A SECURE COMMUNICATION FRAMEWORK FOR WIRELESS SENSOR NETWORKS

A Thesis
Presented to
The Academic Faculty

by

Arif Selçuk Uluağaç

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
August 2010

A SECURE COMMUNICATION FRAMEWORK FOR WIRELESS SENSOR NETWORKS

Approved by:

Professor John A. Copeland, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Raheem Beyah, Co-Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Henry L. Owen III
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Geoffrey Ye Li
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Fumin Zhang
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Ellen W. Zegura
School of Computer Science
Georgia Institute of Technology

Date Approved: 8 June 2010

DEDICATION

To my family, my friends, and all mothers in the world..

ACKNOWLEDGEMENTS

I finished this thesis under the supervisions of Dr. John A. Copeland and Dr. Raheem A. Beyah while I was a a proud member of the Communications Systems Center (CSC) Lab in the School of ECE at Georgia Institute of Technology. Drs. Copeland and Beyah have been a perfect model for me and provided me with a perfect setting to flourish my creativity and imagination. They have generously and patiently supported me in all my activities during my PhD. I wholeheartedly thank both of them for their invaluable support, strong guidance, friendship, and trust throughout my Ph.D.

I would like to express my cordial thanks to Dr. Owen for reading this thesis and providing me with valuable and timely feedback. My thanks also extend to the other committee members, Dr. Geoffrey Ye Li, Dr. Ellen W. Zegura, and Dr. Fumin Zhang, for being on my dissertation defense committee. I also thank Dr. Akyildiz, who helped me open the door to a wonderful institution like Georgia Tech. Similarly, my former advisor, Prof. Jon M. Peha, at Carnegie Mellon University (CMU) and Prof. Peter Steenkiste of CMU deserve special thanks.

Over the last 2 years, I have taught courses in the department of Electrical and Computer Engineering Technology at Southern Polytechnic State University as an adjunct instructor. I would like to thank Dr. Tom Fallon, Dr. Austin Asgill, Dr. Walt Thain, and Prof. Scott Larish for their enormous trust and support in my activities there and for providing me with a tranquil teaching environment. This teaching opportunity was a unique experience for me.

As a graduate teaching assistant (GTA) at Georgia Tech, I had a chance to work closely with Dr. Douglas Williams and Dr. Robert Butera. I learned a lot from them about teaching and education fields through this GTA experience. I also thank them

both for their trust in me.

Also, throughout my Ph.D education at Georgia Tech, I met many dedicated, knowledgeable, and helpful faculty and staff members. I thank Dr. Seymour Goodman of the School of Computer Science for his interesting class. I thank Dr. George Riley for his help in GTNetS simulator. I thank Jill Auerbach for the opportunity to be a mentor in the Opportunity Scholarship program in the School of ECE. I specially thank Gail Palmer of the School of ECE for many valuable discussions with her in and outside the classroom environment. I also thank Sheila Schulte of the Office of International Education for always being an accessible resource and for her friendly conversations. Also, among the ECE staff members I relied on, Kathy Cheek and Marilouise Mycko deserve special mention. They were always ready for us.

Most of my life, almost 21 years as of this writing, I have been away from my home. I learned a lot from my friends. They have supported me in many different ways and potentials. They constitute a very big circle-of-trust in my life. During my stay at Georgia Tech, I have had the opportunity to enlarge this circle. I acknowledge this special circle here. My special thanks go to Senay Solak, Mehmet Gumus, Yonca Toker, Ozgur Kaya, Alper Ozalp, Engin Uzuncaova, Umut Ovali, Serkan/Aysim Carlioglu, Ozden Acar, Selcuk Akcay, Mustafa Temur, Serkan Bolat, Teoman Tepehan, Muhtesem Hakki Onder, Murat Uzman, Mehmet Bakkaloglu, Ziya Koksal, Cihan Cobanoglu, Nihal/Cemil Yucer, James Malcom, Mehmet Can Vuran, Vehbi Cagri Gungor, Nitya Sundareswaran, Bharat Jayakumar, Fatma Karzan, Orhan Karzan, Moazzam Khan, Safayet Ahmed, Kevin Fairbanks, Chris Lee, Bongkyoung Kwon, Elmoustapha Ould Ahmed Vall, Mert Cevik, Bahadir Polat, Tarik Guetarni, Myounghwan Lee, Michael Nowatkowski, Jackie Wold, Virosius Putra, Ying Xia, Sung Jin Park, Chang Shi-in, Nevin Altunyurt, Chaoting Xuan, Can Envarli, Sibel Yaman, Manoj Deshpande, Erdem Coskun, Adrian Moore-Pleasant, Melanie Brisse, Apurva Mohan, Ramya Srinivasan, Roma Kane, Siddharth Joshi, Swapnil Shinde,

Sean Roberts.

I graduated from the Turkish Naval High School and the Turkish Naval Academy. Here, I also would like to acknowlege the Turkish Naval Forces (Deniz Kuvvetleri Komutanligi) for providing me with world-class excellent education and training opportunities in these institutions.

Last but not least, special thanks and gratitude to my family, my dad, Selim Uluagac, my mom Nurcan Uluagac, my brothers Alpaslan Uluagac, Bahadir Uluagac, my big-brother Ibrahim Oktay, and my big-sister Sidika Oktay for their constant support, love, encouragement, and sacrifices. There is no way I could finish my PhD work without the unconditional solid support I get from them. I love you, all!

In short, I was very lucky to be bestowed with the freedom provided by my advisors, the unique opportunities in and around Georgia Tech, the love of my family, and the support of my friends. I also think I just was at the right place at the right time. Nonetheless, I am solely responsible for the shortcomings of this thesis as to some extent this work is the artifact of an imagination like many others :)

"But it was just my imagination, running away with me."
from *The Temptations - Just My Imagination (Running Away With Me), 1971*

"This is it!"

Arif Selçuk Uluağaç

June 8, 2010, Atlanta, GA, USA

TABLE OF CONTENTS

DEDICATION iii

ACKNOWLEDGEMENTS iv

LIST OF TABLES ix

LIST OF FIGURES x

SUMMARY xii

I INTRODUCTION 1

1.1 Research Objectives and Solutions 3

1.2 Thesis Outline 6

II DESIGNING SECURE PROTOCOLS FOR WIRELESS SENSOR NETWORKS 7

2.1 Motivation 7

2.2 Related Work 9

2.3 The WSN Communication & Threat Models 10

2.4 Desired Security Services from the WSNs Perspective 11

2.5 When to Employ Specific Security Services 21

III VEBEK: ***V***IRTUAL ***E***NERGY-***B***ASED ***E***NCRYPTION AND ***K***EYING FOR WIRELESS SENSOR NETWORKS 25

3.1 Motivation 26

3.2 Related Work 28

3.3 An Analysis of the Rekeying Cost for WSNs 30

3.4 Semantics of VEBEK 33

3.5 Operational Modes of VEBEK 45

3.6 Performance Analysis 49

3.7 Benefits and Limitations of VEBEK 63

IV TIME-BASED DYNAMIC KEYING AND EN-ROUTE FILTERING (TICK) FOR WIRELESS SENSOR NETWORKS 66

4.1 Motivation 66
4.2 Related Work 68
4.3 Overview of TICK 69
4.4 Time Uncertainty 75
4.5 Performance Evaluation 80

V SECURE SOURCE-BASED LOOSE TIME SYNCHRONIZATION (SOBAS) FOR WIRELESS SENSOR NETWORKS 86
5.1 Motivation 86
5.2 Related Work 89
5.3 Protocol Architecture 92
5.4 Performance Evaluation 99
5.5 Impact of Selective Re-Encoding 108
5.6 Benefits and Limitations 110

VI CONCLUSIONS AND FUTURE WORK 112
6.1 Research Contributions 113
6.2 Future Research Directions 115

APPENDIX A RC4 STREAM CIPHER 118

REFERENCES 120

VITA 127

INDEX 129

LIST OF TABLES

1 Notations Used in VEBEK 33

2 VEBEK Example Encoding Operations 40

3 VEBEK General Simulation Parameters 50

4 VEBEK Energy Related Parameters 50

5 TICK Simulation Parameters 82

6 SOBAS Simulation Parameters 104

LIST OF FIGURES

1 WSNs communication model 10

2 WSNs threat model including the new Target-Based Attacks 11

3 Keying cost of dynamic key-based schemes based on $E[nh]$ vs. VEBEK. 32

4 Modular structure of the VEBEK framework. 34

5 An illustration of the watching concept with forwarding. 37

6 An illustration of the use of RC4 encryption mechanism in VEBEK. . 39

7 Illustration of a sample encoding operation. 40

8 Illustration of the watching concept with forwarding when there is a packet loss. 44

9 Operational Modes. 46

10 VEBEK simulation topology with GTSNetS. 49

11 Theoretical and simulation results with varying number of watched nodes. 51

12 Comparison of filtering efficiency for VEBEK-I and VEBEK-II with varying number of malicious nodes. 52

13 Computation costs under AS-1. 53

14 Transmissions costs under AS-1. 54

15 Total energy under AS-1. 55

16 Computation costs under AS-2. 56

17 Transmissions costs under AS-2. 57

18 Total energy under AS-2. 58

19 DEF. 59

20 SEF. 60

21 STEF. 61

22 Comparison of VEBEK [4], DEF[1], SEF[2], and STEF[3]. 62

23 Synchronization ratio of nodes along the path to the sink. 63

24 TICK Modules 69

25 TICK packet structure. 71
26 An illustration of packet delivery path. 72
27 TICK uncertainity parameters. 75
28 CDF of finding a correct key. 77
29 T_w (Tick window) versus ϕ (tick). 79
30 Comparison of TICK, VEBEK, DEF, SEF, STEF, and TPSKD. . . . 81
31 TICK simulation topology with GTSNetS. 82
32 Computation, transmission, and total energy consumption under an attack scenario. 83
33 Avg. number of key-trials. 84
34 SOBAS's modular protocol architecture 92
35 Synchronization of nodes on more than one synch path in SOBAS. . . 96
36 Summary of operational modes in SOBAS: Hi-Comp: High computation; Lo-Comp: Low computation; E2E-Sync: End-to-end loose synchronization; P-Sync: Loose Path synchronization. 97
37 Illustrations of SRCS and TPSKD 98
38 Illustrations of SOBAS and STM 99
39 Comparison of SOBAS, STM, SRCS, and TPSKD. 102
40 SOBAS simulation topology with GTSNetS. 103
41 Computation cost under an attack scenario. 105
42 Transmissions cost under an attack scenario. 106
43 Total energy consumption under an attack scenario. 107
44 SOBAS Avg. key trials. 108
45 Stateless SOBAS FPR analysis under an attack scenario. 109
46 Stateful SOBAS FPR analysis under an attack scenario. 110

SUMMARY

Today, wireless sensor networks (WSNs) are no longer a nascent technology and future networks, especially Cyber-Physical Systems (CPS) [5] will integrate more sensor-based systems into a variety of application scenarios. Typical application areas include medical, environmental, military, and commercial enterprises. Providing security to this diverse set of sensor-based applications is necessary for the healthy operations of the overall system because untrusted entities may target the proper functioning of applications and disturb the critical decision-making processes by injecting false information into the network. One way to address this issue is to employ *en-route-filtering*-based solutions utilizing keys generated by either *static* or *dynamic* key management schemes in the WSN literature. However, current schemes are complicated for resource-constrained sensors as they utilize many keys and more importantly as they transmit many keying messages in the network, which increases the energy consumption of WSNs that are already severely limited in the technical capabilities and resources (i.e., power, computational capacities, and memory) available to them [6].

Nonetheless, further improvements without too much overhead are still possible by *sharing a dynamically created cryptic credential.* Building upon this idea, the purpose of this thesis is to introduce an efficient and secure communication framework for WSNs. Specifically, three protocols are suggested as contributions using virtual energies and local times onboard the sensors as dynamic cryptic credentials: (1) Virtual Energy-Based Encryption and Keying (VEBEK) [4]; (2) TIme-Based DynamiC Keying and En-Route Filtering (TICK) [7]; (3) Secure Source-Based Loose Time Synchronization (SOBAS) for WSNs [8].

CHAPTER I

INTRODUCTION

Wireless sensors are small wireless devices that communicate with each other in an ad-hoc manner over wireless channels, forming Wireless Sensor Networks (WSNs). Throughout the last decade, their introduction to the networking field has rapidly attracted the attention of academia and industry. Thus, today, WSNs are no longer a nascent technology and future networks, especially Cyber-Physical Systems (CPS) [5], will require the integration of more sensor-based systems into a variety of application scenarios such as in military operations, medical systems, aerospace systems, transportation vehicles and intelligent highways, robotic systems, process control, factory automation, building and environmental control, and smart spaces [9].

Securing the diverse set of sensor-based applications is necessary for the healthy operations of the overall system because adversaries may target the proper functioning of applications and disturb the critical decision-making processes by injecting false information into the network. For instance, it is very important to provide authentic and accurate data to surrounding sensor nodes and to the sink to trigger time-critical responses (e.g., troop movement, evacuation, and first response deployment.) Therefore, protocols should be resilient against false data injected into the network by malicious entities. However, securing sensor networks poses unique challenges to protocol builders because these tiny wireless devices are deployed in large numbers, usually in unattended environments, and are severely limited in the technical capabilities and resources (e.g., power, computational capacities, and memory) available to them [6].

One way to eliminate injected malicious data from WSNs is to utilize an *en-route-filtering* scheme as in [1, 2, 3]. The en-route-filtering schemes generally utilize keys generated by either *static* [10] and *dynamic* [11] key management schemes [6]. In static key management schemes, key management functions (i.e., key generation and distribution) are handled statically. The sensors have a fixed number of keys loaded either prior to or shortly after network deployment. Thus, with static management, once an attacker compromises a sensor node, the attacker will have access to all the keys in the network. On the other hand, dynamic key management schemes perform keying functions either periodically or on demand as needed by the network. The sensors dynamically exchange keys to communicate. Although dynamic schemes provide more attack-resillient services to WSNs, one significant disadvantage with them is that they increase the communication overhead due to keys refreshed or redistributed from time to time in the network [12]. Moreover, the common observation with current en-route-filtering schemes [1, 2, 3] are as follows: (1) They are complicated for resource-constrained sensors as they utilize many keys; (2) they transmit many keying messages in the network, which increases the energy consumption of WSNs; (3) they have not been designed to handle dire communication scenarios; and (4) the energy cost, especially the communication cost, associated with the operations of the protocols are often not discussed by researchers when building secure WSN protocols.

Nonetheless, further improvements for dynamic key management and en-route-filtering schemes without too much overhead are still possible by *sharing a dynamically created cryptic credential.* Specifically, depending on a unique piece of information that a sensor posseses, keys can be generated instead of being exchanged. For this, the residual battery life or energy on a node [13], virtual energies, local time in the node, or identity of the node could be utilized as the shared dynamic cryptic credential and would be updated appropriately as necessary.

1.1 Research Objectives and Solutions

The objective of this thesis is to develop efficient and secure communication frameworks for WSN applications by building upon the idea of sharing a dynamic cryptic credential. More specifically, motivated with the downsides of current dynamic key management and en-route-filtering schemes and the fact that the communication cost is the most dominant factor in a sensor's energy consumption [14, 15], in this thesis, the problem of providing security to sensor-based applications is tackled with a new approach. As opposed to other "chatty" dynamic key management and en-route filtering schemes, we focus on eliminating specific control messages for keying or rekeying in the network so that some of the energy savings from transmission cost can be utilized for the computation of local security operations. Specifically, the following four areas are investigated under this thesis and each of them is described in the following subsections:

- Designing Secure Protocols for Wireless Sensor Networks
- Virtual Energy-Based Encryption and Keying (VEBEK) protocol for Wireless Sensor Networks
- TIme-Based Dynamic Keying and En-Route Filtering (TICK) for Wireless Sensor Networks
- Secure SOurce-BAsed Loose Time Synchronization (SOBAS) for Wireless Sensor Networks

1.1.1 Designing Secure Protocols for Wireless Sensor Networks

Over the years, a myriad of protocols have been proposed for resource-limited Wireless Sensor Networks (WSNs). Similarly, security research for WSNs has also evolved over the years. Although fundamental notions of WSN research are well established, optimization of the limited resources has motivated new research directions in the

field. In this research, we present general principles to aid in the design of secure WSN protocols [6]. Therefore, building upon both the established and the new concepts, envisioned applications, and the experience garnered from the WSNs research, we first review the desired security services (i.e., confidentiality, authentication, integrity, access control, availability, and nonrepudiation) from WSNs perspective. Then, we question which services would be necessary for resource-constrained WSNs and when it would be most reasonable to implement them for a WSN application.

1.1.2 Virtual Energy-Based Encryption and Keying (VEBEK) protocol for Wireless Sensor Networks

Since the communication cost is the most dominant factor in a sensor's energy consumption, in this work, we introduce an energy-efficient Virtual Energy-Based Encryption and Keying (VEBEK) scheme [4] for WSNs that significantly reduces the number of transmissions needed for rekeying to avoid stale keys. It is the first protocol in this thesis that is based on the idea of sharing a dynamic cryptic credential. VEBEK is a secure communication framework where sensed data is encoded using a scheme based on a permutation code generated via the RC4 encryption mechanism. The key to the RC4 encryption mechanism dynamically changes as a function of the residual virtual energy of the sensor. Thus, a one-time dynamic key is employed for one packet only and different keys are used for the successive packets of the stream. The intermediate nodes along the path to the sink are able to verify the authenticity and integrity of the incoming packets using a predicted value of the key generated by the sender's virtual energy, thus requiring no need for specific rekeying messages. VEBEK is able to efficiently detect and filter false data injected into the network by malicious outsiders. We have evaluated VEBEK's feasibility and performance analytically and through simulations. Our results show that VEBEK, without incurring transmission overhead (increasing packet size or sending control messages for rekeying), is able to eliminate malicious data from the network in an energy efficient

manner. We also show that our framework performs better than other comparable schemes in the literature with an overall 60%−100% improvement in energy savings without the assumption of a reliable medium access control (MAC) layer.

1.1.3 TIme-Based Dynamic Keying and En-Route Filtering (TICK) for Wireless Sensor Networks

In this thesis, the TIme-Based DynamiC Keying and En-Route Filtering (TICK) protocol [7] for WSNs is the second application based on the idea of sharing a dynamic cryptic credential. As opposed to current chatty schemes that incur regular keying message overhead, it avoids the transmission of explicit keying messages needed to avoid stale keys. Nodes intelligently use their local time values as a one-time dynamic key to encrypt each message. TICK secures events as they occur and provides a mechanism to prevent malicious nodes from injecting false packets into the network. TICK is as a worst case two times more energy efficient than existing related work. Both an analytical framework and simulation results are presented to verify the feasibility of TICK as well as the energy consumption of the scheme under normal operation and attack from malicious nodes.

1.1.4 Secure SOurce-BAsed Loose Time Synchronization (SOBAS) for Wireless Sensor Networks

Many applications that will utilize WSNs will require that event reports extracted from the network are received in the order that they were sensed. One effective way to achieve this goal is to time-stamp the messages/reports using clocks available onboard the sensors. However, effects of the physical environment that cause drift in the clocks of sensors has necessitated the need for synchronization protocols for WSNs. Further, given that these WSNs may be deployed in hostile regions, it is imperative that this synchronization process is secured against malicious intruders. To address this concern, in this research, the Secure SOurce-BASed Loose-Time Synchronization (SOBAS) protocol [8] is developed for WSNs. Essentially, SOBAS is a derivative of

the TICK protocol and supports WSN applications that do not need perfect synchronization. SOBAS is used to securely synchronize the data path in the network, *without* the transmission of explicit synchronization control messages. Instead of synchronizing each sensor globally as opposed to approaches providing perfect synchronization, we focus on ensuring that each source node is synchronized with the sink and nodes along the data delivery path such that event reports generated by the sink are ordered properly. As in TICK, nodes use their local time values as a one-time dynamic key to encrypt each message. Once the message arrives at the sink (or at the next sensor), it attempts to *guess* the drift value of the source by trying several keys (time values) within a specific window to attempt to decode the message. Once the key is intelligently *guessed*, the sink knows the drift value for the source node and can now compensate for that value before sending the complete report to an external network/application. Further, the one-time dynamically changing encryption operation prevents malicious nodes from injecting false timing packets into the network. With SOBAS, we are able to achieve our main goal of synchronizing events loosely at the sink and at the data delivery path as quick, as accurate, and as surreptitious as possible. SOBAS is perfectly suitable for WSN applications that do not need perfect synchronization and it is able to provide 7.24μs clock precision on the data delivery path given today's sensor technology. Simulations show that SOBAS is an energy efficient scheme under normal operation and attack from malicious nodes.

1.2 Thesis Outline

The remainder of this thesis is organized as follows: Chapter 2 discusses general principles for researchers who seek to design secure WSN protocols. The VEBEK protocol is introduced in Chapter 3. While Chapter 4 presents the TICK scheme, the SOBAS protocol is introduced in Chapter 5. Finally, Chapter 6 summarizes the research results and suggests a number of problems for future investigation.

CHAPTER II

DESIGNING SECURE PROTOCOLS FOR WIRELESS SENSOR NETWORKS

The main goal of this chapter is to present general principles to aid in the design of secure WSN protocols. Building upon the established concepts and the experience garnered from the previous research efforts in the literature, all the security services from the WSNs' perspective (confidentiality, authentication, integrity, access control, availability, and nonrepudiation) are studied. Questions of which services would be necessary for resource-constrained WSNs and when it would be most reasonable to implement them for a WSN application are discussed. Suggestions for protocol designers to consider before attempting to build secure WSN protocols were first introduced in [6].

The remainder of this chapter is organized as follows. A motivation for the chapter is given in Section 2.1. Related work is presented in Section 2.2. Section 2.3 briely gives the communication and the threat models for WSNs. In the threat models section, we also introduce a new threat model, called Target-Based attacks as a complementary threat model to those in the current literature. Desired security services are explored in Section 2.4. Finally, Section 2.5 concludes the chapter by discussing which service should be provided for a particular scenario.

