in data intensive sensor networks. In *Proc. Int. Conf. on Sensor Technologies and Applications (SensorComm '07)*, pages 176–180, Valencia, Spain, Oct. 2007.

[67] M. Omar, Y. Challal, and A. Bouabdallah. Reliable and fully distributed trust model for mobile ad hoc networks. *Elsevier Comput. & Secur.*, 28(3-4):199–214, May 2009.

[68] A. Penrig, D. Song, and D. Tygar. ELK, a new protocol for efficient large-group key distribution. In *Proc. IEEE Security Privacy (S&P '01)*, pages 247 –262, Oakland, CA , USA, May 2001.

[69] A. Perrig, J. Stankovic, and D. Wagner. Security in wireless sensor networks. *Communications of the ACM*, 47(6):53–57, Jun. 2004.

[70] A. Perrig, R. Szewczyk, J. Tygar, V. Wen, and D. Culler. SPINS: security protocols for sensor networks. *Wireless Networks*, 8(5):521–534, Sep. 2002.

[71] B. Przydatek, D. Song, and A. Perrig. SIA: secure information aggregation in sensor networks. In *Proc. ACM SenSys '03*, pages 255–265, Los Angeles, CA, USA, Nov. 2003.

[72] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker. GHT: a geographic hash table for data-centric storage. In *Proc. ACM Int. Workshop on Wireless Sensor Networks and Applications (WSNA '02)*, pages 78–87, Atlanta, GA, USA, Sep. 2002.

[73] M. Raya, P. Papadimitratos, V. Gligor, and J.-P. Hubaux. On data-centric trust establishment in ephemeral ad hoc networks. In *Proc. IEEE INFOCOM '08*, pages 1238 –1246, Phoenix, AZ, USA, Apr. 2008.

[74] I. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics*, 8(2):300–304, 1960.

[75] M. T. Refaei, L. A. DaSilva, M. Eltoweissy, and T. Nadeem. Adaptation of reputation management systems to dynamic network conditions in ad hoc networks. *IEEE Trans. Comput. (TC)*, 59(5):707–719, May 2010.

[76] A. Rezgui and M. Eltoweissy. TARP: a trust-aware routing protocol for sensor-actuator networks. In *Proc. IEEE Int. Conf. Mobile Adhoc and Sensor Systems (MASS '07)*, pages 1 –9, Pisa, Italy, Oct. 2007.

[77] R. Sandhu, E. Coyne, H. Feinstein, and C. Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, Aug. 1996.

[78] R. Sandhu and P. Samarati. Access control: principle and practice. *IEEE Commun. Mag.*, 32(9):40 –48, 1994.

[79] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[80] A. Sherman and D. McGrew. Key establishment in large dynamic groups using one-way function trees. *IEEE Trans. Softw. Eng. (TSE)*, 29(5):444–458, May 2003.

[81] E. Shi and A. Perrig. Designing secure sensor networks. *IEEE Wireless Commun.*, 11(6):38–43, Dec. 2004.

[82] W. Stallings, L. Brown, M. Bauer, and M. Howard. *Computer security: principles and practice.* Pearson Prentice Hall, 2008.

[83] M. Steiner, G. Tsudik, and M. Waidner. Diffie-Hellman key distribution extended to group communication. In *Proc. ACM Conf. on Computer and Communications Security (CCS '96)*, pages 31–37, New Delhi, India, Mar. 1996.

[84] N. Subramanian, C. Yang, and W. Zhang. Securing distributed data storage and retrieval in sensor networks. In *Proc. IEEE PERCOM '07*, volume 3, pages 659 – 676, NYC, USA, Mar. 2007.

[85] Y. L. Sun, W. Yu, Z. Han, and K. Liu. Information theoretic framework of trust modeling and evaluation for ad hoc networks. *IEEE J. Sel. Areas Commun. (JSAC)*, 24(2):305–317, Feb. 2006.

[86] Sun Microsystems, Inc. Sun SPOT.

[87] S. Tanachaiwiwat, P. Dave, R. Bhindwale, and A. Helmy. Location-centric isolation of misbehavior and trust routing in energy-constrained sensor networks. In *Proc. IEEE Int. Conf. Perform. Comput. Commun. (IPCCC '04)*, pages 463–469, Phoenix, AZ, USA, Apr. 2004.