2.1 Motivation

Throughout the last decade, the introduction of WSNs to the networking field has gathered the attention of academia and industry. Today, WSNs are no longer a

nascent technology and future advances in technology will bring more sensor applications into our daily lives as well as into many diverse and challenging application scenarios. For example, WSNs would be very instrumental in applications from real-time target tracking, homeland security, battlefield surveillance, surveillance of territorial waters, to biological and chemical attack detection [9].

In this regard, designing secure protocols for wireless sensor networks is vital. However, designing secure protocols for WSNs first requires the detailed understanding of the WSN technology and its relevant security aspects. Compared to other wireless networking technologies, WSNs have unique characteristics that need to be taken into account when building protocols. Among many factors, the available resources (e.g., power, computational capacities, and memory) onboard the sensor nodes are severely limited. For instance, a typical sensor [16] operates at the frequency of 2.4 GHz, has a data rate of 250Kbps, 128KB of program flash memory, 512KB of memory for measurements, transmit powers between 100μW and 1mW, and 30m to 100m of communications range. Thus, the most important design parameter for WSN protocols is to be energy efficient. This fundamental fact heavily influences protocols that are designed for the WSN.

Although, over the years, a myriad of protocols have been proposed for WSNs and fundamental notions have been established well, trying to be energy efficient and optimize the limited resources available in WSN protocols have further brought new notions and directions in the WSN research. Some of these notions are directly in contrast to what have been considered and studied as reasonable for other types of wireless networks. For instance, today, it is believed that not all the communication layers from the protocol stack are needed to be implemented in the sensors [17]. This is reasonable as it both saves space from the implementation and reduces complexity. Thus, this chapter constitutes a bridge between salient features of the WSN protocols, applications and their security aspects by addressing the desired security services for

WSNs.

The main goal of this chapter is to provide a basin of concepts for protocol designers to consider before attempting to build secure WSN protocols. Specifically, building upon the established concepts and the experience garnered from the previous research efforts in the literature, we sift through all the security services (confidentiality, authentication, integrity, access control, availability, and nonrepudiation). First, what a particular security service means from the WSN's perspective is discussed. Second, how that service has been studied in the literature is briefly addressed. Finally, we present further suggestions by questioning the need of that service for WSNs. We believe further improvements can be accomplished by unbundling some of the unnecessary security services, which may be contrary to most of the established principles.

2.2 Related Work

There are many surveys on WSN security. Some provide classifications and address the relevant issues from a general perspective [18], [19], [20], [21] [22], [23]. On the other hand, some others focus only on a particular layer of protocols such as [19], [24] or approach the problems from a single security problem perspective [25], [26], [27]. Overall, these are all useful studies that would constitute a good starting point for security research in the WSN domain. However, some of them address security from a traditional perspective as applicable to other existing wireless networks. Moreover, there are very few dedicated studies such as [28], [29] which primarily focus on vulnerabilities of various WSN protocols.

We fundamentally differ from earlier efforts in two ways: (1) We question and identify the need of a particular security service for WSNs, and provide suggestions to protocol builders to consider before attempting to build secure WSN protocols. (2) Given the unique features of WSNs, we introduce a new threat model, called Target-Based attacks as a complementary threat model to the current literature.

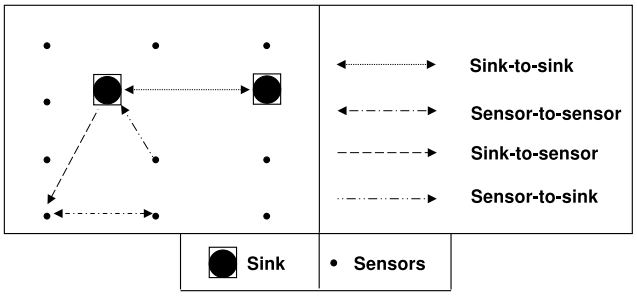

Figure 1: WSNs communication model

2.3 The WSN Communication & Threat Models

In this section, we articulate the communication and the threat models for the WSN, which is significant to capture the security aspect of the problem. In WSNs, only sensor-to-sensor, sink-to-sensor, and sensor-to-sink communications can occur. In some applications, where more than one sink is present, there may be a sink-to-sink communication as well. The possible communications are illustrated in Figure 1.

There are several threats to a WSN protocol. Conceptually, the threats could be listed from different perspectives. The previous research have listed threats according to how attacks are accomplished (e.g., Passive-Active Attacks)[30], on which layer of the communication stack they are realized (e.g., Layered Attacks) [22], and finally whether the malicious node becomes a member of the network during the attack or not (e.g., Member and Non-Member Attacks) [31]. Essentially, current literature for threat models resemble the ones done for wireless networks in general, which is a legitimate starting point, because many of the attacks could be borrowed from the literature for wireless networks. However, given the unique nature of a WSN, threats can be studied from another perspective. For instance, different functionalities could have been implemented at different parts of the network in order to efficiently utilize the resources of the WSN. Thus, an attacker first identifies where the critical functionalities are implemented in the network and then perpetrates its malicious intent on those identified targets. Thus, motivated to define another proper threat model

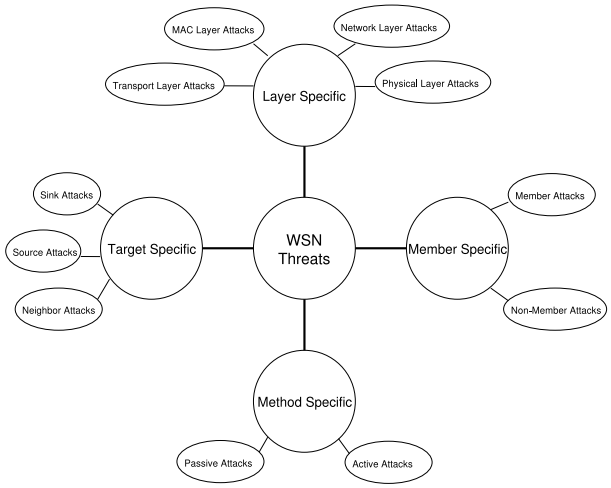

Figure 2: WSNs threat model including the new Target-Based Attacks

for WSNs, in this chapter, we also introduce a new threat model, *Target-Based Threat Model*, which is distinguished according to where and on which networking components the attacks are targeted (i.e., Sink, Neighbor, and Source Attacks). Target-based model complements the previous research on the issue. In reality, there is no hard line between these attacking types. The threat model for the WSN is given in Figure 2.

2.4 Desired Security Services from the WSNs Perspective

A structured definition of desired security services and mechanisms for the interconnection of open systems have been developed as an international standard by the International Telecommunication Union (ITU) inside Recommendation X.800 [32], which is referred to as the Security Architecture for OSI. This security architecture has been a valuable guideline for many researchers and practitioners who aim to develop secure systems. Thus, in this section we look at this reference security architecture from the perspective of WSNs.

Inside X.800, there are five major service categories: *Authentication, Access Control, Data Confidentiality, Data Integrity*, and *Nonrepudiation*. Although *Availability* has not originally been considered as one of the security services in X.800, it is also

included in our discussion below, as it pertains to desired security services for WSNs.

Similar to other WSNs protocols and applications, three performance metrics are pertinent when providing security services for WSNs. These performance metrics are independent of the chosen encyption mechanism. One is the *storage*, another is the *communication*, and the last is *computational* cost. For WSNs, the *communication cost* is the costliest among all the others and the chosen security mechanism implemented should try to use these scare resources efficiently.

These security services are studied below. Specifically, first, what the particular security service means in the WSN's domain is given; and second, how that service has been addressed in the literature is articulated briefly.

2.4.1 Confidentiality

2.4.1.1 WSN Perspective Definition:

Confidentiality refers to the protection of the exchanged content (e.g., gathered data, reports, commands) among the sink(s) and the sensors. An adversary which has the privilege to access the content, should not be able to decode the exchanged messages in the network.

2.4.1.2 Current Approaches:

Providing a confidential service to WSN applications requires the usage of cryptographic measures like encryption techniques. In general, two distinct forms of encryption approaches are in common use: *symmetric* and *asymmetric* key based schemes. Symmetric key based encryption uses the same key at both ends of the communication to encrypt and decrypt the information from ciphertext to plaintext and vice verse. On the other hand, with asymmetric key based encryption, a different key (one private and one public) are utilized to convert and recover the information.

The general important observation about encryption mechanisms is that one cannot claim that one encryption method is superior to another as it is essentially a

matter of the key size and the computational effort in breaking the encryption algorithm [30]. The second aspect to confidentiality research in WSNs entails designing efficient *key management* schemes because regardless of the encryption mechanism chosen for WSNs, the keys must be made available to the communicating nodes (e.g., sources, sink(s)) to maintain the privacy of the channels. The key management process involves two fundamental steps: generation (after an analysis) and distribution of keys; and it is triggered by keying events (e.g., due to node addition, stale keys) in the network. Nonetheless, it is not an easy task and even in some applications it may be daunting operation to visit a large number of sensors and update their keys (e.g., for underwater sensor applications). Thus, intelligent key management schemes are necessary for WSN.

There are two further observations for confidentiality research in WSNs. First, the research mainly focuses on different keying mechanisms rather than on building efficient symmetric or asymmetric encryption algorithms. This is reasonable because it is not easy to devise a new encryption technique due to the complex and rigorous mathematical processes involved. Second, as for the keying mechanisms, it is seen that current research mainly revolves around the key distribution step because for the resource limited WSN, it is not efficient to repeat the analysis and key generation with every occurrence of a keying event.

The following list gives an overview of the research for both the encryption and key management mechanisms for WSNs.

- *Encryption mechanisms*: In recent works, the feasibility of two encryption techniques have been well scrutinized and understood for the WSN domain. With the current technological advances in the field of micro-electro-mechanical systems, symmetric encryption techniques is more tailored to WSNs. There are several reasons for this. First of all, using the same key at both ends saves the storage space. For instance in a simple worst case scenario assume that

there are N number of nodes in the network. While for symmetric encryption, a given node must posses N-1 number of keys in order to communicate to the other N-1 nodes, for asymmetric encryption, the same node must have N keys, N-1 for others' public keys, one for its own private key. Considering the fact that the key sizes for symmetric algorithms (e.g., 128 bits for AES) are generally smaller than those of asymmetric ones (e.g., recommended 1024 bits for RSA and 160 bits for Elliptic Curve Cryptography (ECC) Based Public Key Scheme), one can conclude that depending on the specified key size of the particular algotrithm chosen, the symmetric encryption algorithms may help reduce the per-node storage space. Secondly, the symmetric encryption algorithms have been known to utilize the resources more efficiently than their asymmetric counterparts as their cryptographic operations take less time and require much less energy consumption than that of asymmetric cryptographic ones [10]. This is primarily due to the fact that the symmetric encryption algorithms are faster in computation as they employ more primitive operations in their algorithms, like substitution and permutation of symbols that are implemented at the hardware level via shifts and XORs, rather than operations applying mathematical functions like modular arithmetic and exponentiation, which are the basis of public key encryption mechanisms. Lastly, the exchange of smaller size keys, when needed in a WSN application, consumes less communication resources, which favors symmetric schemes. A detailed discussion of key mechanisms are given below.

- *Key management mechanisms*: As mentioned above, there are two fundamental steps in the key management process: *generation* and *distribution* of keys. The *key generation* step deals with generation of the keys. Depending on the key type that is going to be deployed in the WSN, the keys can be generated once or multiple times during the lifetime of the WSN. The practical approach

adopted so far in this avenue of research has been to generate one time different keys such as session, network-wise, master, and group-wise keys depending on the topology and on the application requirements of WSNs. While this helps decrease the computation cost for WSNs, it may increase the storage cost on nodes depending on the key distribution scheme. The second step is the *distribution* of keys. The keys should be made available to the nodes without allowing others to see the keys. Traditionally, the keys have been exchanged between the end-points of the communication directly, or indirectly through trusted intermediaries (e.g., Key Distribution Center). The keys could be distributed to the sensors before the network is deployed or they could be re-distributed to nodes on demand as triggered by keying events. In the jargon of security research for WSNs, the former is phrased as *Static Key* management whereas the latter is as *Dynamic Key* management. For WSNs, the communication cost dominates other critical cost parameters, e.g., storage and computation [33]. Thus, the research for key distribution has focused more on static key management schemes. *Static key* management schemes perform key management functions statically prior to or shortly after network deployment. One famous pioneering work in this avenue is by Eschenauer and Gligor [10, 33], where each sensor in the WSN is pre-configured with a random subset of keys from a large key pool. To agree on a key for communication, two sensor nodes find one common key within their subsets and use this key as their shared secret key. On the other hand, *dynamic key* management schemes perform the key management steps either periodically or on demand due to keying events in the network. The leading approach in dynamic keying schemes involves exclusion-based systems [11], the basic notion of which requires each node to have k keys out of $k + m$ keys. m keys are disguised from the attackers and are used only when new keys need to be created once keying events are triggered in the network.

2.4.2 Authentication

2.4.2.1 WSN Perspective Definition:

Authentication service involves genuineness of the communication. An authentication mechanism verifies if the exchanged information is emanating from the legitimate participant of the WSN. This is needed because a malicious entity (e.g., a compromised node) may be able to inject counterfeit content or resend the same content into the network. Moreover, the X.800 specification recommends two sub-cases for authentication. The first involves the authentication of a peer entity and the second deals with the authentication of the origin of the data. For WSNs, the former means authentication of all the nodes that participate in the communication. Authentication can be done between two nodes communicating or one node (e.g., cluster head) and several other nodes around that node (i.e, broadcast authentication). The latter can be implemented at the sink or at an intermediary sensor node where data aggregation takes place.

2.4.2.2 Current Approaches:

There are several traditional methods of authentication in the literature [30]. One is password and depends on the premise of showing that one knows a secret. The node sends a password with its login information. The receiver verifies that the node is legitimate by checking that the password is associated with the sender node.

The other method is cryptographic-based, which is also called challenge-response. A classic technique to provide authentication would be to utilize Message Authentication Codes (MACs). The authenticated sensor node is required to provide the MAC code to be authenticated by the the authenticator sensor node. For MACs, hashes, symmetric key-based encryption, asymmetric key-based encryption methods may all be utilized. Thus, there are several practical ways of creating MACs, but simply creating a MAC involves possesing the same secret at both ends and either

encrypting the hash of the content with that key or hashing both the key and the content together. However, as discussed in the confidentiality subsection above, the encryption mechanisms have their associated costs, thus they should be employed with caution.

The last authentication method is address-based or identy-based. For this, the authenticator sensor node can check the identity or the location of the sender node. The passwords are not sent across the network with these schemes. In comparison to the previous two mechanism, this method would be very practical for WSNs but would not provide a strong authentication mechanism because it is trivial to spoof a sensor ID.

Two of the former leading works include SPINS [34] and TinySec [35]. They both employ symmetric encryption algorithms and work at the link layer.

2.4.3 Integrity

2.4.3.1 WSN Perspective Definition:

The recipients in the WSN should be able to detect if the exchanged content between the communicating participants of the WSN have been altered. Furthermore, for the WSN, the integrity service should also ensure that the exchanged content is not deleted, replication of old data, counterfeit, or stale.

2.4.3.2 Current Approaches:

Integrity of the exchanged content is usually provided with the digest of the content appended to the content itself. When the recipient sensor node receives the message it checks to see if the digest of the content that it computes and the digest received equals each other. If they are equal, then it accepts it as a legitimate message.

Content digests in integrity are created with the usage of hashing algorithms. There are many hashing algorithms in use today. Usually, hashing algorithms do not require the presence of keys unless they are specifically designed to work with keys

like keyed-hashing (e.g., HMAC, CMAC). Thus, their impact on a sensor node is only confined by their computational efficiencies. However, as for the keyed-hashing algorithms, previously discussed issues emanating from key generation, key storage, and key exchange are also pertinent here, hence the keyed-hashing techniques must utilize the resources (computation, communication, and storage) efficiently.

Staleness of the data is of utmost significance in the integrity checking because decision processes of some applications may especially depend on if the data is recent or not. For example, in one very specific WSN application, a certain territory (e.g., territorial waters) could be protected with mines that are detonated by sinks. The freshness and the correct timing of the messages from the sensor nodes in this type of application is very important. A simple solution for these types of applications would be to use counters for the exchanged content. Lastly, another desired aspect of the integrity service may involve providing a recovery mechanism from the altered content.

2.4.4 Access Control

2.4.4.1 WSN Perspective Definition:

With access control, unauthorized use of a resource is prevented in WSNs. It addresses which participant of the network reaches which content or service. For instance, sensor nodes should not be allowed to have the privileges of sinks such as changing network-wide parameters of the WSN protocols. Thus, limiting services or functionalities depending on the participant would be appropriate.

2.4.4.2 Current Approaches:

One of the most challenging security services for WSNs is access control; perhaps, this is why access control for WSNs is one of the security services that have not been studied well in the literature [36]. We believe that part of this is because it is hard to formulate an access control scenario for WSNs. In practical implementations,

normally there is one terminating point (i.e., sink) in the network where all the data collected from the network is collected. Thus, other sensors are not expected to access any resource that may be hosted by other nodes. This is a reasonable expectation for WSN applications where sensors send their readings based on an event. However, there may be sensor applications where source sensor nodes are queried by other sensor nodes as well. For these circumstances, an access control policy can be used. An access control policy should prevent unauthorized nodes from accessing important information.

Setting access policies may also be practical and instrumental for cluster-based or hierarchical sensor node implementations.

2.4.5 Nonrepudiation

2.4.5.1 WSN Perspective Definition:

Nonrepudiation ensures that a sensor can not refute the reception of a message from the other involved party or the sending of a message to the other involved party in the communication. According to the X.800 recommendation, the former is destination nonrepudiation and the latter is called origin nonrepudiation.

2.4.5.2 Current Approaches:

Similar to access control, nonrepudiation has not been formulated well for the WSN's domain. This could be attributed to the lack of need of such a service for WSNs. Or, it could have been thought to exist inside integrity or authentication services implicitly.

Although the need for nonrepudiation service may not seem to be obvious, we think that it is an achievable important service to contemplate and that there are some practical advantageous in providing this service. A digital signature scheme (DSS) [30], which is based on utilizing encryption methods would also address nonrepudiation. Symmetric and asymmetric encryptions can be utilized for DSS. However,

their viabilities should be explored in more detail for WSNs. For instance, on the one hand, using the same key for both signature and verification may be vulnerable to another sensor's impersonation of the original sensor's signature. On the other hand, however, employing asymmetric encryption based algorithms may be costly. Naturally, providing nonrepudiation service may facilitate the endorsement or proof by another entity for a sent or received message in the WSN. Thus, alternatively, some other trusted node, either the sink or an aggregator node, in the network could provide this service.

2.4.6 Availability

2.4.6.1 WSN Perspective Definition:

Due to threats to the WSN, some portion of the network or some of the functionalities or services provided by the network could be damaged and unavailable to the participants of the network. For instance, some sensors could die earlier than their expected lifetimes. Thus, availability service ensures that the necessary functionalities or the services provided by the WSN are always carried out, even in the case of attacks or premature deaths.

2.4.6.2 Current Approaches:

Availability is a security service that has not been originally considered as one of the security services inside the X.800 recommendation. It may be claimed that it is independent of the security services. The outcome of the secure services provided by the network should guarantee the operations and functionalities aimed by the WSN application. Availability service for WSNs have been mostly studied from the perspective of Denial-of-Service type attacks [21] in the literature. One other pertinent study regarding availability has focused on the connectivity properties of WSNs [23].

2.5 When to Employ Specific Security Services

Sensor nodes are severely limited in their capabilities. There are three important design parameters for WSNs: communication, computation, and storage cost. The cost of communication dominates over those of the computation and storage. So, any security service designed for WSNs should always try to minimize the cost of these parameters. Thus, providing a security service comes with its associated costs naturally as it is an additional service on top of whatever is provided by the network.

When we look at the security services in general, we see that they are often provided as bundled services. Another observation from the literature is that in comparison to other security services, confidentiality has been explored more because it is fundamental to all of the other security services, except for availability. We believe that for resource constrained devices like sensor nodes in WSNs, there can be further minimization of the associated cost by just unbundling the unnecessary services. This would require the understanding of the needs of the network. Therefore, security services should be tailored to the applications, as it would be a waste of important resources in the network if all the security services are unnecessarily implemented. Looking at the security services and the improvements in the field, below is a discussion of how the security services should be analyzed for WSNs.

- Confidentiality of data should always be questioned as the confidentiality will always be the most costly security service among all the security services. Unless it is utmost necessary for the WSN, it may not have to be employed. An integrity check on the data may suffice to determine the activity of a malicious entity in the WSN. Thus, confidentiality can be unbundled from the rest of the services and provided as an additional security service for the WSN and be addressed separately from the other services.

- The authentication service can be considered as a prevention mechanism for

WSNs applications. This is reasonable because when authenticating an untrusted sensor node, if that node is a malicious one, it may have or not have perpetrated its malicious intent yet. With authentication, the malicious node may be blocked from its intended activity. Thus, authentication may be used as a prevention mechanism. Furthermore, authentication may be necessary for aggregator sensor nodes that collect the sensors' readings, where the agregator sensor nodes asks the source sensor nodes for their sensor readings. The source nodes may need to authenticate the aggregator node.

- Providing integrity definitely determines if a malicious activity exists in the network. It can be considered as a detection mechanism rather than a prevention mechanism like authentication. Specifically, integrity check for WSNs can be done either at every sensor node or at data-aggregating nodes or sink(s). Checking at every node increases the computation cost, but eliminates the fake data immediately and prevents that data from propagating further. On the other hand, checking the integrity at aggregator nodes or sinks save from the computation, but not from the communication cost. This is an application specific parameter that should be considered when providing integrity for WSNs, which is a topic for further investigation.

- Intelligent bundling of the services is possible. For instance, the integrity can be embedded inside an authentication service. The nice thing about asymmetric systems is that they can be used for both authentication and integrity purposes. It is even possible to use an asymmetric encryption algorithm to provide authentication, integrity, and nonrepudiation. Although asymmetric encryption mechanisms are costlier than symmetric encryption mechanisms, further security services can be addressed in an all-in-one fashion. However, their applicability for WSNs needs further investigation.

- Access control comes naturally after authentication; thus, it may be beneficial to bundle these two. However, confidentiality and access control are separate issues that can be de-coupled and addressed separately.

- It is always cost effective for WSNs to employ security algorithms with smaller key sizes. Smaller key sizes will help save the network storage, and further, if the keys are exchanged in the network, it will save from the communication as well because communication of smaller keys consumes less communication overhead. Moreover, when smaller keys and asymmetric encryption is necessary, ECC based algorithms should be favored over the others as ECC based ones, have much better efficient utilization of the resources in place of others (e.g., RSA)

- Usage of different keys such as session, network-wise, master, and group-wise keys should be considered to isolate and to futher help counter malicious activities. Furthemore, albeit costlier than the static key management schemes, dynamic key management schemes are more tailored to WSN applications. There may be ways to generate keys dynamically without too much overhead. For instance, depending on something unique that a sensor posses, keys can be generated instead of being exchanged. For instance, the residual battery life or energy on a node [4, 13] or identity of the node could be utilized for this. However, depending on the application type and the needs, if the lifetime of the network is more important than security, then static key management schemes may be preferred in place of dynamic.

- Due to the resource constrainted nature of WSNs, there have been new ideas that are shaping the future of WSNs. Some of the promising ones include collaboration of sensors for the distributed networking functionalities, and delayering of the TCP/IP stack. There would be further savings of scarce resources

of WSNs if these are considered when building secure WSN protocols. For instance, collaborative security, application-oriented security, and non-layered security approaches may be promising, but they need further investigation.

- Availability should not be considered outside of security services, the network should have worst case secure data delivery scenarios in case of any security breach or malicious attack. However, this can be thought of in a layered fashion. Unless there is a security problem in the network, the alternative availability mechanism may not be considered. However, this is again an application oriented issue for WSNs. For some applications, where the timely collection of data is utmost important, the availability should be considered at the same as security services.

- For application where different types of sensor nodes co-exist or a composite of events [37] occur in the same WSN application, it may be very important to provide an access control service. Similarly, having access policies may be instrumental for cluster-based or hierarchical sensor node implementations.

CHAPTER III

VEBEK: *V*IRTUAL *E*NERGY-*B*ASED *E*NCRYPTION AND *K*EYING FOR WIRELESS SENSOR NETWORKS

In this chapter, ***V***irtual ***E***nergy-***B***ased ***E***ncryption and ***K***eying protocol for Wireless Sensor Networks is introduced. VEBEK is a secure communication framework where sensed data is encoded using a scheme based on a permutation code generated via the RC4 encryption mechanism. The key to RC4 dynamically changes as a function of the residual virtual energy of the sensor. Thus, a one-time dynamic key is employed for one packet only and different keys are used for the successive packets of the stream without specific rekeying messages. The nodes forwarding the data along the path to the sink are able to verify the authenticity and integrity of the data and to provide non-repudiation. Moreover, the protocol is able to continue its operations under dire communication cases as it may be operating in an high-error-prone deployment area like under water. The initial concept of dynamic energy-based encoding and filtering was originally introduced by the DEEF [13] framework, then it was revised and significantly enhanced in the VEBEK protocol [4].

This chapter proceeds as follows. A motivation for the VEBEK scheme is given in Section 3.1. Related work is presented in Section 3.2. To further motivate the work, an analysis of the rekeying cost with and without explicit control messages is given in Section 3.3. Section 3.4 discusses the semantics of VEBEK. VEBEK's different operational modes are discussed in Section 3.5. An analytical framework and performance evaluation results including a comparison with other relevant works are given in Section 3.6. Finally, Section 3.7 concludes the chapter by summarizing the design rationale and benefits of the VEBEK framework.

3.1 Motivation

Rapidly developed WSN technology is no longer nascent and will be used in a variety of application scenarios. Typical application areas include environmental, military, and commercial enterprises [9]. For example, in a battlefield scenario, sensors may be used to detect the location of enemy sniper fire or to detect harmful chemical agents before they reach troops. In another potential scenario, sensor nodes forming a network under water could be used for oceanographic data collection, pollution monitoring, assisted navigation, military surveillance, and mine reconnaissance operations. Future improvements in technology will bring more sensor applications into our daily lives and the use of sensors will also evolve from merely capturing data to a system that can be used for real-time compound event alerting [37].

From a security standpoint, it is very important to provide authentic and accurate data to surrounding sensor nodes and to the sink to trigger time-critical responses (e.g., troop movement, evacuation, first response deployment) [6]. Protocols should be resilient against false data injected into the network by malicious nodes. Otherwise, consequences for propagating false data or redundant data are costly, depleting limited network resources and wasting response efforts.

However, securing sensor networks poses unique challenges to protocol builders because these tiny wireless devices are deployed in large numbers, usually in unattended environments, and are severely limited in their capabilities and resources (e.g., power, computational capacity, and memory). For instance, a typical sensor [16] operates at the frequency of 2.4 GHz, has a data rate of 250Kbps, 128KB of program flash memory, 512KB of memory for measurements, transmit power between 100μW and 1mW, and a communications range of 30m to 100m. Therefore, protocol builders must be cautious about utilizing the limited resources onboard the sensors efficiently.

In this chapter, we focus on keying mechanisms for WSNs. There are two fundamental key management schemes for WSNs: *static* and *dynamic*. In static key

management schemes, key management functions (i.e., key generation and distribution) are handled statically. That is, the sensors have a fixed number of keys loaded either prior to or shortly after network deployment. On the other hand, dynamic key management schemes perform keying functions (rekeying) either periodically or on demand as needed by the network. The sensors dynamically exchange keys to communicate. Although dynamic schemes are more attack-resilient than static ones, one significant disadvantage is that they increase the communication overhead due to keys being refreshed or redistributed from time to time in the network. There are many reasons for key refreshment, including: updating keys after a key revocation has occurred, refreshing the key such that it does not become stale, or changing keys due to dynamic changes in the topology. In this chapter, we seek to minimize the overhead associated with refreshing keys to avoid them becoming stale. Because the communication cost is the most dominant factor in a sensor's energy consumption [14] [15], the message transmission cost for rekeying is an important issue in a WSN deployment (as analyzed in the next section). Furthermore, for certain WSN applications (e.g., military applications), it may be very important to minimize the number of messages to decrease the probability of detection if deployed in an enemy territory. That is, being less "chatty" intuitively decreases the number of opportunities for malicious entities to eavesdrop or intercept packets.

The purpose of this chapter is to develop an efficient and secure communication framework for WSN applications. Specifically, in this chapter we introduce Virtual Energy-Based Encryption and Keying (VEBEK) for WSNs, which is primarily inspired by our previous work [13]. VEBEK's secure communication framework provides a technique to verify data in line and drop false packets from malicious nodes, thus maintaining the health of the sensor network. VEBEK dynamically updates keys without exchanging messages for key renewals and embeds integrity into packets as opposed to enlarging the packet by appending message authentication codes (MACs).

Specifically, each sensed data is protected using a simple encoding scheme based on a permutation code generated with the RC4 encryption scheme and sent toward the sink. The key to the encryption scheme dynamically changes as a function of the *residual virtual energy of the sensor*, thus requiring no need for rekeying. Therefore, a one-time dynamic key is used for one message generated by the source sensor and different keys are used for the successive packets of the stream. The nodes forwarding the data along the path to the sink are able to verify the authenticity and integrity of the data and to provide non-repudiation. The protocol is able to continue its operations under dire communication cases as it may be operating in a high-error-prone deployment area like under water. VEBEK unbundles key generation from other security services, namely authentication, integrity, and non-repudiation; thus, its flexible modular architecture allows for adoption of other encryption mechanisms if desired. The contributions of this chapter are as follows: (1) a dynamic en-route filtering mechanism that does not exchange explicit control messages for rekeying; (2) provision of one-time keys for each packet transmitted to avoid stale keys; (3) a modular and flexible security architecture with a simple technique for ensuring authenticity, integrity and non-repudiation of data without enlarging packets with MACs; (4) a robust secure communication framework that is operational in dire communication situations and over unreliable MACs. Both analytical and simulation results verify the feasibility of VEBEK. We also illustrate that VEBEK is significantly more energy efficient than other comparable schemes in the literature with an overall $60\%-100\%$ improvement.

3.2 Related Work

En-route dynamic filtering of malicious packets has been the focus of several studies, including dynamic en-route filtering (DEF) by Yu and Guan [1], statistical en-route filtering (SEF) [2], and Secure Ticket-Based En-route Filtering (STEF) [3]. As the

details are given in the performance evaluation section (Section 3.6) where they were compared with the VEBEK framework, the reader is referred to that section for further details as not to replicate the same information here. Moreover, Ma's work [38] applies the same filtering concept at the sink and utilizes packets with multiple MACs appended. A work [39] proposed by Hyun and Kim uses relative location information to make the compromised data meaningless and to protect the data without cryptographic methods. In [40], using static pairwise keys and two MACs appended to the sensor reports, "an interleaved hop-by-hop authentication scheme for filtering of injected false data" was proposed by Zhu et al to address both the insider and outsider threats. However, the common downside of all these schemes is that they are complicated for resource-constrained sensors and they either utilize many keys or they transmit many messages in the network, which increases the energy consumption of WSNs. Also, these studies have not been designed to handle dire communication scenarios unlike VEBEK. Another significant observation with all of these works is that a realistic energy analysis of the protocols was not presented. Lastly, the concept of dynamic energy-based encoding and filtering was originally introduced by the DEEF [13] framework. Essentially, VEBEK has been largely inspired by DEEF. However, VEBEK improves DEEF in several ways. First, VEBEK utilizes virtual energy in place of actual battery levels to create dynamic keys. VEBEK's approach is more reasonable because in real life, battery levels may fluctuate and the differences in battery levels across nodes may spur synchronization problems, which can cause packet drops. Second, VEBEK integrates handling of communication errors into its logic, which is missing in DEEF. Lastly, VEBEK is implemented based on a realistic WSN routing protocol, i.e., Directed Diffusion [41], while DEEF articulates the topic only theoretically.

Another crucial idea of this chapter is the notion of sharing a dynamic cryptic credential (i.e., virtual energy) among the sensors. A similar approach was suggested

inside the SPINS study [42] via the SNEP protocol. In particular, nodes share a secret counter when generating keys and it is updated for every new key. However, the SNEP protocol does not consider dropped packets in the network due to communication errors. Although another study, Minisec [43], recognizes this issue, the solution suggested by the study still increases the packet size by including some parts of a counter value into the packet structure. Finally, one useful pertinent work [15] surveys cryptographic primitives and implementations for sensor nodes.

3.3 An Analysis of the Rekeying Cost for WSNs

One significant aspect of confidentiality research in WSNs entails designing efficient *key management* schemes. This is because regardless of the encryption mechanism chosen for WSNs, the keys must be made available to the communicating nodes (e.g., sources, sink(s)). The keys could be distributed to the sensors before the network deployment or they could be re-distributed (rekeying) to nodes on demand as triggered by keying events. The former is *static key* [10] management and the latter is *dynamic key* [11] management. There are myriads of variations of these basic schemes in the literature. In this chapter, we only consider dynamic keying mechanisms in our analysis since VEBEK uses the dynamic keying paradigm. The main motivation behind VEBEK is that the communication cost is the most dominant factor in a sensor's energy consumption [14] [15]. Thus, in this section, we present a simple analysis for the rekeying cost with and without the transmission of explicit control messages. Rekeying with control messages is the approach of existing dynamic keying schemes whereas rekeying without extra control messages is the primary feature of the VEBEK framework.

Dynamic keying schemes go through the phase of rekeying either periodically or on demand as needed by the network to refresh the security of the system. With rekeying, the sensors dynamically exchange keys that are used for securing the communication.

Hence, the energy cost function for the keying process from a source sensor to the sink while sending a message on a particular path with dynamic key-based schemes can be written as follows (assuming computation cost, E_{comp}, would approximately be fixed):

$$E_{Dyn} = (E_{K_{disc}} + E_{comp}) * E[\eta_h] * \frac{\chi}{\tau} \tag{1}$$

where χ is the number of packets in a message, τ is the key refresh rate in packets per key, $E_{K_{disc}}$ is the cost of shared-key discovery with the next hop sensor after initial deployment, and $E[\eta_h]$ is the expected number of hops. In dynamic key-based schemes, τ may change periodically, on-demand, or after a node-compromise. A good analytical lower bound for $E[\eta_h]$ is given in [44] as

$$E[\eta_h] = \frac{D - t_r}{E[d_h]} + 1 \tag{2}$$

where D is the end-to-end distance (m) between the sink and the source sensor node, t_r is the approximated transmission range (m), and $E[d_h]$ is the expected hop distance (m) [45]. An accurate estimation of $E[d_h]$ can be found in [45]. Finally, $E_{K_{disc}}$, can be written as follows:

$$E_{K_{disc}} = \{(E[N_e] + 1) * E_{node} * M - E[N_e] * (E_{tx} + E_{rx})\} \tag{3}$$

$$E_{node} = E_{tx} + E_{rx} + E_{comp} \tag{4}$$

where E_{node} is the approximate cost per node for key generation and transmission, $E[N_e]$ is the expected number of neighbors for a given sensor, M is the number of key establishment messages between two nodes, and E_{tx} and E_{rx} are the energy cost of transmission and reception, respectively. Given the transmission range of sensors (assuming bi-directional communication links for simplicity), t_r, total deployment area, A, total number of sensors deployed, N, $E[N_e]$ can be computed as

$$E[N_e] = \frac{N * \pi * t_r^2}{A} \tag{5}$$

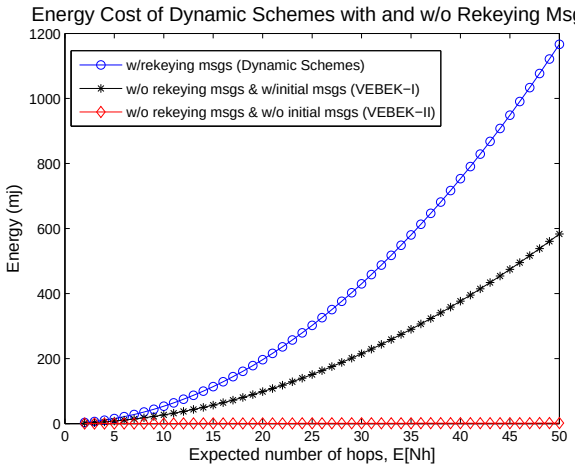

Figure 3: Keying cost of dynamic key-based schemes based on $E[nh]$ vs. VEBEK.

On the other hand, VEBEK does rekeying without messages. There are two operational modes of VEBEK (VEBEK-I and VEBEK-II). The details of these modes are given in Section IV. However, for now it suffices to know that VEBEK-I is representative of a dynamic system without rekeying messages, but with some initial neighborhood info exchange whereas VEBEK-II is a dynamic system without rekeying messages and without any initial neighborhood info exchange. Using the energy values given in [16], Figure 3 shows the analytical results for the above expressions. For both VEBEK modes, we assume there would be a fixed cost of E_{comp} [1] because VEBEK does not exchange messages to refresh keys, but for VEBEK-I, we also included the cost of $E_{K_{disc}}$.

With this initial analysis, we see that dynamic key-based schemes, in this scenario, spend a large amount of their energy transmitting rekeying messages. With this observation, the VEBEK framework is motivated to provide the same benefits of dynamic key-based schemes, but with low energy consumption. It does not exchange extra control messages for key renewal. Hence, energy is only consumed for generating the keys necessary for protecting the communication. The keys are dynamic; thus,

[1] A more rigorous analysis is presented in Section V.

one key per packet is employed. This makes VEBEK more resilient to certain attacks (e.g., replay attacks, brute-force attacks, masquerade attacks).

3.4 Semantics of VEBEK

The VEBEK framework is comprised of three modules: Virtual Energy-Based Keying, Crypto, and Forwarding.

The virtual energy-based keying process involves the creation of dynamic keys. Contrary to other dynamic keying schemes, it does not exchange extra messages to establish keys. A sensor node computes keys based on its residual *virtual energy* of the sensor. The key is then fed into the crypto module.

The crypto module in VEBEK employs a simple encoding process, which is essentially the process of permutation of the bits in the packet according to the dynamically created permutation code generated via RC4. The encoding is a simple encryption mechanism adopted for VEBEK. However, VEBEK's flexible architecture allows for adoption of stronger encryption mechanisms in lieu of encoding.

Lastly, the forwarding module handles the process of sending or receiving of encoded packets along the path to the sink.

A high level view of the VEBEK framework and its underlying modules are shown in Figure 4. These modules are explained in further detail below. Important notations used are given in Table 1.

Table 1: Notations Used in VEBEK

E_{tx}	Tx energy	E_{sens}	Sensing energy	E_{Fw}	Forwarding energy	P_{drop}	Drop probability
E_{rx}	Rx energy	E_{sa}	Staying alive energy	E_{Kdisc}	Key discovery energy	φ	Synch ratio
E_{comp}	Computation energy	E_{vc}	Virtual cost	E_{Dyn}	Dynamic keying cost	l	packet size
E_{enc}	Encoding energy	E_p	Perceived energy	E_{So}	Source node energy	N	# of nodes
E_{dec}	Decoding energy	E_b	Bridge energy	$E[\eta_h]$	Expected # of hops	r	# of watched nodes

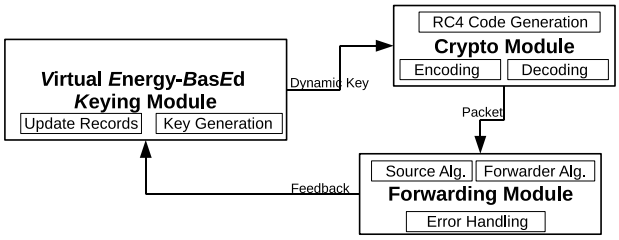

Figure 4: Modular structure of the VEBEK framework.

3.4.1 Virtual Energy-based Keying Module

The virtual energy-based keying module of the VEBEK framework is one of the primary contributions of this chapter. It is essentially the method used for handling the keying process. It produces a dynamic key that is then fed into the crypto module.

In VEBEK, each sensor node has a certain virtual energy value when it is first deployed in the network. The rationale for using virtual energy as opposed to real battery levels as used in DEEF [13], is that in reality battery levels may fluctuate and the differences in battery levels across nodes may spur synchronization problems, which can cause packet drops. These concerns have been addressed in VEBEK and are discussed in detail in the performance evaluation section (Section V).

After deployment, sensor nodes traverse several functional states. The states mainly include node-stay-alive, packet reception, transmission, encoding and decoding. As each of these actions occur, the virtual energy in a sensor node is depleted. The current value of the virtual energy, E_{vc}, in the node is used as the key to the key generation function, F. During the initial deployment, each sensor node will have the same energy level E_{ini}, therefore the initial key, K_1, is a function of the initial virtual energy value and an initialization vector (IV). The IVs are pre-distributed to the sensors. Subsequent keys, K_j, are a function of the current virtual energy, E_{vc}, and the previous key K_{j-1}. VEBEK's virtual energy-based keying module ensures that each detected packet[2] is associated with a new unique key generated based on the

[2]Indeed, the same key can be used for a certain number of transmissions, n, to further save

transient value of the virtual energy. After the dynamic key is generated, it is passed to the crypto module, where the desired security services are implemented. The process of key generation is initiated when data is sensed; thus, no explicit mechanism is needed to refresh or update keys. Moreover, the dynamic nature of the keys makes it difficult for attackers to intercept enough packets to break the encoding algorithm. The details are given in Algorithm 1. As mentioned above, each node computes and

Algorithm 1 Compute Dynamic Key

```
1: ComputeDynamicKey(E_vc, ID_clr)
2: begin
3: j ← tx_cnt^{ID_clr}
4: if j = 1 then
5:    K_j ← F(E_ini, IV)
6: else
7:    K_j ← F(K_(j−1), E_vc)
8: end if
9: return K_j
10: end
```

updates the transient value of its virtual energy after performing some actions. Each action (or state traversal) on a node is associated with a certain predetermined cost. Since a sensor node will be either forwarding some other sensor's data or injecting its own data into the network, the set of actions and their associated energies for VEBEK includes packet reception (E_{rx}), packet transmission (E_{tx}), packet encoding (E_{enc}), packet decoding (E_{dec}) energies, and the energy required to keep a node alive in the idle state (E_a).[3] Specifically, the transient value of the virtual energy, E_v, is computed by decrementing the total of these predefined associated costs, E_{vc}, from the previous virtual energy value.

The exact procedure to compute virtual cost, E_{vc}, slightly differs if a sensor node is the originator of the data or the forwarder (i.e., receiver of data from another sensor). In order to successfully decode and authenticate a packet, a receiving node must keep

energy.

[3]The set of actions can be extended to include other actions depending on the WSN application or functionality of the network.

track of the energy of the sending node to derive the key needed for decoding. In VEBEK, the operation of tracking the energy of the sending node at the receiver is called *watching* and the energy value that is associated with the watched sensor is called *Virtual Perceived Energy* (E_p) as in [13]. More formal definitions for watching are given as follows.