[88] **Yi Ren**. Unattended wireless sensor networks @ Simu, 2010. http://sourceforge.net/projects/uwsn.

[89] **Yi Ren**, V. A. Oleshchuk, and F. Y. Li. Optimized secure and reliable distributed data storage scheme and performance evaluation in Unattended WSNs. *Elsevier Computer Communications (COMCOM), under review.*

[90] **Yi Ren**, V. A. Oleshchuk, and F. Y. Li. A distributed data storage and retrieval scheme in unattended WSNs using homomorphic encryption and secret sharing. In *Proc. IFIP Wireless Days (WD '09)*, Paris, France, Dec. 2009.

[91] **Yi Ren**, V. A. Oleshchuk, and F. Y. Li. A spatial role-based authorization framework for sensor network-assisted indoor WLANs. In *Proc. Int. Conf. on Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology (Wireless VITAE '09)*, pages 782–787, Aalborg Danmark, May 2009.

[92] **Yi Ren**, V. A. Oleshchuk, and F. Y. Li. Location and role based authorization framework for sensor networks assisted indoor WLAN. In *Proc. Norwegian Information Security Conference (NISK '08)*, Kristiansand , Norway, Nov. 2009.

[93] **Yi Ren**, V. A. Oleshchuk, and F. Y. Li. Secure and efficient data storage in unattended wireless sensor networks. In *Proc. IFIP Int. Conf. on New Technologies, Mobility and Security (NTMS '09)*, Cairo, Egypt, Dec. 2009.

[94] **Yi Ren**, V. A. Oleshchuk, and F. Y. Li. A scheme for secure and reliable distributed data storage in unattended WSNs. In *Proc. IEEE Global Communications Conference (GLOBECOM '10)*, Miami, FL, USA, Dec. 2010.

[95] **Yi Ren**, V. A. Oleshchuk, and F. Y. Li. An efficient Chinese remainder theorem based node capture resilience scheme for mobile WSNs. In *Proc. IEEE Int. Conf. Information Theory and Information Security (ICITIS '10)*, Beijing, China, Dec. 2010.

[96] **Yi Ren**, V. A. Oleshchuk, F. Y. Li, and X. Ge. Security in mobile wireless sensor networks - a survey. *Journal of Communications (JCM)*, 6(2):128–142, Apr. 2011.

[97] **Yi Ren**, V. A. Oleshchuk, F. Y. Li, and S. Sulistyo. SCARKER: a sensor capture resistance and key refreshing scheme for mobile WSNs. In *Proc. IEEE Local Computer Networks (LCN '11)*, Bonn, Germany, Oct. 2011.

[98] **Yi Ren**, V. A. Oleshchuk, F. Y. Li, and S. Sulistyo. FoSBaS: an efficient key management scheme for body area networks. In *Proc. IEEE Wireless Communications and Networking Conference (WCNC '12)*, Paris, France, Apr. 2012.

[99] **Yi Ren**, W. Ren, R. W. Fensli, V. A. Oleshchuk, and F. Y. Li. RoFa: a robust and flexible fine-grained access control scheme in cloud-IOT paradigm for remote mobile medical monitoring. *Springer Journal of Wireless Personal Communications, under review.*

[100] **Yi Ren**, V. I. Zadorozhny, V. A. Oleshchuk, and F. Y. Li. A novel approach to trust management in unattended wireless sensor networks. *IEEE Trans. Mobile Computing, under review.*

[101] **Yi Ren**, V. I. Zadorozhny, V. A. Oleshchuk, and F. Y. Li. An efficient, robust and scalable trust management scheme for unattended wireless sensor networks. In *Proc. IEEE Int. Conf. Mobile Data Management (MDM '12)*, Bengaluru, India, Jul. 2012.

[102] G. Theodorakopoulos and J. Baras. On trust models and trust evaluation metrics for ad hoc networks. *IEEE J. Sel. Areas Commun. (JSAC)*, 24(2):318–328, Feb. 2006.

[103] G. Tsudik. Confronting a mobile adversary in unattended sensor networks. In *Proc. ACM Symp. on Information, Computer and Communications Security (ASIACCS '08)*, pages 1–1, Tokyo, Japan, Mar. 2008.