Definition 1 *Given a finite number of sensor nodes, N ($N = \{1, .., N\}$), deployed in a region, watching is defined as a node's responsibility for monitoring and filtering packets coming from a certain (configurable) number of sensor nodes, r, where $r <= N$. $\lessdot$ is used to denote the watching operation.*

Definition 2 *Given a sensor node i, the total number of watched nodes, r, which the node is configured to watch, constitutes a watching list, WL_i for node i and $WL_i = (1, 2, .., r)$. Node i watches node k if $ID_k \in WL_i$.*

Deciding which nodes to watch and how many depends on the preferred configuration of the VEBEK authentication algorithm, which we designate as the operational mode of the framework. Specifically, we propose two operational modes VEBEK-I and VEBEK-II and they are discussed in the next section.

When an event is detected by a source sensor, that node has remained alive for t units of time since the last event (or since the network deployment if this is the first event detected). After detection of the event, the node sends the l-bit length packet toward the sink. In this case, the following is the virtual cost associated with the source node:

$$E_{vc} = l * (e_{tx} + e_{enc}) + t * e_a + E_{synch} \tag{6}$$

In the case where a node receives data from another node, the virtual perceived energy value can be updated by decrementing the cost associated with the actions performed by the sending node using the following cost equation. Thus, assuming that the receiving node has the initial virtual energy value of the sending node and

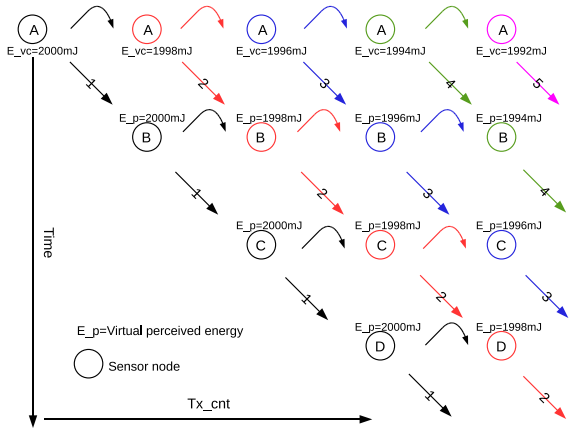

Figure 5: An illustration of the watching concept with forwarding.

that the packet is successfully received and decoded associated with a given source sensor, k, the virtual cost of the perceived energy is computed as follows:

$$E_p^k = l * (e_{rx} + e_{dec} + e_{tx} + e_{enc}) + t * 2 * e_a \tag{7}$$

where in both the equations, the small $e's$ refer to the one bit energy costs of the associated parameter. However, E_{synch} in (6) refers to a value to synchronize the source with the watcher-forwarders toward the sink as watcher-forwarder nodes spend more virtual energy due to packet reception and decoding operations, which are not present in source nodes. Hence, $E_{synch} = l * (e_{rx} + e_{dec}) + e_a * t$. The watching concept is illustrated with an example in Figure 5. In the figure, there is one source sensor node, A, and other nodes B, C, and D are located along the path to the sink. Every node watches its downstream node, i.e., B watches A ($B \lessdot A$); C watches B ($C \lessdot B$); D watches C ($D \lessdot C$). All the nodes have the initial virtual energy of 2000mJ and as packets are inserted into the network from the source node (A) over time, nodes decrement their virtual energy values. For instance, as shown in Figure 5, node A starts with the value of 2000mJ as the first key to encode the packet (key generation based on the virtual energies is explained in the crypto module). Node A sends the first packet and decrements its virtual energy to 1998mJ. In the figure, the diagonal arrows on the nodes represent the virtual energy transition from one

packet transmission to a subsequent one. After node B receives this first packet, it uses the virtual perceived energy value (E_p=2000mJ) as the key to decode the packet, and updates its E_p (1998mJ) after sending the packet. Note that nodes C and D do the similar operations as B on the packet. Finally, when the packet travels up to the sink, the virtual energy becomes a *shared dynamic cryptic credential* among the nodes along the path.

3.4.2 Crypto Module

Due to the resource constraints of WSNs, traditional digital signatures or encryption mechanisms requiring expensive cryptography is not viable. The scheme must be simple, yet effective. Thus, in this subsection we introduce a simple encoding operation similar to that used in [13]. The encoding operation is essentially the process of permutation of the bits in the packet according to the dynamically created permutation code via the RC4 encryption mechanism. The key to RC4 is created by the previous module (virtual energy-based keying module). The purpose of the crypto module is to provide simple confidentiality of the packet header and payload while ensuring the authenticity and integrity of sensed data without incurring transmission overhead of traditional schemes. However, since the key generation and handling process is done in another module, VEBEK's flexible architecture allows for adoption of stronger encryption mechanisms in lieu of encoding.

The packets in VEBEK consists of the ID (i-bits), type (t-bits) (assuming each node has a type identifier), and data (d-bits) fields. Each node sends these to its next hop. However, the sensors' ID, type, and the sensed data are transmitted in a pseudo random fashion according to the result of RC4. More specifically, the RC4 encryption algorithm takes the key and the packet fields (byte-by-byte) as inputs and produces the result as a permutation code as depicted in Figure 6. The concatenation of each 8-bit output becomes the resultant permutation code. As mentioned earlier, the key

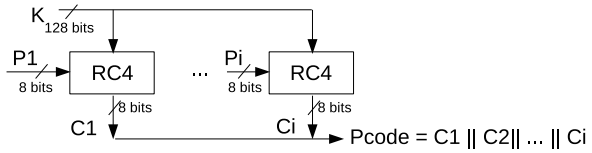

Figure 6: An illustration of the use of RC4 encryption mechanism in VEBEK.

to the RC4 mechanism is taken from the core virtual energy-based keying module, which is responsible for generating the dynamic key according to the residual virtual energy level. The resultant permutation code is used to encode the $\langle ID|type|data\rangle$ message. Then, an additional copy of the ID is also transmitted in the clear along with the encoded message. The format of the final packet to be transmitted becomes $Packet = [ID, \{ID, type, data\}_k]$ where $\{x\}_k$ constitutes encoding x with key k. Thus, instead of the traditional approach of sending the hash value (e.g., message digests, message authentication codes) along with the information to be sent, we use the result of the permutation code value locally. When the next node along the path to the sink receives the packet, it generates the local permutation code to decode the packet.

Another significant step in the crypto module involves how the permutation code dictates the details of the encoding and decoding operations over the fields of the packet when generated by a source sensor or received by a forwarder sensor.

Specifically, the permutation code P can be mapped to a set of actions to be taken on the data stream combination. As an example, the actions and their corresponding bit values can include simple operations such as shift, interleaving, taking the 1's complement, etc. Other example operations can be seen in Table 2.

For example, if a node computed the following permutation code $P = \{1100100101\}$, the string in Figure 7.a becomes the string in Figure 7.d before it is transmitted. The receiver will perform the same operations (since the inputs to RC4 are stored and updated on each sensor) to accurately decode the packet. To ensure correctness, the receiver compares the plaintext ID with the decoded ID. Moreover, although it is

Table 2: VEBEK Example Encoding Operations

Order of fields in pkt		**1's complement**	
ID, Type, Data	00	Yes	1
ID, Data, Type	01	No	0
Data, ID, Type	10	**Circular Shift**	
Data, Type, ID	11	Yes	1
Order of bits in field		No	0
Little Endian	0	**1-bit interleave**	
Big Endian	1	Yes	1
Shift Direction		No	0
Left	1	**Shift Amount**	
Right	0		

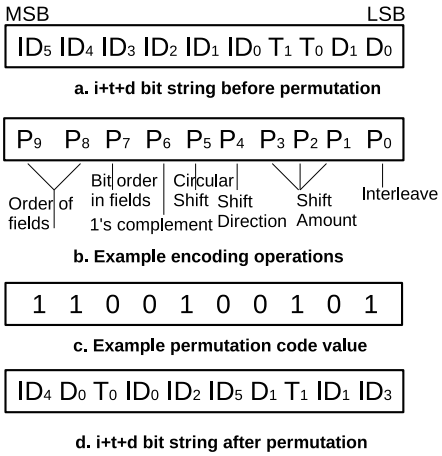

Figure 7: Illustration of a sample encoding operation.

theoretically possible (1 in 2^{i+t+d}) for a hacker to accurately inject data, it becomes increasingly unlikely as the packet grows.

The benefits of this simple encoding scheme are: 1) since there is no hash code or message digest to transmit, the packet size does not grow, avoiding bandwidth overhead on an already resource constrained network, thus increasing the network lifetime; 2) the technique is simple, thus ideal for devices with limited resources (e.g., PDAs); and 3) the input to the RC4 encryption mechanism, namely the key, changes dynamically without sending control messages to rekey.

3.4.3 Forwarding Module

The final module in the VEBEK communication architecture is the forwarding module. The forwarding module is responsible for the sending of packets (reports) initiated at the current node (source node) or received packets from other sensors (forwarding nodes) along the path to the sink. The reports traverse the network through forwarding nodes and finally reach the terminating node, the sink. The operations of the forwarding module are explained in this subsection.

3.4.3.1 Source Node Algorithm

When an event is detected by a source node the next step is for the report to be secured. The source node uses the local virtual energy value and an IV (or previous key value if not the first transmission) to construct the next key. As discussed earlier, this dynamic key generation process is primarily handled by the VEBEK module. The source sensor fetches the current value of the virtual energy from the VEBEK module. Then, the key is used as input into the RC4 algorithm inside the crypto module to create a permutation code for encoding the $\langle ID|type|data\rangle$ message. The encoded message and the cleartext ID of the originating node are transmitted to the next hop (forwarding node or sink) using the following format: $[ID, \{ID, type, data\}_{Pc}]$, where $\{x\}_{Pc}$ constitutes encoding x with permutation code Pc. The local virtual energy value is updated and stored for use with the transmission of the next report.

3.4.3.2 Forwarder Node Algorithm

Once the forwarding node receives the packet it will first check its watch-list to determine if the packet came from a node it is watching. If the node is not being watched by the current node, the packet is forwarded without modification or authentication. Although this node performed actions on the packet (received and forwarded the packet), its local virtual perceived energy value is not updated. This is done to maintain synchronization with nodes watching it further up the route. If the node is

being watched by the current node, the forwarding node checks the associated current virtual energy record (Algorithm 2) stored for the sending node and extracts the energy value to derive the key. It then authenticates the message by decoding the message and comparing the plaintext node ID with the encoded node ID. If the packet is authentic, an updated virtual energy value is stored in the record associated with the sending node. If the packet is not authentic it is discarded. Again, the virtual energy value associated with the current sending node is only updated if this node has performed encoding on the packet.

3.4.3.3 Addressing Communication Errors via Virtual Bridge Energy

In VEBEK, to authenticate a packet, a node must keep track of the virtual energy of the sending node to derive the key needed for decoding. Ideally, once the authenticating node has the initial virtual energy value of the sending node, the value can be updated by decrementing the cost associated with the actions performed by the sending node using the cost equations defined in the previous sub-sections on every successful packet reception. However, communication errors may cause some of the packets to be lost or dropped. Some errors may be due to the deployment region (e.g., underwater shadow zones) while operating on unreliable underlying protocols (e.g., MAC protocol). For instance, ACK or data packets can be lost and the sender may not be able to determine which one actually was lost. Moroever, malicious packets inserted by attackers who impersonate legitimate sensors will be dropped intentionally by other legitimate sensors to filter the bad data out of the network. In such communication errors or intentional packet drop cases, the virtual energy value used to encode the next data packet at the sending node may differ from the virtual energy value that is stored for the sending node at its corresponding watching node. Specifically, the node that should have received the dropped packet and the nodes above that node on the path to the sink lose synchronization with the nodes below (because

Algorithm 2 Forwarding Node Algorithm with Communication Error Handling

```
1: Forwarder(i ← currentNode, k ← WatchedNode, WL_i ← WatchList)
2: begin
3: enc ← 0; src ← 0; j ← 0
4: E_rx_i, ⟨ID_clr, {msg}_K⟩ ← ReceivePacket()
5: if ID_clr ∈ WL_i then
6:    while (keyFound = 0)and(j <= thresHold) do
7:       E^k_p_i ← FetchVirtualEnergy(i, ID_clr, enc, src)
8:       K ← ComputeDynamicKey(E^k_p_i, ID_clr)
9:       Pc ← RC4(K, ID_clr)
10:      E_dec_i, Msg_ID ← decode(Pc, {msg}_K)
11:      if ID_clr = Msg_ID then
12:         keyFound ← true
13:      else
14:         j + +
15:         E^k_p_i ← E^k_p_i − E_tx_i − E_enc_i − E_rx_i − E_dec_i − 2 ∗ E_a_i
16:      end if
17:   end while
18:   if keyFound = true then
19:      if j > 1 then
20:         reEncode ← true
21:      else
22:         if E_b_i > 0 then
23:            reEncode ← true
24:         else
25:            reEncode ← false
26:         end if
27:      end if
28:      if reEncode = true then
29:         enc ← 1
30:         E_b_i ← FetchVirtualEnergy(i, ID_clr, enc, src)
31:         K ← ComputeDynamicKey(E_b_i, ID_clr)
32:         Pc ← RC4(K, ID_clr)
33:         E_enc_i, {msg}_Pc ← encode(Pc, msg)
34:         packet ← ⟨ID_clr, {msg}_Pc⟩
35:         E_tx_i ← ForwardPacket()
36:         E_b_i ← E_b_i − E_tx_i − E_enc_i − E_rx_i − E_dec_i − 2 ∗ E_a_i
37:      else
38:         ForwardPacket() //Without any modification
39:      end if
40:   else
41:      DropPacket() //Packet not valid
42:   end if
43: else
44:   ForwardPacket() //Without any modification
45: end if
46: end
```

the upper portion never sees the lost packet and does not know to decrement the virtual energy associated with servicing the lost transmission). If another packet were to be forwarded by the current watching node using its current virtual energy, the upstream node(s) that watch this particular node would discard the packet. Thus, this situation needs to be resolved for proper functioning of the VEBEK framework. An illustration of this case is given in Figure 8.

To resolve potential loss of packets due to possible communication errors in the

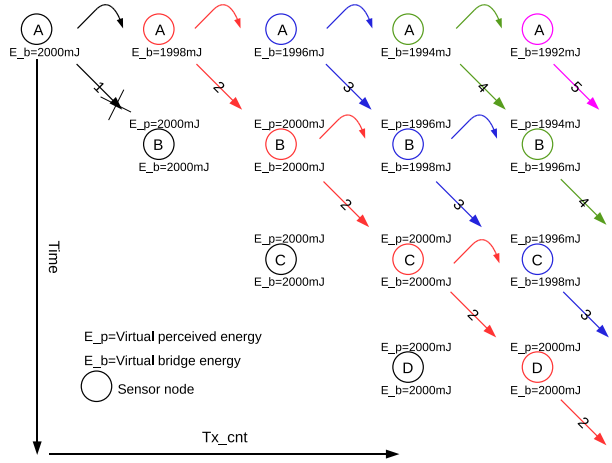

Figure 8: Illustration of the watching concept with forwarding when there is a packet loss.

network, all the nodes are configured to store an additional virtual energy value, which we refer to as the *Virtual Bridge Energy,* E_{b_i}, value to allow resynchronization (*bridging*) of the network at the next watching sensor node that determines that packets were lost.

Definition 3 *Given a node, i, bridging is defined as the process of encoding the incoming packet coming from any sensor node in* WL_i *for the upstream sensor node, j, with the key generated using the local copy of* E_{b_i}.

That is, as subsequent packets generated from the node of interest pass through the next watching node, the next watching node will decode the packet with the virtual perceived energy key of the originating node and re-encode the packet with the virtual bridge energy key, thus the network will be kept synchronized. It is important to note that once this value is activated for a watched node, it will be always used for packets coming from that node and used even if an error does not occur for the later transmissions of the same watched node. The watching node always updates and uses this parameter to keep the network bridged.

Another pertinent point is the determination of packet loss by the first upstream watching node who will bridge the network. The VEBEK framework is designed to

avoid extra messages and not increase the packet size to determine packet loss in the network. Thus, the next watching node tries to find the correct value of the virtual perceived energy for the key within a window of virtual energies. For this, a sensor is configured with a certain *VirtualKeySearchThreshold* value. That is, the watching node decrements the predefined virtual energy value from the current perceived energy at most virtualKeySearchThreshold times. When the node extracts the key successfully, it records the newest perceived energy value and associates it with the sender node (lines 7−18 in Algorithm 2). This approach may also be helpful in severe packet loss cases (i.e., bursty errors) by just properly configuring the virtualKeySearchThreshold value. However, if the watcher node exhausts all of the virtual energies within the threshold, it then classifies the packet as malicious.

The combined use of virtual perceived and bridge energies assure the continued synchronization of the network as whole. The forwarding node algorithm including the handling of communication errors is shown in Algorithm 2.

3.5 Operational Modes of VEBEK

The VEBEK protocol provides three security services: Authentication, integrity, and non-repudiation. The fundamental notion behind providing these services is the watching mechanism described before. The watching mechanism requires nodes to store one or more records (i.e., current virtual energy level, virtual bridge energy values, and Node-Id) to be able to compute the dynamic keys used by the source sensor nodes, to decode packets, and to catch erroneous packets either due to communication problems or potential attacks. However, there are costs (communication, computation, and storage) associated with providing these services. In reality, applications may have different security requirements. For instance, the security need of a military WSN application (e.g., surveiling a portion of a combat zone) may be higher than that of a civilian application (e.g., collecting temperature data from a national park).

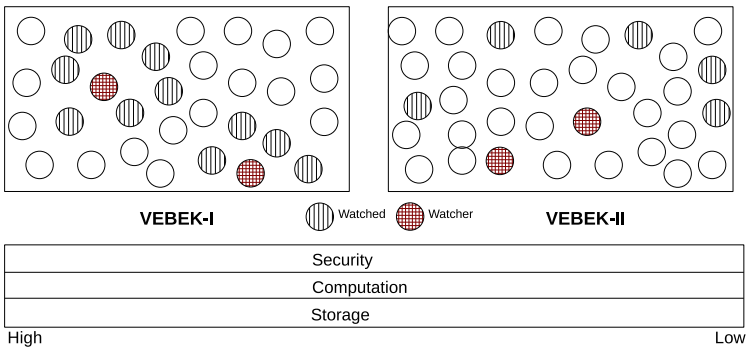

Figure 9: Operational Modes.

The VEBEK framework also considers this need for flexibility and thus, supports two operational modes: *VEBEK-I* and *VEBEK-II*. The operational mode of VEBEK determines the number of nodes a particular sensor node must watch. Depending on the vigilance required inside the network, either of the operational modes can be configured for WSN applications. Different modes and the range of associated costs of each mode are given in Figure 9. The details of both operational modes are given below. The performance evaluation of both modes is given in Section 3.6.

3.5.1 VEBEK-I

In the VEBEK-I operational mode, all nodes watch their neighbors; whenever a packet is received from a neighbor sensor node, it is decoded and its authenticity and integrity are verified. Only legitimate packets are forwarded toward the sink. In this mode, we assume there exists a short window of time at initial deployment that an adversary is not able to compromise the network, because it takes time for an attacker to capture a node or get keys. During this period, route initialization information may be used by each node to decide which node to watch and a record r is stored for each of its 1-hop neighbors in its watch-list. To obtain a neighbor's initial energy value, a network-wise master key can be used to transmit this value during this period similar to the shared-key discovery phase of other dynamic key management schemes. Alternatively, sensors can be pre-loaded with the initial energy value.

When an event occurs and a report is generated, it is encoded as a function of a dynamic key based on the virtual energy of the originating node, and transmitted. When the packet arrives at the next-hop node, the forwarding node extracts the key of the sending node (this could be the originating node or another forwarding node) from its record (the virtual perceived energy value associated with the sending node and decodes the packet). After the packet is decoded successfully, the plaintext ID is compared with the decoded ID. In this process, if the forwarding node is not able to extract the key successfully, it will decrement the predefined virtual energy value from the current perceived energy (line 16 in Algorithm 2) and tries another key before classifying the packet as malicious (because packet drops may have occurred due to communication errors). This process is repeated several times; however, the total number of trials that are needed to classify a packet as malicious is actually governed by the value of virtualKeySearchThreshold. If the packet is authentic, and this hop is not the final hop, the packet is re-encoded by the forwarding node with its own key derived from its current virtual bridge energy level. If the packet is illegitimate, the packet is discarded. This process continues until the packet reaches the sink. Accordingly, illegitimate traffic is filtered before it enters the network.

Re-encoding at every hop refreshes the strength of the encoding. Recall that the general packet structure is $[ID, \{ID, type, data\}_k]$. To accommodate this scheme, the ID will always be the ID of the current node and the key is derived from the current node's local virtual bridge energy value. If the location of the originating node that generated the report is desired, the packet structure can be modified to retain the ID of the originating node and the ID of the forwarding node.

VEBEK-I reduces the transmission overhead as it will be able to catch malicious packets in the next hop, but increases processing overhead because of the decode/encode that occurs at each hop.

3.5.2 VEBEK-II

In the VEBEK-II operational mode, nodes in the network are configured to only watch some of the nodes in the network. Each node randomly picks r nodes to monitor and stores the corresponding state before deployment. As a packet leaves the source node (originating node or forwarding node) it passes through node(s) that watch it probabilistically. Thus, VEBEK-II is a statistical filtering approach like SEF [2] and DEF [1]. If the current node is not watching the node that generated the packet, the packet is forwarded. If the node that generated the packet is being watched by the current node, the packet is decoded and the plaintext ID is compared with the decoded ID. Similar to VEBEK-I, if the watcher-forwarder node cannot find the key successfully, it will try as many keys as the value of virtualKeySearchThreshold before actually classifying the packet as malicious. If the packet is authentic, and this hop is not the final destination, the original packet is forwarded unless the node is currently bridging the network. In the bridging case, the original packet is re-encoded with the virtual bridge energy and forwarded. Since this node is bridging the network, both virtual and perceived energy values are decremented accordingly. If the packet is illegitimate, which is classified as such after exhausting all the virtual perceived energy values within the virtualKeySearchThreshold window, the packet is discarded. This process continues until the packet reaches the sink.

This operational mode has more transmission overhead because packets from a malicious node may or may not be caught by a watcher node and they may reach the sink (where it is detected). However, in contrast to the VEBEK-I mode, it reduces the processing overhead (because less re-encoding is performed and decoding is not performed at every hop). The trade-off is that an illegitimate packet may traverse several hops before being dropped. The effectiveness of this scheme depends primarily on the value r, the number of nodes that each node watches. Note that in this scheme, re-encoding is not done at forwarding nodes unless they are bridging the network.

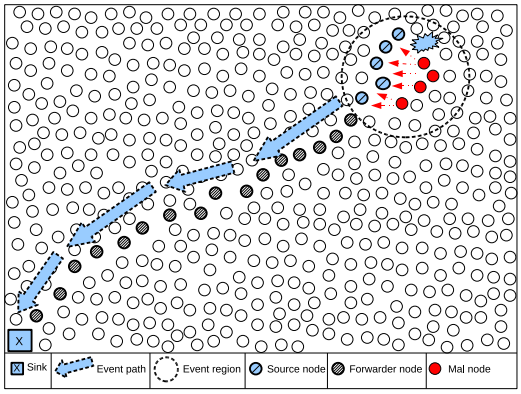

Figure 10: VEBEK simulation topology with GTSNetS.

3.6 Performance Analysis

In this section we evaluate the effectiveness of the VEBEK framework via both simulations and analysis.

3.6.1 Assumptions

Due to the broadcast nature of the wireless medium used in sensor networks, attackers may try to eavesdrop, intercept, or inject false messages. In this work, we mainly consider the false injection and eavesdropping of messages from an outside malicious node; hence, similar to [2], insider attacks are outside the scope of this work. This attacker is thought to have the correct frequency, protocol, and possibly a spoofed valid node ID. Throughout this chapter, the following assumptions are also made:

- Directed Diffusion [41] routing protocol is used, but others such as [46] can also be used. According to specifics of Directed Diffusion, after the sink asks for data via interest messages, a routing path is established from the sources in the event region to the sink. We assume that the path is fixed during the delivery of the data and the route setup is secure.

- The routing algorithm is deployed on an unreliable MAC. The network may experience ACK or data packet drops.

- The sensor network is densely populated such that multiple sensors observe and generate reports for the same event.

- Sensors are assumed to have the same communication ranges and may have different initial battery supplies.

3.6.2 Simulation Parameters

We use the Georgia Tech Sensor Network Simulator (GTSNetS) [47], which is an event-based object-oriented sensor network simulator with C++, as our simulation platform to perform the analysis of the VEBEK communication framework. The topology used for the simulation is shown in Figure 10, while the parameters used in the simulation are summarized in Tables 3 and 4. Nodes were distributed randomly in the deployment region and on average, the distance between the source nodes and the sink was around 25−35 hops. The virtualKeySearchThreshold value was 15 [7]. The energy costs for different operations in the table are computed based on the values given in [16]. However, the costs for encoding and decoding operations are computed based on the the reported values of the implementation of RC4 [48] on real sensor devices.

Table 3: VEBEK General Simulation Parameters

# of Nodes	500	SensSize	32 bytes
Area	1000x1000 m	RecvInterval	50s
# of Watched	(0..60)	SensRate	30s
Link Rate	250Kbps	SimTime	3000s
Range	75 m	#of Mal Node	(0..10)

Table 4: VEBEK Energy Related Parameters

E_{rx}	85.1μJ	E_{dec}	15.5μJ
E_{tx}	78μJ	E_{enc}	15.5μJ
E_{sens}	36μJ	Voltage	3V
E_{sa}	18.6μJ		

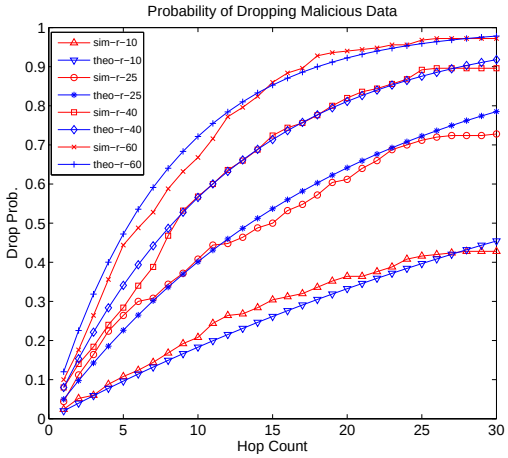

Figure 11: Theoretical and simulation results with varying number of watched nodes.

3.6.3 Attack Resilience

In this sub-section, the performance of VEBEK is analyzed when there are malicious source nodes in the data collection field who insert bad packets into the network. Specifically, the analytical basis of the VEBEK framework's resilience against malicious activities is formulated. Then, this theoretical basis is verified with the simulation results. We compare VEBEK-I and VEBEK-II considering the drop probability vs. number of hops. We also take a closer look at VEBEK-II and how it is affected by the parameter, r (the number of nodes to watch).

In VEBEK-I and VEBEK-II, in order for an attacker to be able to successfully inject a false packet, an attacker must forge the packet encoding (which is a result of dynamically created permutation code via RC4). Given that the complexity of the packet is 2^l where l is the sum of the ID, TYPE, and DATA fields in the packet, the probability of an attacker correctly forging the packet is:

$$P_{forge} = \frac{1}{2^{packetsize}} = \frac{1}{2^l} \tag{8}$$

Accordingly, the probability of the hacker incorrectly forging the packet and therefore

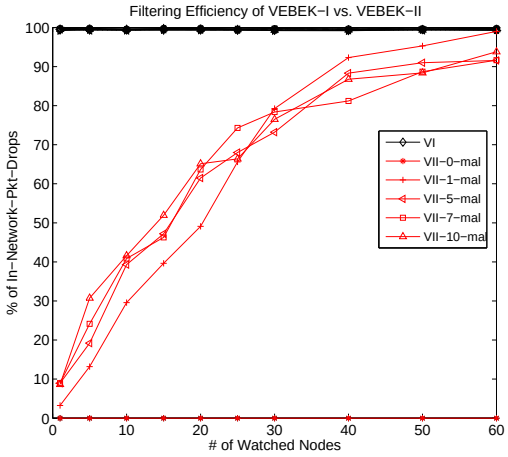

Figure 12: Comparison of filtering efficiency for VEBEK-I and VEBEK-II with varying number of malicious nodes.

the packet being dropped (p_{drop-I}) is:

$$P_{drop-I} = 1 - P_{forge} \tag{9}$$

Since VEBEK-I authenticates at every hop, forged packets will always be dropped at the first hop with a probability of P_{drop-I}.

On the other hand, VEBEK-II statistically drops packets along the route. Thus, the drop probability for VEBEK-II ($P_{drop-II}$) is a function of the effectiveness of the watching nodes as well as the ability for a hacker to correctly guess the encoded packet structure. Accordingly, the probability of detecting and dropping a false packet at one hop when randomly choosing r records (nodes to watch) is:

$$P_{drop-II} = \frac{r}{N} * (1 - P_{forge}) \tag{10}$$

Thus, the probability to detect and drop the packet when choosing r records after h hops is:

$$P^{r,h}_{drop-II} = 1 - (1 - p_{drop-II})^h \tag{11}$$

Moreover, even if one false packet successfully makes it to the sink, we assume that the sink has enough resources to determine which data to process and accept.

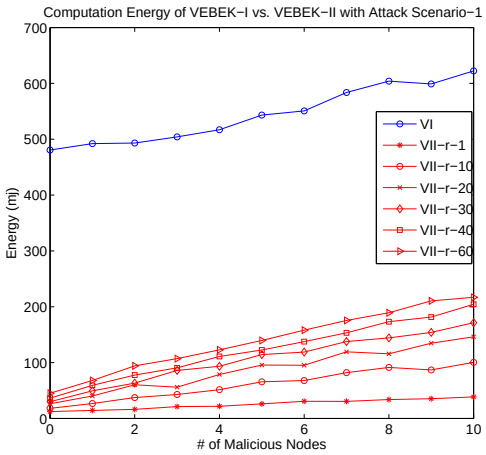

Figure 13: Computation costs under AS-1.

Figure 11 shows both the theoretical and simulation results for VEBEK-II based on the above equations for a varying number of watched nodes, r, in the WSN. Note that VEBEK-I is not shown in this figure because it eliminates malicious data immediately. The x-axis represents the number of hops a malicious packet travels before it has been detected and taken out of the network. As can be seen from the figure, VEBEK-II is able to eliminate malicious packets from the WSN within 15 hops with 0.5 probability when nodes watch 25 randomly chosen nodes (r value). However, if more storage is available on the sensors, then VEBEK-II can detect and remove malicious packets within 15 hops with 0.90 probability when r is 60. A similar trend is observed in the same figure with the simulation results.

On the other hand, Figure 12 presents the comparison of VEBEK-I (VI in the figure) and VEBEK-II (VII in the figure) via simulation in terms of their filtering efficiency. The x-axis represents the number of watched nodes (r) that each node is configured to watch in VEBEK-II and the y-axis shows the percent of in-network malicious packet dropped with varying number of malicious nodes in the simulation. As expected, we see that VEBEK-I is always able to filter malicious packets from the network with its 100% filtering efficiency. This is mainly due to the fact that malicious packets are immediately taken out from the network at the next hop. However, the

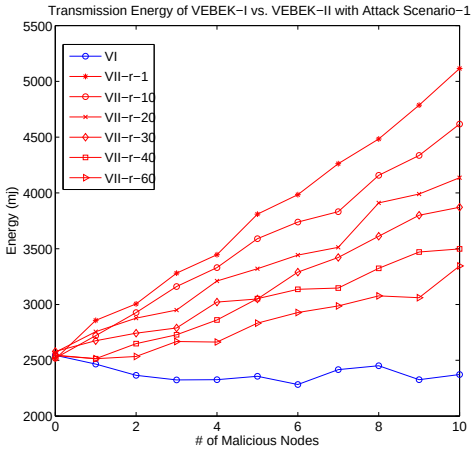

Figure 14: Transmissions costs under AS-1.

filtering efficiency of VEBEK-II is closely related to the number of nodes (r) that each node watches. The more nodes watched by other nodes, the more efficient VEBEK-II is with filtering malicious data. Additionally, as seen when r is equal to 40, it is possible to achieve almost 90% filtering efficiency. This particular observation with VEBEK-II is significant because for some WSN applications, energy can be saved by properly configuring the r parameter. Finally, with respect to Figure 12, we observe that the VEBEK framework is independent of the number of malicious nodes as the framework still filters the malicious data from the network successfully.

3.6.4 Energy Consumption of VEBEK-I and VEBEK-II

In this sub-section we look at the associated costs to transmit valid data in VEBEK-I and VEBEK-II. In both operational modes, there is a single cost (E_{So}) to stay-alive, sense the event, encode the packet, and transmit the packet ($E_{sa}, E_{sens}, E_{enc}, E_{tx}$) at the source sensor. Thus,

$$E_{So} = E_{sens} + E_{enc} + E_{tx} + E_{sa} \tag{12}$$

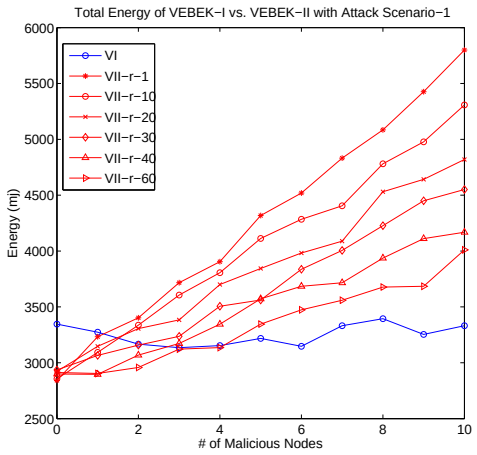

Figure 15: Total energy under AS-1.

Additionally, there is a recurring forwarding cost (E_{FW}) to marshal the packet through the network depending on the number of hops. In VEBEK-I, this cost is

$$E_{FW} = E_{rx} + E_{dec} + E_{enc} + E_{tx} + E_{sa} \tag{13}$$

for all of the intermediate nodes since all of the nodes perform the same operations. Hence, the average cost to transmit a packet in VEBEK-I using $E[\eta_h]$ from (2) is:

$$E_{FW_I} = E_{So} + (E[\eta_h] * E_{FW}) \tag{14}$$

On the other hand, in VEBEK-II the cost of $E_{FW_{II}}$ consists of E_{FW_w} and $E_{FW_{nw}}$ for variable fractions of the forwarding nodes depending on the number of nodes each node chose to watch, where $E_{FW_w} = E_{FW}$ and $E_{FW_{nw}} = E_{rx} + E_{tx} + E_{sa}$. Hence, the average cost to transmit a packet using VEBEK-II is:

$$E_{FW_{II}} = E_{So} + (E[\eta_{h_w}] * E_{FW_w}) + (E[\eta_{h_{nw}}] * E_{FW_{nw}}) \tag{15}$$

where $E[\eta_{h_w}]$ and $E[\eta_{h_{nw}}]$ represent the expected number of nodes along the path who are watchers and non-watcher nodes, respectively. The values for these expectations can be computed given the total expected number of hops with $E[\eta_h]$ from (2) where $E[\eta_h] = E[\eta_{h_w}] + E[\eta_{h_{nw}}]$ for $i = 1, 2, 3, .., \eta_h$.

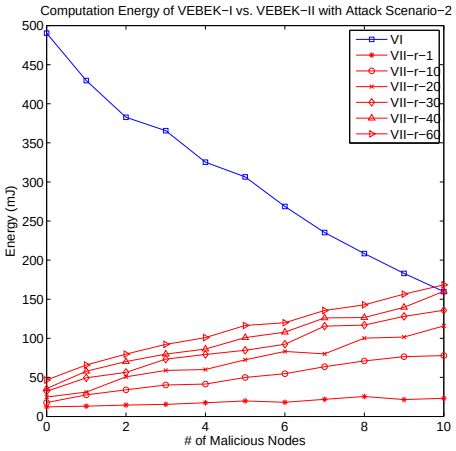

Figure 16: Computation costs under AS-2.

Let $X_i = 1$ if the i^{th} sensor is a watcher and let $X_i = 0$ otherwise for a given path to the sink with probabilities $P\{p = 1\} = \frac{r}{N}$, $P\{q = 0\} = \frac{N-r}{N}$, and N sensors. Then, $X_i \sim Bernoulli(p)$ i.i.d. random variables and $\eta_{h_w} = X_1 + ... + X_{\eta_h}$.

$$E[\eta_{h_w}] = E[\sum_{i=1}^{\eta_h} X_i] = E[E[\sum_{i=1}^{\eta_h} X_i | \eta_h]] \tag{16}$$

Hence, by the independence of X_i and η_h;

$$E[\eta_{h_w}] = E[\eta_h] * E[X_i] = \frac{r}{N} * E[\eta_h] \tag{17}$$

With a similar reasoning, an expression for the expected number of non-watchers, $E[\eta_{h_{nw}}]$, can be written as follows.

$$E[\eta_{h_{nw}}] = E[\eta_h] * E[X_i] = \frac{N-r}{N} * E[\eta_h] \tag{18}$$

Implementing these costs inside the GTSNetS simulator, we have evaluated the energy performance of the scheme both for VEBEK-I and VEBEK-II and plotted the results. In all the figures, the x-axis represents the number of malicious nodes while the y-axis is the energy consumption. Different values for the number of watched nodes (r) were analyzed for VEBEK-II. Furthermore, two attack scenarios were considered: Attack-Scenario-1 and Attack-Scenario-2. VEBEK-I and VEBEK-II are abbreviated as VI and VII in the figures.

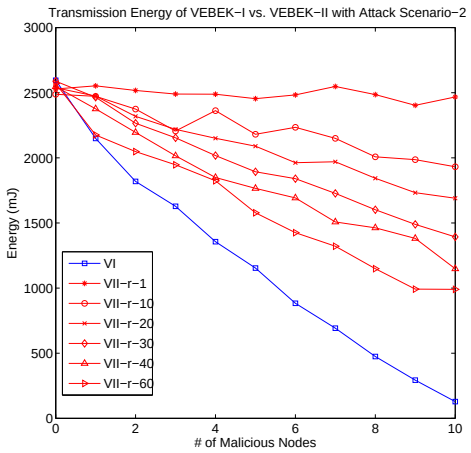

Figure 17: Transmissions costs under AS-2.

In Attack-Scenario-1 (AS-1), less powerful malicious nodes are assumed. The total number of healthy source nodes that collect the event information and send it toward the sink is assumed to be fixed, whereas the number of malicious nodes are increased over time. Letting i be the number of healthy source nodes and j be the number of malicious nodes, in Attack-Scenario-1, $j \leq i$, where $i = n$ and $n > 0$. Figures 13−15 show the results for Attack-Scenario-1 (AS-1). As seen from the computation costs (i.e., E_{enc}, E_{dec}) (Figure 13), VEBEK-II's consumption is less than that of VEBEK-I. The primary reason for this behavior stems from decoding and re-encoding of packets at every hop in the network for VEBEK-I. Also, as the number of watched nodes (r) increases, VEBEK-II's computation cost increases because more packets are processed for the filtering operation. On the other hand, the more malicious nodes in the system, the more resources are consumed to filter the increased number of malicious packets in the network. As for the transmission costs (i.e., E_{tx}, E_{rx}) in Figure 14, VEBEK-I is better as the nodes are able to catch and drop malicious packets and do not let malicious packets traverse the network. As r decreases, fewer nodes are watched by the sensors. Thus, the transmission cost increases in the network because more traffic traverses the network as a result of less filtering capability with smaller r values. Furthermore, as the number of malicious nodes increases in the network,

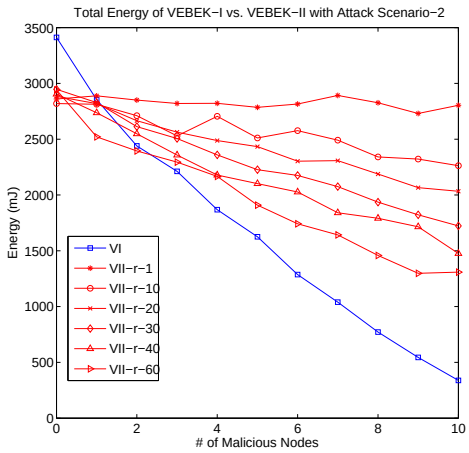

Figure 18: Total energy under AS-2.

the transmission cost increases due to more malicious traffic. Finally, analyzing the results for the total energy consumption, we see that the total energy consumption in the network exhibits a similar behavior as transmission costs because the overall energy consumption is greatly dominated by the transmission costs. Moreover, we observe that the total energy consumption for VEBEK-II is smaller than VEBEK-I up to a certain number of malicious nodes (1 and 2) for certain values of r (all watching values at 1 malicious node; and watching values of 30, 40, and 60 at 2 malicious nodes). The implication of this result is interesting. If the deployment region is a relatively safe environment (< 2 malicious nodes in our scenario), a similar filtering efficiency of VEBEK-I can be achieved using VEBEK-II (100% for VEBEK-I vs. 99% for VEBEK-II with $r = 60$) (Figure 12) if more storage is available on the nodes. This can be accomplished while consuming less energy than VEBEK-I (3400mJ for VEBEK-I vs. 2800 mJ for VEBEK-II). In Attack-Scenario-2 (AS-2), more powerful malicious nodes are assumed. For instance, they can jam the signal and not allow healthy nodes to transmit. Over time, more powerful nodes are assumed to replace the number of healthy source nodes. Hence, $j = 0, 1, 2, .., n$ and $i = n, n-1, n-2, .., 0$ where again $n > 0$. Figures 16−18 present the results for Attack-Scenario-2. In all the figures, it is possible to observe the same patterns as Attack-Scenario-1. The only

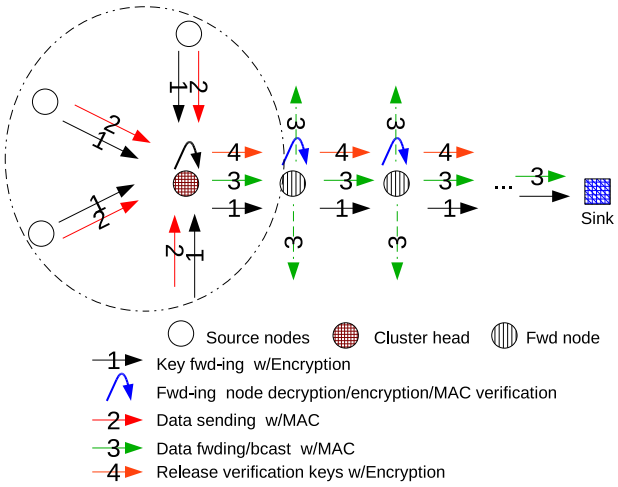

Figure 19: DEF.

difference is the downward slope with some of the plots. This is attributed to the fact that the ratio of the healthy traffic diminishes in this attack scenario as the number of bad packets increases due to the number of malicious nodes in the network. So, if a more secure application is desired or if the WSN application is deployed in an hostile environment, then VEBEK-I is recommended because VEBEK-I provides security services at every hop. VEBEK-I also watches fewer nodes in comparison to VEBEK-II. Thus, the lower storage requirement (i.e., fewer watched nodes) and providing security at every hop make VEBEK-I well suitable for military WSN applications where immediate reaction to enemy units is necessary. However, the downside of the VEBEK-I operational mode is its high processing costs. On the other hand, if the deployment region is expected to be a relatively safe environment, which may be true for some civilian WSN applications, then VEBEK-II can be utilized. But, as discussed above, to provide a comparable level of vigilance to the network, this operational mode uses much more storage than VEBEK-I.