[104] W. Tzeng. A secure fault-tolerant conference-key agreement protocol. *IEEE Trans. Computers (TC)*, 51(4):373–379, Apr. 2002.

[105] L. Vieira, U. Lee, and M. Gerla. Phero-trail: a bio-inspired location service for mobile underwater sensor networks. *IEEE J. Sel. Areas Commun. (JSAC)*, 28(4):553 –563, May 2010.

[106] M. Waldvogel, G. Caronni, D. Sun, N. Weiler, and B. Plattner. The versakey framework: Versatile group key management. *IEEE J. Sel. Areas Commun. (JSAC)*, 17(9):1614–1631, Aug. 2002.

[107] D. Wallner, E. Harder, and R. Agee. Key management for multicast: issues and architectures, 1999. IETF RFC 2627.

[108] K. Walsh and E. G. Sirer. Experience with an object reputation system for peer-to-peer filesharing. In *Proc. USENIX NSDI '06*, pages 1–14, Berkeley, CA, USA, May 2006.

[109] Q. Wang, K. Ren, W. Lou, and Y. Zhang. Dependable and secure sensor data storage with dynamic integrity assurance. In *Proc. IEEE INFOCOM '09*, pages 954–962, Rio de Janeiro, Brazil, Apr. 2009.

[110] C. Wong, M. Gouda, and S. Lam. Secure group communications using key graphs. *IEEE/ACM Trans. Networking (TON)*, 8(1):16–30, Feb. 2000.

[111] L. Xiong and L. Liu. PeerTrust: supporting reputation-based trust for peer-to-peer electronic communities. *IEEE Trans. Knowl. Data Eng. (TKDE)*, 16(7):843–857, Jul. 2004.

[112] X. Xu, J. Luo, and Q. Zhang. Delay tolerant event collection in sensor networks with mobile sink. In *Proc. IEEE INFOCOM '10*, pages 1 –9, San Diego, CA, USA, Mar. 2010.

[113] K. Yadav and A. Srinivasan. iTrust: an integrated trust framework for wireless sensor networks. In *Proc. ACM Symp. on Applied Computing (SAC '10)*, pages 1466–1471, Sierre, Switzerland, Mar. 2010.

[114] A. Yavuz and P. Ning. Hash-based sequential aggregate and forward secure signature for unattended wireless sensor networks. In *Proc. MobiQuitous '09*, pages 1 –10, Toronto, Canada, Jul. 2009.

[115] S. Yu, K. Ren, and W. Lou. FDAC: toward fine-grained distributed data access control in wireless sensor networks. In *Proc. IEEE INFOCOM '09*, pages 963–971, Rio de Janeiro, Brazil, Apr. 2009.

[116] D. Zeinalipour-Yazti, V. Kalogeraki, D. Gunopulos, A. Mitra, A. Banerjee, and W. Najjar. Towards in-situ data storage in sensor databases. *Advances in Informatics*, 3746:36–46, 2005.

[117] R. Zhang, Y. Zhang, and K. Ren. DP^2AC: Distributed Privacy-Preserving Access Control in sensor networks. In *Proc. IEEE INFOCOM '09*, pages 1251–1259, Rio de Janeiro, Brazil, Apr. 2009.

[118] X. Zheng, C.-T. Huang, and M. Matthews. Chinese remainder theorem based group key management. In *Proc. ACM Southeast Conf. (ACM-SE '07)*, pages 266–271, Winston-Salem, NC, USA, Mar. 2007.

[119] R. Zhou and K. Hwang. PowerTrust: a robust and scalable reputation system for trusted peer-to-peer computing. *IEEE Trans. Parallel Distribut. Syst. (TPDS)*, 18(4):460–473, Apr. 2007.

[120] Y. Zhou, Y. Fang, and Y. Zhang. Securing wireless sensor networks: a survey. *IEEE Commun. Surveys Tutorial*, 10(3):6–28, Sep. 2008.

[121] S. Zhu, S. Setia, and S. Jajodia. LEAP+: Efficient security mechanisms for large-scale distributed sensor networks. *ACM Tran. Sen. Netw. (TOSN)*, 2(4):500–528, Nov. 2006.