3.6.5 Comparison of VEBEK-II with Other Statistical Schemes

In this sub-section, we evaluate the energy performance of VEBEK-II with other "en-route dynamic filtering" works in the literature. We focus on statistical schemes

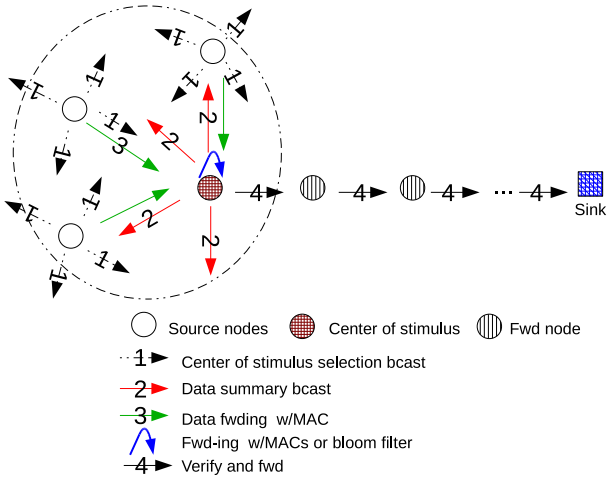

Figure 20: SEF.

because they have received a lot of attention in recent years. Specifically, we compare the expected energy costs of DEF [1], SEF [2], and STEF [3][4] with that of VEBEK-II because VEBEK-II is the statistical mode of the VEBEK framework. First, we briefly summarize each protocol and discuss their drawbacks. Then, the comparison results are presented. Illustration of each protocol are given in Figures 19−21.

In the Dynamic En-route Filtering scheme (DEF) by Yu and Guan [1], a legitimate report is endorsed by multiple sensing nodes using their own authentication keys. Before deployment, each node is preloaded with a seed authentication key and $l + 1$ secret keys randomly chosen from a global key pool. Before sending reports, the cluster head disseminates the authentication keys to forwarding nodes encrypted with secret keys that will be used for endorsing. The forwarding nodes stores the keys if they can decrypt them successfully. Later, cluster heads send authentication keys to validate the reports. The DEF scheme involves the usage of authentication keys and secret keys to disseminate the authentication keys; hence, it uses many keys and is complicated for resource-limited sensors.

Ye et al. proposed statistical en-route filtering (SEF) [2]. In SEF, each sensing

[4]Although STEF is not a statistical approach, we included in our comparison because it is a relevant en-route filtering study.

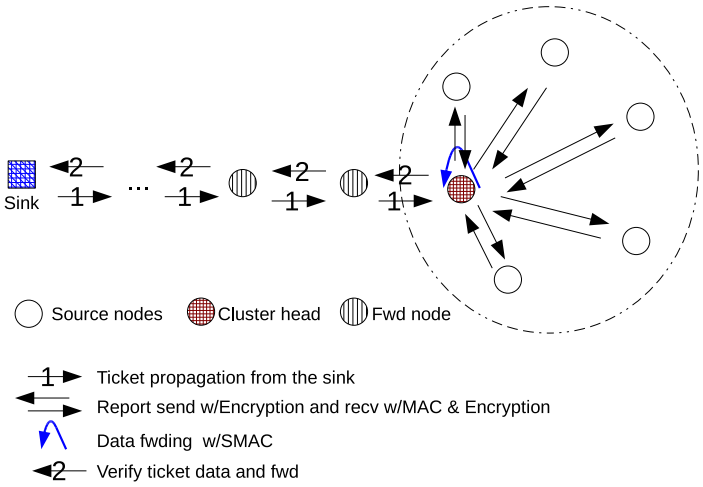

Figure 21: STEF.

report is validated by multiple keyed message authentication codes (MACs). Specifically, each node is equipped with some number of keys that are drawn randomly from the global key pool. First, a center of stimulus is selected among the source sensor nodes in the event region. Then, once a report is generated by a source node, a MAC is appended to the report. Next, another upstream node that has the same key as the source can verify the validity of the MAC and filters the packet if the MAC is invalid. However, the downside of SEF is that the nodes must store keys and packets are enlarged by MACs. Although the authors suggest the use of bloom-filters to decrease the MAC overhead, SEF is a static key-based scheme and it inherits all the downsides of static key management schemes.

The scheme, Secure Ticket-Based En-route Filtering (STEF) [3], by Krauss et al. proposes using a ticket concept, where tickets are issued by the sink and packets are only forwarded if they contain a valid ticket. If a packet does not contain a valid ticket, it is immediately filtered out. STEF is similar in nature to SEF and DEF. The packets contain a MAC and cluster heads share keys with their immediate source sensor nodes in their vicinity and with the sink. The downside of STEF is its one way communication in the downstream for the ticket traversal to the cluster head.

Since DEF and SEF are probabilistic schemes, a comparison of each scheme with

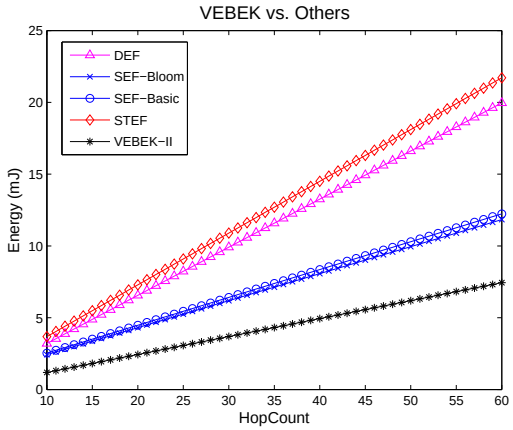

Figure 22: Comparison of VEBEK [4], DEF[1], SEF[2], and STEF[3].

VEBEK-II in terms of their energy consumption is presented in Figure 22. The results are generated for one round of communication from a source node to the sink, which is assumed to be located n hops away from the source node. The x-axis represents the hop count and is varied, while the y-axis is the energy. To simplify the comparisons, we assumed that all the nodes in DEF, SEF and VEBEK-II would have the necessary keying material with 0.7 probability to do the desired security features imposed by the specific protocol in a benign environment (no malicious nodes). We also assumed that the protocols that use hashing and encryption mechanisms would use MD5 and RC4, respectively. The real sensor implementation values for these crypto mechanisms are taken from [48] and [49]. Another necessary assumption was that all protocols would work in perfect communication cases without packet loss because only the VEBEK framework has been designed with handling communication error cases and it would not be meaningful to compare VEBEK with others when others were not designed to handle errors. As can be seen, VEBEK-II is better than all the schemes, exhibiting a performance improvement of 60%−100% in energy consumption than the closest scheme, SEF. We note that all other schemes provide a nice framework for filtering malicious data en-route; however, the other schemes exchange many messages, involve the use of many keys, and do not have any mechanism to cope with packet loss.

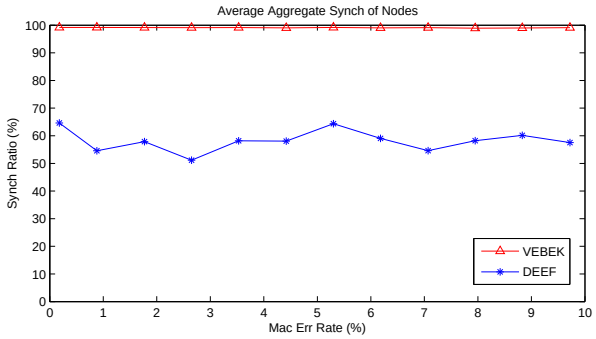

Figure 23: Synchronization ratio of nodes along the path to the sink.

Moreover, we analyze how VEBEK improves the synchronization problems that may occur due to communication errors in DEEF [13]. Since DEEF is based on generating communication keys with real battery levels, packet drops may cause the nodes to easily loose synchronization with other nodes along the path to the sink. To analyze the synchronization problem, we define *synchronization ratio* as a metric to measure the performance of the VEBEK framework during packet drops. Specifically, we denote the synchronization ratio, φ, as follows:

$$\varphi = \sum_{i=1}^{\eta_{h_w}} \frac{\gamma_i}{\gamma_i + \varepsilon_i} \tag{19}$$

where i is the node, γ is the number of forwarded-watched packets, ε is the number of dropped-watched packets, and η_{h_w} is the number of watcher nodes between the source and the sink. Figure 23 presents the simulation results of the synchronization ratio with respect to DEEF and VEBEK. As can be seen, VEBEK outperforms DEEF and it is able to keep its synchronization even in dire communication scenarios. The x-axis is the the percent of the packets that are dropped due to communication errors.

3.7 Benefits and Limitations of VEBEK

This section briefly summarizes several benefits and limitations of the VEBEK secure communication framework. The VEBEK framework has the following benefits:

- *No control messages for rekeying:* VEBEK does not exchange control messages

for key renewals as opposed to other dynamic key management schemes. Therefore, VEBEK is able to save more energy and is less chatty in nature.

- *One-time key:* In VEBEK, one key per message is employed. For the successive packets of the stream, different keys are used while the previous schemes use basically the same key for different packets. This dynamic nature also makes the VEBEK framework more resilient to certain attacks (e.g., replay attacks, brute-force attacks, masquarade attacks).

- *Modular architecture:* Since keys are generated in a separate module in VEBEK, other key-based encryption or hashing schemes can also be adopted easily.

- *Security in realistic communication cases:* In VEBEK, we acknowledge the fact that sensor networks would be deployed in hostile error-prone environments. Nonetheless, current state-of-the-art en-route malicious data filtering schemes do not consider communication errors in their architectures. This can also be observed with many of the dynamic and static WSN keying schemes. In designing VEBEK, we were motivated with this fact, and tried to provide security with communication in mind.

The VEBEK framework has the following limitations:

- *No insider threats:* The insider attacks are outside the scope of this work similar to [2]. Our future work will be based on addressing insider threats.

- *No rekeying due to key revocation:* The VEBEK architecture does not address rekeying as a result of key revocation.

- *Static sensors:* The VEBEK communication framework is designed without mobility in mind. The typical usage scenario for VEBEK is one where the sensors are dropped from an airplane or unmanned aerial vehicle (UAV) and would be fixed in the deployment region.

- *Fixed routing path during data delivery:* In VEBEK, we assume that the path is fixed during the delivery of the data. Although routing paths may change, we expect this not to be a very common case for most of the statically deployed sensors. We plan to address dynamic paths in our future work.

CHAPTER IV

TIME-BASED DYNAMIC KEYING AND EN-ROUTE FILTERING (TICK) FOR WIRELESS SENSOR NETWORKS

The conribution of this chapter is the introduction of the TIme-Based DynamiC Keying and En-Route Filtering (TICK) protocol for WSNs [7]. TICK is the second protocol proposed in this thesis that is based on the idea of sharing a dynamic cryptic credential. In TICK, nodes utilize the local time available onboard the sensors as the shared dynamic cryptic credential when creating dynamic keys. TICK is also an effective dynamic en-route filtering mechanism, where the malicious data is filtered out from the network.

The remainder of this chapter is organized as follows. A motivation for the TICK protocol is presented in Section 4.1. Related work is given in Section 4.2. An overview of the TICK scheme is explained in Section 4.3. A performance evaluation with simulations, an analytical analysis, and a comparison with other schemes are presented in Sections 4.4 and 4.5.

4.1 Motivation

One way to eliminate injected malicious data from WSNs is to utilize an *en-route-filtering* scheme as in [1, 2, 3]. The en-route-filtering schemes generally utilize keys generated by either *static* [10] and *dynamic* [11] key management schemes [6]. However, as discussed in Chapter 3 in greater detail, current dynamic key management and en-route-filtering schemes have significant downsides in terms of their energy efficiency.

Hence, motivated by the downsides of current dynamic key management and en-route-filtering schemes (see Chapter 3), and the fact that the communication cost is the most dominant factor in a sensor's energy consumption [14, 15], we tackle the problem of providing security to sensor-based applications with a new approach. Specifically, TICK uses the local time value of the node, where data is originated, as the dynamic key to encrypt the messages. Then, the receiving nodes on the path to the sink use their local time to successfully decode the timing key of the source node and verify the security of the packet. As time progresses, the subsequent transmissions use different time values to derive the key for the encryption mechanism, which increases the resiliency of the network against adversaries. Thus, the protocol avoids extra overhead of control messages. The nodes forwarding the data along the path to the sink are able to filter out the malicious data verifying its authenticity and integrity with the provision of non-repudiation. With TICK as in VEBEK, our main goal is to send events to the sink as energy-efficient and surreptitious as possible to reduce the likelihood of interception by an adversary. More importantly, we seek to minimize the overhead associated with refreshing keys to avoid them becoming stale.

Our novel approach using local clocks is well suitable for both WSNs and sensor-based CPS applications where utmost silence is necessary, like in military scenarios, as TICK is not "chatty" in nature. For instance, radio silence is very important for military operations as any radio transmission may reveal troop positions; so, restrictive EMCON[1] orders may be in effect [51]. Both analytical and simulation results verify the feasibility of the TICK framework. TICK is at least two times more energy efficient than other related schemes [1, 2, 3] as examined later in this chapter.

[1]EMCON: "The selective and controlled use of electromagnetic, acoustic, or other emitters to optimize command and control capabilities while minimizing, for operations security: a. detection by enemy sensors; b. mutual interference among friendly systems; and/or c. enemy interference with the ability to execute a military deception plan" [50].

4.2 Related Work

As also discussed in 3.2, en-route dynamic filtering of malicious packets has been the focus of several studies, including Dynamic En-route Filtering (DEF) by Yu and Guan [1], Statistical En-route Filtering (SEF), [2], and Secure Ticket-Based En-route Filtering (STEF) [3]. The brief details of these works as well as their performance are discussed in Section 4.5. However, the common downside of all these schemes is that they are complicated for resource-constrained sensors and they either utilize many keys or they transmit many keying messages in the network, which increases the energy consumption of WSNs. Another significant observation with all of these works is that a realistic energy analysis of the protocols was not presented. Moreover, VEBEK [4] and DEEF [13] utilize virtual energies and real battery levels to create dynamic keys, respectively. However, TICK is slightly different from these as the shared dynamic cryptic credential is based on usage of local clocks.

Furthermore, two pertinent studies based on associating keys with time information available in sensor nodes are presented in [52, 34]. In μTESLA [34], a broadcast authentication scheme is introduced utilizing the notions of loose-time synchronization and delayed key disclosure. However, sending keys as a separate message is not cost effective and keys may be lost due to communication errors. In fact, another worthwhile study [53] shows how μTESLA would be vulnerable to attacks due to its delayed key disclosure and loose-time synchronization concepts. On the other hand, in Time information-based Pre-deployed Secure Key Distribution (TPSKD) [52], time is used to create session keys between the communicating nodes. Several disadvantages exist in this study. First, the nodes still exchanges Δ_i (drift) values when establishing a pairwise session key with each other, the communication cost of the nodes is increased. Second, the scheme loads the sensors with a randomly chosen fixed Δ_i value initially and assumes the sensors will always drift with this static value. However, in reality, nodes may have different drift values due to the effects of different

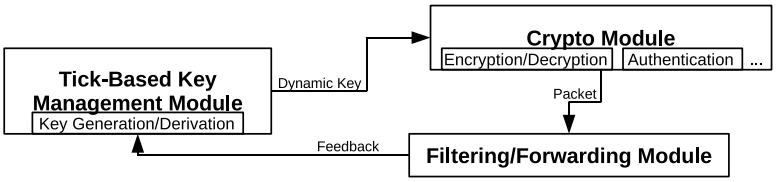

Figure 24: TICK Modules

environmental conditions around them.

In short, TICK is different from earlier studies in several ways: (1) TICK is a dynamic en-route filtering scheme that does not exchange explicit control messages for rekeying; (2) instead of using the same key multiple times, it provides one-time keys for each packet transmitted and hence avoiding stale keys; (3) TICK has a modular and flexible security architecture with a simple technique for providing authenticity, integrity and non-repudiation of data.

4.3 Overview of TICK

There are three main components of the TICK protocol: Time-Based Key Managemet (TKM), Crypto (CRYPT), and Filtering-Forwarding (FFWD) Modules. The TKM module is responsible for creation of the keys that will be used by the crypto module. The CRYPT module addresses the security part of the problem. Finally, the FFWD module filters the incoming decoded packet out of the network if it is classified as a bad packet or otherwise forwards it to the upstream nodes. The relevant modules are explained in the order they function in the TICK protocol and are shown in Figure 24.

4.3.1 Threat Model and Assumptions

Source nodes are synchronized and loaded with a network-wise initialization vector (IV) pre-deployment. The IV and local time information will be used to generate the initial and subsequent dynamic keys. Note that the sensor nodes *do not* have perfect clocks and over time the sensors' clocks gradually diverge from the real clock

value due to changes in the environmental conditions such as temperature, humidity, pressure, and vibration. In the worst case, they can accumulate up to several seconds of error per day [54]. Thus, the dynamic keys generated in TICK will change as a function of time and random drift. As such, the same event reported by different sources located nearby or separate events reported by different sources located elsewhere in the deployment region use different keys. In fact, this is an instrumental and desired property, which we refer to as *Spatio-Temporal* Security Property for WSN applications. Since the keys change dynamically due to time and drift, even if attackers are distributed throughout the WSN and can capture a significant amount of packets, they would not have enough packets encrypted with the same key to break the encryption because of this *spatial-temporal* property. Hence, this situation will increase the effort of brute-forcing by the adversary. Also, TICK does not incur a cost to discover which keys are shared between any two neighboring node (shared-key discovery phase [12]) because the nodes use the local time and tick information to create the dynamic keys. Nonetheless, using real clocks requires designing both a flexible and an error-cognizant scheme that would compensate for drifting clocks. This issue is investigated more in Section 4.4.

Similar to [2], we mainly consider the false injection and eavesdropping of messages from an outside malicious node; hence the insider attacks are outside the scope of this work. Moreover, we assume that attacks on clocks (e.g., pulse-delay (replay) and wormhole) are detected by the extra delay they will introduce into the network as in [55, 56, 57].

The sink is the ultimate terminating point and decision maker. Nodes are statically deployed with the same communication ranges. Note that more than one sink may exist in the network and more resources are available to the sink. Finally, the report (packet) size exchanged between the nodes is assumed to be fixed.

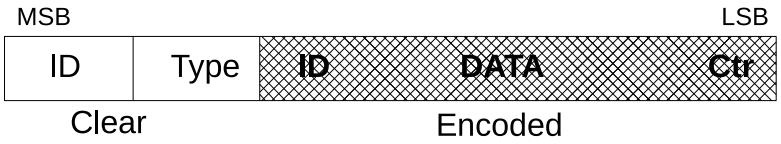

Figure 25: TICK packet structure.

4.3.2 Time-Based Key Management (TKM) Module

One of the primary contributions of TICK is the generation of keys dynamically using local time. This is addressed in the Time-Based Key Management (TKM) module. When a source node has data to send to the sink due to either an external stimulation by the sink [41] or a self-initiated periodic report, it uses its local clock value as the key. Specifically, the keys are a function of the current time value (t_l) and an initialization vector (IV) (i.e., $K_j^t \leftarrow F(t_l, IV)$). Dynamic local-time-based key generation algorithm is given in Algorithm 3. For example, assume in Figure

Algorithm 3 Compute Dynamic Local-Time-Based Key

```
1: ComputeDynamicTimeKey(t_l)
2: begin
3: j ← tx_cnt
4: K_j^t ← F(t_l, IV)
5: return K_j^t
6: end
```

26 the source node is $N1$, and the forwarder nodes $N2$, and $N3$ are on the path to the sink that the report by $N2$ will traverse. Note that $N1$ inserts a copy of its ID and a local counter value inside the report (packet) sent to the sink. The counter serves as a protection against replay attacks. It is increased each time a packet is sent from the source. The packet structure is illustrated in Figure 25. The ID is used to verify the integrity of the packet. As in Figure 26, $N1$ uses its local clock value *18* as the key. This key is used by the CRYPT module to perform the desired cryptic operations depending on the security service (e.g., encryption, authentication, integrity) provided by the WSN application. When $N2$ receives the report from $N1$,

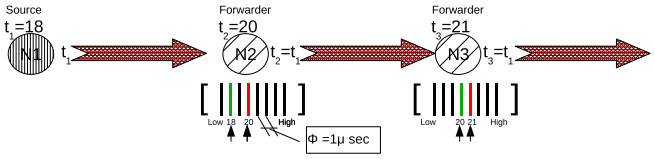

Figure 26: An illustration of packet delivery path.

it tries to find the value of the time at $N1$. First $N2$ substracts the approximate packet flight time ($\Theta = \rho + \tau + \varphi + \varepsilon$) between itself and $N1$ from its local time in order to be closer to the local time at $N1$, where ρ is the propagation time, τ is the packet transmission time, φ is the packet processing time, and ε is the approximation of errors for variability in transmissions due to fading, obstructions, and software errors, etc[2]. Furthermore, in order for a forwarder node to find the local clock value at the source node easily, all nodes are associated with a window of values, which we refer to as the *tick window, (T_w)* and a tick value (ϕ). Thus, $N2$ will try all values inside its tick window beginning from its local clock value. Once $N2$ finds the correct key value associated with the time at $N1$, using the found key, it will be able to perform other security actions on the packet in the crypto module and will also be able to compute the time offset from the sender. However, to combat against counterfeit values and to ensure a forwarder node does not futily attempt to brute-force all time-based keys, lower and upper bounds are associated with each node's tick window. Note that proper choice for the size of the tick window depends on, among other parameters, the tick value and it is explained more in the next section. The details of the key-derivation operation are given in Algorithm 4.

4.3.3 Crypto (CRYPT) Module

The CRYPT module obtains the dynamic key from the TKM module and performs the necessary security service. This is also the module where the key from the TKM is verified. If the key value received from the TKM module is not correct then

[2]A realistic analysis of the uncertainty associated with errors is presented in the next section.

Algorithm 4 Key Derivation at Forworder Nodes

```
1: DeriveKey()
2: begin
3: keyFound ← false; i ← 0; trialTime ← 0
4: Θ ← ρ + τ + φ + ε //packetFlightTime
5: startTime ← FetchLocalTime() − Θ
6: trialFwd ← TrialBack = startTime
7: while ((keyFound = 0)and(i <= thresHold)) do
8:    if ((i%2 = 1)and(i! = 0)) then
9:       trialFwd ← trialFwd + φ
10:      trialTime ← trialFwd
11:   end if
12:   if ((i%2 = 1)and(i! = 0)) then
13:      trialBack ← trialBack − φ
14:      trialTime ← trialBack
15:   end if
16:   if (i = 0) then
17:      trialTime ← startTime
18:   end if
19:   K = ComputeDynamicTimeKey(trialTime)
20:   timeDecoded ← RC4(K)
21:   if (ID_decoded = ID_Clr) then
22:      keyFound ← true
23:   end if
24:   i ← i + 1
25: end while
26: return keyFound
27: end
```

a new key is obtained from the the TKM module. This process continues until the correct key is found or the packet is discarded in the next filtering-forwarding (FFWD) module. The CRYPT module incorporates the RC4 algorithm into its body as the encryption mechanism. The rationale for choosing RC4 is due to its proven lightweight computational energy consumption on sensors [48, 49]. As each time a new dynamic key is fed into the RC4 block, it eliminates the risk of the differential cryptanalysis of the cipher [58]. Moreover, since the key is generated in another module, any desired encryption (e.g., DES, 3DES), authentication, or integrity mechanism (e.g., HMAC, CMAC) can be implemented together or separately depending on the security service

desired from the WSN application. After the correct value of the key used by the sender is determined by the current node, the offset value for the sender node is stored by the current node.

Two operational modes in the crypto module to determine how to forward the incoming packet can be conceived. In the first mode, *No-reEncode* mode, the original incoming packet is forwarded to the upstream node without any re-encryption whereas in the second mode, *reEncode* mode, the incoming packet is forwarded to the upstream node after re-encryption with the key associated with the local time at this receiver node. For *reEncode* mode, the forwarder node uses its current local clock value and IV value to create a new key when re-encryption the incoming packet. The advantage of the *No-reEncode* mode is one encryption computation, hence energy is saved by forwarding the original packet. This is the recommended mode of operation for TICK. However, if the current forwarding node is located too far away from the source node, the forwarding node may classify a healthy incoming packet as malicious. Specifically, this case occurs if the time difference between the local times of the source and the far-away node is bigger than the total time covered with $T_w * \phi$. Nonetheless, this is not an issue for *reEncode* mode because the forwarder nodes refresh the the key, used to encrypt the forwarded packet. Eventually, in both modes when the sink receives the report along the path, it also goes through the same intelligent key-finding procedure as forwarder nodes.

4.3.4 Filtering-Forwarding (FFWD) Module

The filtering-forwarding (FFWD) module in TICK is the module that filters the malicious data out of the network if the incoming packet is malicious or forwards the data toward the sink otherwise. Specifically, it receives the decision about the decrypted packet from the CRYPT module. If the packet is not malicious, then the original incoming packet is forwarded to the next hop sensor intact toward the sink.

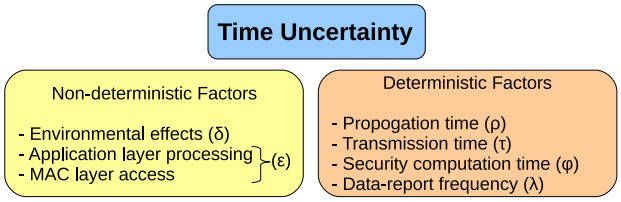

Figure 27: TICK uncertainity parameters.

Note that the original packet is not enlarged in any way (e.g., with MACs) to keep the energy costs at minimum as much as possible.

4.4 Time Uncertainty

Uncontrolled environmental conditions such as changes of temperature, humidity, pressure, and sudden vibrations in the deployment area cause internal clocks to gradually diverge from the real clock. Moreover, channel access time (at the medium access control layer) and send-time (including the time for preparing the packet at the application layer and passing it to the lower layers), can be considered as contributing to the unpredictable [56] clocks. In TICK, the environmental factors are captured with the parameter δ, which is the daily value of the drift per sensor given a deployment area, while the software-based factors are captured with ε. We adopt the values reported in [54, 55, 56, 57] for ε and δ. Deterministic factors, on the other hand, depend on more predictable parameters. In TICK, as in [55, 56], these include the transmission time of one packet (τ), the propagation delay (ρ), the packet processing time (φ) (e.g., due to cryptographic operations), and the average period of data from sensors (λ). The TICK uncertainity parameters used in coping with non-deterministic and deterministic factors are summarized in Figure 27.

The effect of all the factors are captured by the tick window, T_w, and it is the most significant parameter in dealing with the uncertainty in TICK. It provides a window of time values. However, even though the TICK protocol is designed with a flexible T_w mechanism, a quantitative analysis is still needed. Therefore, in this section, first

an analytical model is presented to investigate the relationship between the size of T_w and the tick value (ϕ). Then, a realistic tick window (T_w) value is derived considering the capabilities of today's wireless sensor devices. With its current treatment of the uncertainty, the TICK protocol is conceived as a software-based solution consorting with other approaches and suggestions in the literature [59, 57, 55].

4.4.1 The Choice of Tick Window T_w

As briefly mentioned previously, the tick window T_w is available for the receiver node to choose from to decode the received packets. The window has upper and lower boundary values. The efficacy of the TICK protocol depends on the size of this window because the larger the size of T_w, the more time it takes for a receiver to find the key. In TICK, the T_w value is basically a function of the tick value (ϕ). The smaller the value of the tick, the more keys could be tried by each sensor, hence T_w is larger and the accuracy of the scheme is increased. Also, from the sender's perspective, as the system becomes more precise (i.e., the smaller the tick value), the chance of using a different key per packet transmitted increases. As long as the frequency of the events (packets) is larger than $\frac{1}{\phi}$, the system will use a different key per packet. Hence, assuming that the sensor application sends its data periodically (or on the average) at certain time intervals to the sink [41, 60], T_w can be computed as

$$T_w = \frac{(\lambda + \rho + \tau + \varphi + \varepsilon) * \delta}{3600 * 24 * \phi} \tag{20}$$

where τ is the transmission time of one packet, $\tau = \frac{l}{R}$ with l and R being packet length and rate of the WSN link, respectively; ρ is the propagation delay, $\rho = \frac{\chi}{c}$ with χ and c being distance between the sensors and the speed of light in the medium, respectively; λ is the average period of data in between sensed reports sent from a sensor; φ is the packet processing time, ε is the physical transmission error; δ is the daily value of the drift per sensor given a deployment area; and ϕ is the desired tick

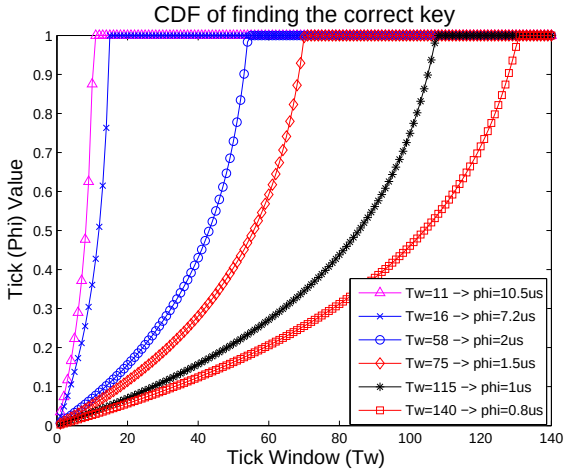

Figure 28: CDF of finding a correct key.

value. Note that (20) governs all the uncertainty factors into its body. Also, χ is taken based on the possible maximum distance to consider the worst case scenarios although nodes may be located closer than the maximum distance.

Moreover, the probability that k^{th} trial out of T_w keys is the first success is geometrically distributed with parameter p, where p is $\frac{1}{T_w}$. Hence, the probability that T_w keys are tried in a tick window is

$$P\{K = T_w\} = (1-p)^{T_w-1} * p, \qquad T_w = 1, 2, 3, .. \tag{21}$$

Analytical results governing equations (20) and (21) with $l = 32$ bytes, $R = 250Kbps$, $\lambda = 5$s, $\chi = 100m$, $\varphi = 558\mu$s [48], $\varepsilon = 10\mu$s [55, 57], and $\delta = 2$s are shown in Figure 28 for four different configurations of the tick window. For each T_w value, its corresponding ϕ value is also shown in the plot. As shown in Figure 28, the probability of success with a smaller value of T_w is greater, and therefore, less computational effort is required to guess the correct key of compared to when T_w is larger.

Several observations are possible with a close examination of (20). When sensors send less frequently to the sink, hence λ is larger, the value of the T_w becomes larger. This obviously increases the computational effort of the sensor to find the correct key. A similar remark can be made for ε as well. On the other hand, when λ is smaller

(i.e., more frequent data), the T_w is smaller. Hence, the scheme does not spend too much time trying to find the correct key; and the computational effort is smaller.

4.4.2 A Realistic Analysis of Tick Window (T_w) and Tick (ϕ) Value

A realistic value of the T_w considering the technical capabilities of today's wireless sensors is analyzed in this section. We see in equation (20) that more precision of ϕ comes with the cost of an increased number of keys that a sensor would try. TICK was designed to be energy efficient. If the computational effort of trying to find a key on a sensor is more than the communication cost of sending a separate keying message, then it may be better to send a separate keying message like other schemes in the literature. This depends on whether the benefit of a silent protocol is still desired at a cost of increased energy. Thus, in this part, we question what value of T_w is a plausible choice. In other words, can we derive a feasible T_w value given the capabilities of sensors today.

Assuming that in the worst case, a sensor will find the correct key at its last trial in the T_w window, the following inequality governs this case,

$$T_w * \zeta > \psi \tag{22}$$

where ζ is the computational effort of finding a correct key, and ψ is the transmission cost of the separate keying message. Thus, if the left side of the inequality is bigger than the right side, then the energy advantage of the TICK scheme (not other advantages like using one key, not "chatty") would become obsolete and one can conclude that sending a separate keying message would be better than using TICK. The transmission cost of the keying message can be written as follows [16] (ignoring energy cost of sensing the event and staying-alive for simplicity):

$$\psi = (I_{tx} + I_{on}) * \tau * V \tag{23}$$

$$\zeta = I_{on} * \mu * V \tag{24}$$

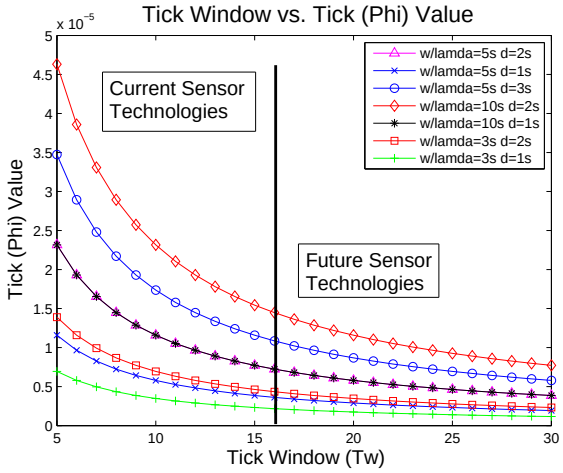

Figure 29: T_w (Tick window) versus ϕ (tick).

where μ is the execution time required to process the desired encryption algorithm, τ is the packet transmission time, I_{tx} and I_{on} are the current consumptions in mA for packet transmission and CPU processing, and V is the supply voltage of a given sensor node. Hence, an upper bound for T_w can be found as follows:

$$T_w \leq \frac{\psi}{\zeta} \tag{25}$$

Figure 29 plots T_w (Synch window) versus ϕ (precision) for several values of λ (average period of data) and δ (daily value of the drift per sensor given a deployment area). Assuming a sensor node with a microcontroller unit (MCU) of MSP430F16x [61] and a transceiver of CC2420 [62, 15, 16], and also assuming RC4 [48] as the encryption scheme, with $l = 32$ bytes, $R = 250Kbps$, $\lambda = 5$s, $\delta = 2$s, T_w can be found to be 16; hence, the tick value, ϕ, of 7.24μs. Thus, given the technical capabilities of sensors today, the value of T_w computed in this section is instrumental in making TICK a realistic protocol as much as possible and will be used in the performance evaluation section.

4.5 Performance Evaluation

In this section we evaluate the effectiveness of the TICK protocol both via simulations and analysis. First, a comparative study considering other similar works is given. Next, simulations results are presented to examine the energy efficiency of our scheme under normal operation and under attack. Note that an analysis for the filtering efficiency is not needed as in [4, 13] because in TICK, malicious packets are immediately taken out from the network at the next hop.

4.5.1 Comparison with Other En-route Filtering Schemes

In this sub-section, the energy performance of TICK is analytically compared with other relevant en-route filtering studies in the literature. Specifically, we compare the expected energy costs of Dynamic En-route Filtering (DEF) [1], Statistical En-route Filtering (SEF) [2], Secure Ticket-Based En-route Filtering (STEF) [3], and Time-based Predeployed Secure Key Distribution (TPSKD) [52] with that of TICK. Note that more detailed discussion of DEF, SEF, and STEF schemes as well as their downsides were given in Chapter 3. Although TPSKD [52] is essentially a time-based secure key pre-distribution scheme and it is not a en-route protocol per se, its main purpose is to create static pairwise keys between the nodes. Hence, it is included in our comparative analysis here as it also uses time information to create keys. The scheme initially loads the sensors with fake clock drift values (Δ). These values are then exchanged by the nodes to create pairwise link keys in the clear.

A comparison of each scheme in terms of their energy consumption is presented in Figure 30. The results are generated for one round of communication from a source node to the sink, which is assumed to be located n hops away from the source node. The x-axis represents the hop count and is varied, while the y-axis is the energy. To simplify the comparisons, we assumed that all the nodes would have the keying material with probability of 1 to do the desired security features imposed by

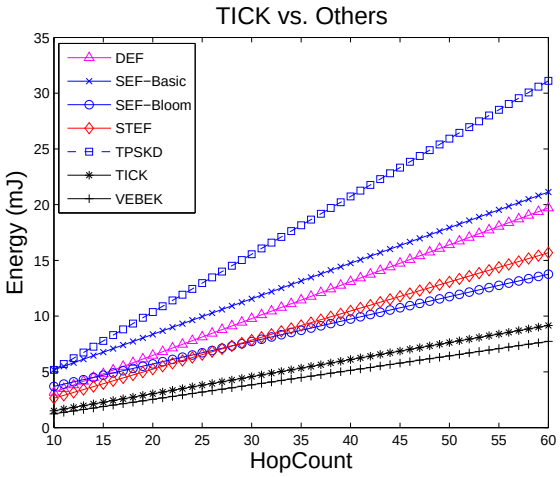

Figure 30: Comparison of TICK, VEBEK, DEF, SEF, STEF, and TPSKD.

the specific protocol in a benign environment (no malicious nodes). Without loss of generality, we assumed that all the schemes would use the same type of cryptographic mechanisms unless specified otherwise by the referenced work. Hence, we assumed that the protocols that use hashing and encryption mechanisms would use MD5 and RC4, respectively. The real sensor implementation values for these crypto mechanisms are taken from [48] and [49]. As can be seen, TICK is very energy-efficient compared to other schemes. The other schemes exchange keying messages and use many static keys. TICK eliminates these from its design and is able to save energy and reduce the opportunity for attackers to intercept packets. Another important observation that should be noted here is that the energy consumption profile of TICK is slightly higher than that of VEBEK because VEBEK does not try to find the key (time-based) associated with every transmitted packet. However, in general they are both energy-efficient schemes as they are protocols both built upon the concept of sharing a dynamic cryptic credentials.

4.5.2 Security and Energy Consumption Analysis

In this sub-section we evaluate the performance of the TICK protocol via simulations. We focus on the energy consumption of the TICK protocol while under attack.

Table 5: TICK Simulation Parameters

# of Nodes	500	SensSize	32 bytes	E_{ini}	5000 mJ	E_{dec}	3.3μJ
Area	1000x1000 m	RecvInterval	5s	E_{rx}	66.7μJ	E_{enc}	3.3μJ
Link Rate	250Kbps	SimTime	3000s	E_{tx}	59.6μJ	E_{mac}	8.6μJ
Range	75 m	#of Mal Node	(0..10)	E_{sens}	9μJ	E_{sa}	11.4μJ
# of Healthy Nodes	10	T_w	16	Time Offset	U$[-3,+3]\mu$s	Voltage	3V

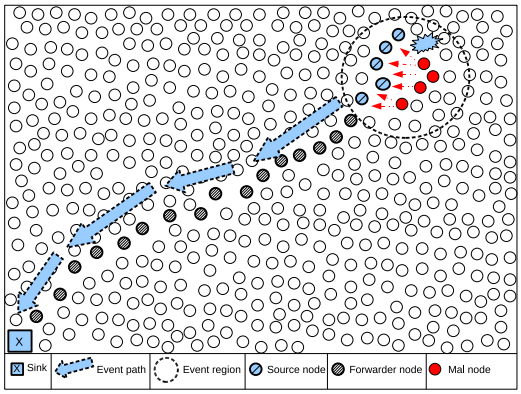

Figure 31: TICK simulation topology with GTSNetS.

4.5.2.1 Simulation Parameters and Assumptions

We use the Georgia Tech Sensor Network Simulator (GTSNetS) [47], which is an event-based sensor network simulator with C++, to perform the analysis of the TICK protocol. The topology and the parameters used are given in Figure 31 and in Table 6. Nodes were located randomly in the deployment region and on average, source nodes were 25−35 hops away from the sink. The energy costs for different operations in the table are computed based on the values given in [61, 16]. However, the costs for encryption and decryption operations are computed based on the the reported values of the implementation of RC4 [48] on real sensor devices. E_{tx}, E_{rx}, and E_{sens} are the energy consumption of sending, receiving a packet and sensing an event, while E_{enc}, E_{dec}, and E_{mac} are the costs of encryption, decryption, and the message authentication code, respectively. We use 16 as the value of T_w as found in the previous sub-section. Due to the broadcast nature of the wireless medium used in WSNs, attackers may try to eavesdrop, intercept, or inject false messages. In this work we mainly consider the false injection and eavesdropping of messages from an outside

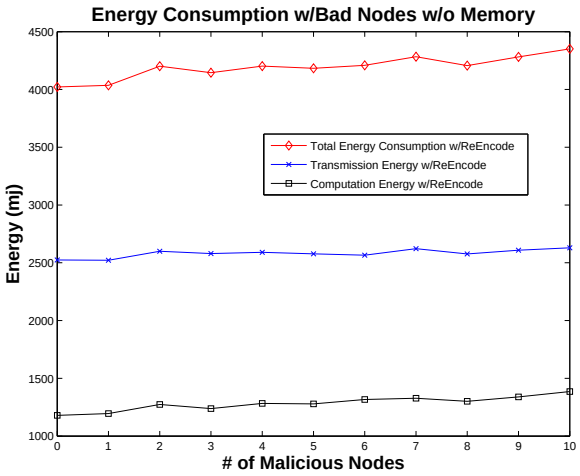

Figure 32: Computation, transmission, and total energy consumption under an attack scenario.

malicious node; hence similar to [2], the insider attacks are outside the scope of this work. In our attack scenario, the total number of healthy source nodes that collect the event information and send it toward the sink is assumed to be fixed, whereas the number of malicious nodes are increased over time. As in Figure 31, the malicious sensors are randomly located inside the event collection region. Throughout this work, the following additional assumptions are made: each node has its local clock and its drift value from the real clock is generated using a uniform distribution between -3 and +3 μs similar to [57]. The Directed Diffusion routing protocol [41] is used, but others such as [46] can also be used. According to specifics of Directed Diffusion, after the sink asks for data via *interest* messages, a routing path is established from the sources in the event region to the sink. Thus, we assume that the path is fixed during the delivery of a particular sensed event report. Sensors are assumed to have the same communication ranges and may have different initial battery supplies. Finally, the simulation results presented in the figures are the average of 50 simulation runs for a specific analyzed parameter.

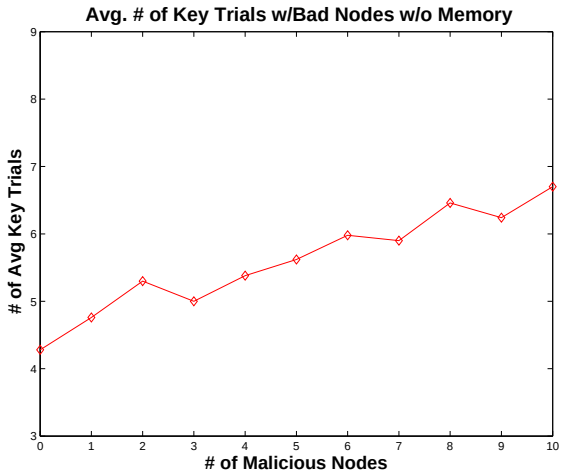

Figure 33: Avg. number of key-trials.

4.5.2.2 Simulation Results for Security and Energy Consumption

Figure 32 shows the results considering the aforementioned attack scenario. The x-axis represents the number of malicious nodes inside the region and y-axis respresents the energy consumption in mJ. We see that as the number of malicious nodes increases inside the network, nodes spend more computation energy. This happens because the number of nodes who use all their key-trial attempts and ultimately classifies a packet as malicious, increases with the increased malicious traffic.

4.5.2.3 Simulation Results for Key-trials

As explained in Section II, when a sensor receives a packet from another sensor, it tries to find the time-based key value associated with this packet used by the sender when encrypting the packet before sending. However, the total trial attempts is limited by the value of key window T_w, not to exhaust the resources onboard the sensor and the nodes immediately eliminate the malicious data from the network once they exhaust all their key-trial attempts. For this, we have used in our simulations a feasible value for T_w (16) given for today's sensor technology as we discussed in the previous section. With this in mind, it is also interesting to look at how many key-trial attempts on average that sensors uses in the simulations when attempting to

decrypt the received packets. We generally observe in Figure 33 that the increase of malicious activity in the network increases the efforts of the sensors. Since packets are dropped immediately when nodes exhaust all of their key-trial attempts, the system does not allow a malicious packet to get through the network. Also, one interesting result is that nodes do not use all the attempts; the highest point for our attack scenario was around 6.9.

CHAPTER V

SECURE SOURCE-BASED LOOSE TIME SYNCHRONIZATION (SOBAS) FOR WIRELESS SENSOR NETWORKS

As an application of the technique proposed in the TICK protocol, in this chapter the Secure SOurce-BAsed Time Synchronization (SOBAS) protocol for WSNs [8] is introduced. Instead of synchronizing each sensor globally, SOBAS focuses on ensuring that each source node is synchronized with the sink such that event reports generated by the sink are ordered properly. Hence, the objective of the SOBAS protocol is to provide a loose-time synchronization protocol for WSNs rather than a perfect synchronization among the nodes. Similar to TICK, SOBAS utilizes local time values to create one time dynamic keys. Therefore, SOBAS is largely built upon the components of the TICK protocol. SOBAS provides an energy efficient and an effective technique to securely synchronize the nodes on the data delivery path in the network, *without* the transmission of explicit synchronization control messages.

This chapter proceeds as follows. A motivation for the SOBAS protocol is given in Section 5.1. Related work is presented in Section 5.2 The architecture of the protocol is discussed in Section 5.3. The performance evaluation of SOBAS and the simulation results are presented in 5.4. Finally, a discussion of the benefits and the limitations of SOBAS is given in 5.6.

5.1 Motivation

Many applications that will utilize WSNs will require that event reports extracted from the network are received in the order that they were sensed. A common method

of properly ordering the reports processed by WSN applications is to utilize timestamps in messages/reports generated by clocks available onboard the sensors. However, changes of temperature, humidity, pressure, and vibration due to uncontrolled environmental conditions in the deployment area cause slight differences in clock frequencies and internal clocks gradually diverge from the real clock [63]. Thus, several useful studies [59, 64, 63] have tried to address the synchronization problem for WSNs. Additionally, adversaries may target the proper functioning of WSNs and disturb the critical decision-making processes by injecting false time information into the network. Therefore, a similar research effort to address the problem of secure synchronization for WSNs in the literature is also underway as in [56, 65, 57, 66].

One significant observation with all the synchronization solutions (regardless of being secure or not), is that they basically try to provide perfect synchronization for WSN applications. Furthermore, current schemes that provide secure perfect time synchronization exchange a significant amount of control messages (albeit necessary for their schemes) to synchronize the network securely, thereby increasing the communication costs of the network. When static or even dynamic pairwise key-based solutions [12] are used to secure the time information among nodes, nodes go through the processes of key discovery and key renewal. The energy cost, especially the communication cost, associated with these processes are often not discussed by researchers when building secure synchronization protocols.

However, *if the sole purpose of the WSN application is to collect data and send it to the sink, and if critical decisions are based on the data collected from the network (which is the case with most of the WSN applications [60, 67]), then what matters is that the sink properly orders the events before sending them to the application or external networks.* This can be achieved by synchronizing the sink with each source, and does not require that each sensor's clock be globally perfectly synchronized. In fact, global accurate knowledge of time [68] may generally suffice for many WSN

applications, where a centralized decision authority (not sensors) acts on the information collected from the network. Moreover, because the communication cost is the most dominant factor in a sensor's energy consumption [14, 15], if the synchronization control messages in the network are eliminated as opposed to current "chatty" schemes, some of the energy savings from transmission cost can be utilized for the computation of local security operations.

Therefore, motivated by the downsides of current schemes, considering the event-based characteristics of WSN applications and the resource-limited nature of sensors, and finally focusing on the energy consumption profile of sensors, we propose the **S**ecure S**O**urce-**BA**sed Loose Time **S**ychronization protocol. Essentially, SOBAS is a derivative of the TICK protocol for WSN applications that do not need perfect synchronization. SOBAS presents an effective technique to securely synchronize the data path in the network, *without* the transmission of explicit synchronization control messages. Instead of synchronizing each sensor globally as opposed to approaches providing perfect synchronization, we focus on ensuring that each source node is synchronized with the sink and nodes along the data delivery path such that event reports generated by the sink are ordered properly.

As in TICK, it uses the local time value of the node, where data is originated, as the dynamic key to encrypt the messages. Then, the receiving nodes on the path to the sink use their local time to successfully decode the timing key of the source node and verify the security of the packet. Thus, the protocol avoids extra overhead of synchronization messages. Eventually, when the sink receives the packet, it will associate each source with an offset value, Δ, and will be able to order the events reported from the WSN. As time progresses, similar to TICK, the subsequent transmissions use different time values as the key to the encryption mechanism, which increases the resiliency of the network against adversaries. Our main goal is to synchronize events at the sink as energy-efficient, precise, and surreptitious as possible to reduce the

likelihood of interception by an adversary. Furthermore, our goal is *not* to provide a pairwise perfect synchronization among the nodes as opposed to other secure time synchronization schemes like [56, 65, 57].

With SOBAS, we are able to achieve our main goal of synchronizing events loosely at the sink and at the data delivery path as quick, as accurate, and as surreptitious as possible. SOBAS is perfectly suitable for WSN applications that do not need perfect synchronization and it is able to provide 7.24μs clock precision on the data delivery path given today's sensor technology. Simulation results show that SOBAS is an energy efficient scheme under normal operation and attack from malicious nodes.

Our novel approach to clock synchronization is well suitable for WSN applications where utmost silence is necessary, like in military scenarios, as SOBAS is not "chatty" in nature. For instance, radio silence is very important for military operations as any radio transmission may reveal troop positions; so, restrictive EMCON orders may be in effect [50, 51, 69]. In this domain, further applications of interest include critical infrastructure monitoring [70], video-surveillance [71], and patient-data collection [72].

5.2 Related Work

Several useful studies exist surveying different insecure and secure time synchronization protocols [54, 73, 63]. In this section, we list several related works from the literature. First, we focus on secure ones, then briefly on insecure ones. Note that the energy performance and the drawbacks of some of the relevant protocols ([56, 52, 57]) were analyzed and discussed in Section 5.4.

Ganeriwal et al. proposed a suite of secure time synchronization protocols [56, 65], where pairwise single-hop, multi-hop, and group synchronization are addressed with a protection against pulse-delay attacks. However, these protocols require the nodes to go through the phase of key discovery with their pre-loaded static keys among themselves. Moreover, the protocols exchange many messages to synchronize

the network securely, both increasing the communication costs of the network and making them not applicable for military-type scenarios where a more surreptitious communication pattern may be preferred. Another work by Song et al. [66] focuses on delay attacks. Specifically, in lieu of using any cryptographic primitives, they address the problem based on the generalized extreme studentized deviate (GESD) algorithm and time thresholds. However, the drawback of this work stems from its statistical nature. In the work by Sun et al. [57], the authors propose a secure time synchronization protocol utilizing GPS devices starting from source nodes. The proposed work requires a shared static key between the communicating nodes and assume that the source nodes will be equipped with GPS devices, which is costly due to periodic communication to GPS satellites and increased radio activity increases the opportunity for malicious exploitation. Also, GPS may not be operational for all sensor applications (e.g., underwater medium) as explained in the previous section. Additionally, similar to [56], the nodes exchange many messages. In [74], authors provided a secure time synchronization protocol for heterogeneous sensor networks based on pairing (PBC) and identity-based (IBC) cryptography over elliptic curve. Although this is a novel adoption of IBC and PBC for time synchronization for WSNs, the work did not present any performance evaluation to provide clock precision values. In SecNav [75], a secure broadcast localization and time synchronization work is proposed for wireless networks (i.e., IEEE 801.11b devices not for WSNs) with Manchester coding, on-off keying, and public-private key systems using outside references. Lastly, another significant observation with all of these works is that an energy analysis of the protocols was not presented.

Two pertinent studies based on associating keys with time information available in sensor nodes are presented in [34, 52]. In μTESLA [34], a broadcast authentication scheme is introduced utilizing the notions of loose-time synchronization and delayed key disclosure. However, sending keys as a separate message is not cost effective

and keys may be lost due to communication errors. In fact, another worthwhile study [53] shows how μTESLA would be vulnerable to attacks due to its delayed key disclosure and loose-time synchronization concepts. On the other hand, in Time information-based Pre-deployed Secure Key Distribution (TPSKD) [52], time is used to create session keys between the communicating nodes. Several disadvantages exist in this study as shown earlier. First, the nodes still exchanges Δ_i (drift) values when establishing a pairwise session key with each other, the communication cost of the nodes is increased. Second, the scheme loads the sensors with a randomly chosen fixed Δ_i value initially and assumes the sensors will always drift with this static value. However, in reality, nodes may have different drift values due to the effects of different environmental conditions around them.

As for the insecure time synchronization WSN protocols, several works are of interest [68, 76, 77, 78, 79]. Li and Rus propose a combination of methods for global clock syhchronization [68]. All-node-based, cluster-based, fully-localized diffusion-based, and fault-tolerant diffusion-based methods are provided in the study. However, these methods either utilize two rounds of specific synchronization messages or depend on exchanging of explicit synchronization messages among the neighbor sensors. Also, no clock precision values are reported in the scheme to justify the claims in the work. In Sommer and Wattenhofer's work [77], gradient clock synchronization is proposed with the purpose of synchronizing the neighbor sensor nodes depending on explicit periodic neighbor broadcasts for synchronization from all the nodes. Tiny-Sync [78] provides two protocols for WSN time synchronization: mini-sync for pairwise node sychronization and tiny-sync for global time synchronization. These protocols assume and require bi-directional data transmission. In the best case, these protocols are able to achieve between 13μs and 18μs of clock precision. Ren and Lin propose a time synchronization scheme (self-correcting time synchronization (SCTS)) using reference broadcast messages. With SCTS, the authors show that they are able to achieve on

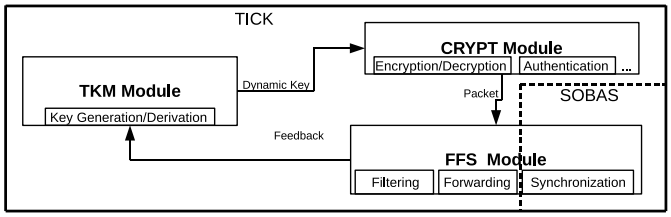

Figure 34: SOBAS's modular protocol architecture

the order of 1 or 2ms synchronization error within a single broadcast domain on real sensors. In [79], vector Kalman filter is adopted to provide a time synchronization protocol for WSNs combining the observations from multiple parents and achieves on average around 1μs global clock error. Although these are useful studies in their domains, none of these works analyze the energy performance of their respective schemes; so it is hard to understand their relative performance for the resource-limited sensor networks and they are not secure protocols by default.

5.3 Protocol Architecture

SOBAS is primarily derived based on the TICK protocol and it utilizes the similar modular protocol architecture presented in Chapter 4. However, since SOBAS is more tailored for synchronization operations, some additional features and functionalities are implemented in SOBAS as necessary. In this section, these new features and functions are described within the context of the modules. The relevant SOBAS modules as well as their interaction with those of TICK are shown in Figure 34.

5.3.1 Threat Model and Assumptions

Although SOBAS is primarily based on the same assumptions made in Section 4.3.1, several new assumptions are made in the SOBAS protocol as explained in this subsection.

The sink is perfectly synchronized with the outside world (e.g., via GPS). Regular sensors do not utilize GPS. Although, for some applications it may be necessary to

utilize a GPS receiver onboard the sensor device, our rationale for not utilizing it is as follows. First, mounting a GPS receiver on a regular sensor requires the sensor to operate in two different frequencies: one is in the ISM band (i.e., 2.4 GHz) for the regular sensor communication, the other is in the L band (e.g, 1575.42 MHz (L1)) for the communication with MEO (Medium-Earth-Orbit) satellites [80]. Regular periodic transmissions (i.e., every 30 seconds) to satellites will increase the cost of communication and, hence, the energy consumption [76]. Second, one of our design goals is to minimize the electronic emission footprint as much as possible to decrease the likelihood by an adversary. Last, there may be environments (e.g., under water) where traditional radio-based GPS receivers would not work [81, 82].

In SOBAS, our main goal is to synchronize and order events at the sink as energy-efficient, precise, and surreptitious as possible to reduce the likelihood of interception by an adversary. However, unlike [56, 65, 57], our goal is not to provide a pairwise synchronization among the nodes.

5.3.2 Time-Based Key Management (TKM) Module

The TKM module is responsible for creation of the keys that will be used by the CRYPT module. The keys are a function of the current time value (t_l) and an initialization vector (IV).

In SOBAS, functionalities of TKM module in TICK has been improved to provide stateful operations. Specifically, two operational modes regarding states for the TKM module are conceived in SOBAS for the incoming packets. The first mode, which we refer to as *Stateless Mode*, essentially governs the procedures explained in Section 4.3.2: Forwarder nodes along a path to the sink try to find keys ascociated with each received packet regardless of whether the forwarder nodes have already seen a packet from the same sender or not. Alternatively, the second mode, *Stateful Mode*, is a new addition to TKM module and able to provide further savings from the computation

with a small increased storage cost. Specifically, in the stateful mode, a receiver sensor can have a table for each sender sensor, where individual offset values for each sender is recorded. The next time the sensor receives a packet from the same sender, it will have a tick window (T_w) centered around the associated offset value for this sender. This makes the effort of the receiver easier when it tries to find the correct key for the sender. In the stateful mode, a sensor also remembers a previously seen malicious node.

5.3.3 Crypto (CRYPT) Module

The CRYPT module addresses the security part of the problem. Any desired encryption mechanism (e.g., RC4, DES, 3DES), authentication, or integrity mechanism (e.g., HMAC, CMAC) can be implemented together or separately depending on the security service desired from the WSN application. Both in SOBAS and TICK, RC4 encryption mechanism is adopted (see Section 4.3.3).

Three operational modes exist in the CRYPT module to determine how to forward the incoming packet. Two of these modes are inherited from TICK: *No-reEncode mode* and *Full-reEncode*. In SOBAS *Full-reEncode* mode, a forwarder node synchronizes itself loosely with the source as explained in the next module (FFS Module) and then uses this new local clock value when re-encoding the incoming packet. However, the third mode is specifically introduced in SOBAS to solve the problem of classifying a healthy incoming packet as malicious (aka false-positive), which may occur in the *No-reEncode mode* if the current forwarding node is located too far away from the source node.[1] The new mode is referred to as *Selective-reEncode* mode, where packets are selectively re-encoded or re-encrypted over some nodes along the data delivery path while these nodes are also loosely synchronized with the source as in *Full-reEncode*.

[1]Recall from Section 4.3.3 that this case occurs if the time difference between the local times of the source and the far-away node is bigger than the total time covered with $T_w * \phi$.

5.3.4 Filtering-Forwarding-Synch (FFS) Module

The FFS module filters the incoming decoded packet out of the network if it is classified as a bad packet by the CRYPT module or otherwise forwards it to the upstream nodes as explained in 4.3.4. In SOBAS, this module is also enhanced to include the synchronization process of the forwarder node with the source node along a data delivery path toward the sink.

More specifically, the FFS module in SOBAS is the module that synchronizes forwarder nodes with the source node in the *Full-reEncode* or *Selective-reEncode* modes of operation. At this module, the forwarder node gets the source's local clock value from the crypto module and updates its local clock value accordingly. For instance, in Figure 35, all nodes along the path to the sink update their clock values as $t_i = t_s + \Theta$ to synchronize themselves with the source (or sender), where $\Theta = \rho + \tau + \varphi + \varepsilon$ is the packet flight-time, comprising of ρ (the propagation time), τ (the packet transmission time), φ (the packet processing time at a given sensor after the receipt of the packet), and ε (the approximation of errors for variability in transmissions due to fading, obstructions, etc.)[2]. Therefore, a source-centric synchronization path is established up to the sink. The next time a packet from the same source travels over the same path, the nodes can put less effort in finding the proper time-based key. Note that in *Full-reEncode* mode all the nodes along the path do aforementioned operations whereas in *Selective-reEncode* mode only a certain fraction of the nodes (e.g., every third node) along the path do the operations.

Furthermore, eventually, when the sink receives the report along the synchronization path, it will be able to see how much a particular source has diverged from the real clock value by extracting the key associated with the source. Note that for this, the sink also goes through the same intelligent key-finding procedure as forwarder

[2]A realistic analysis of the uncertainty associated with errors was presented in Section 4.4.

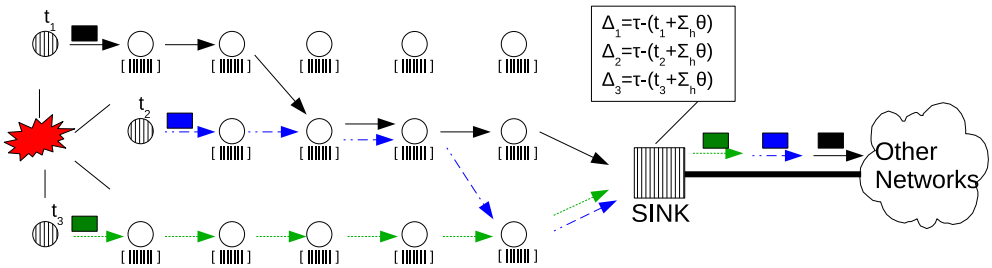

Figure 35: Synchronization of nodes on more than one synch path in SOBAS.

nodes. The sink calculates $\Delta_1 = t_r - (t_1 + \sum_h \Theta)$, with t_r being the real clock and h being the hop length. Thus, the sink can correct the timing of the reports coming from a particular source.

The synchronization described above synchronizes only one path according to a source (one synch path). However, in reality there are more synch paths because there may be more than one source in the deployed region and more than one sensor may be reporting the same event to increase the accuracy of the reports at the sink. Moreover, any node can be a forwarder and a source at the same time. We also note that there may be more than one sink collecting data from the sensors in the region. SOBAS handles this presence of multiple synch paths natively. It simply adheres to the logic of the synchronization of one path. In other words, when all the forwarder nodes receive a report from a source, or from another forwarder node, they simply forward the report directly or first synchronize themselves with the sender and then forward the report, depending on if they do the re-encoding operation or not. The advantage of this approach is that only one value is tracked. Since the sink(s) has more resources available, it can have a database of clock values for each sensor and their difference from the real clock. It can properly order any report from any source in the network using the Δ values associated with sources. The events received by the sink are placed in their proper order before leaving the sink for other networks. The case of multiple synch paths is illustrated in Figure 35.

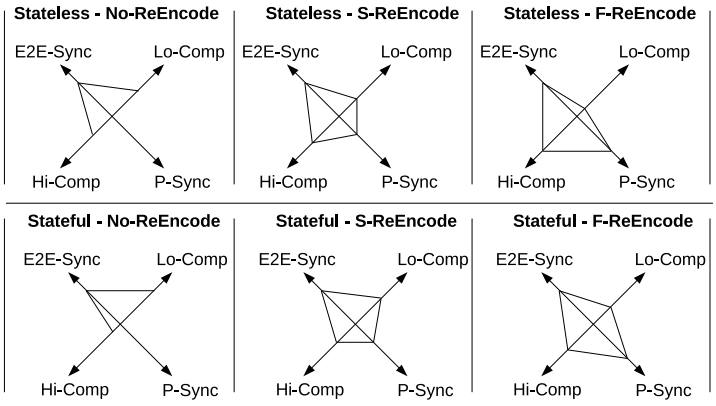

Figure 36: Summary of operational modes in SOBAS: Hi-Comp: High computation; Lo-Comp: Low computation; E2E-Sync: End-to-end loose synchronization; P-Sync: Loose Path synchronization.

5.3.5 Summary of Operational Modes

Six different operational modes exist in SOBAS depending on whether all or some of the nodes along the data delivery path or only the end-to-end nodes (the source and sink) are loosely-sychronized and whether the nodes have memories or not. These modes are summarized briefly below and illustrated in the kiviat diagram in Figure 36.

- ***Stateless-No-reEncode***: This is the default mode of operation in SOBAS where source nodes are synchronized loosely with the sink (end-to-end synchronization). A forwarder node was not designed to remember offset values (local time differences) of other sender nodes that were already discovered by this forwarder node. Also, a forwarder node would not remember a malicious node earlier discovered. However, as discussed earlier in Section 4.3.3 if the current forwarding node is located too far away from the source node, the forwarding node may classify a healthy incoming packet as malicious.

- ***Stateless-Selective-reEncode***: This is the mode of operation in SOBAS where some of the nodes are synchronized loosely with the source node along the data delivery path to the sink. This is introduced specifically to solve the

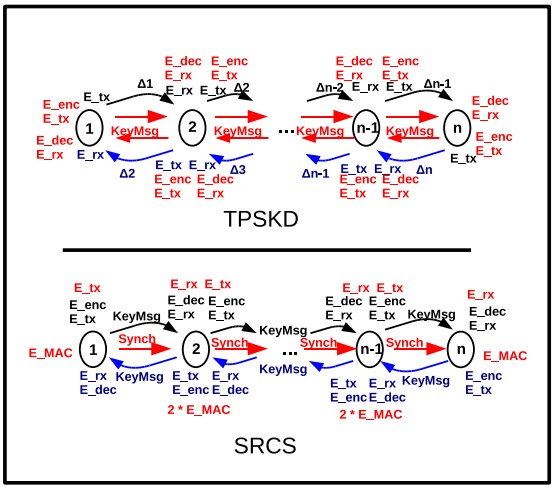

Figure 37: Illustrations of SRCS and TPSKD

problem, which may occur in the *No-reEncode mode*. Again, forwarder nodes do not have the capability (memory) to remember previously discovered offset values nor malicious nodes.

- ***Stateless-Full-reEncode***: This is the mode of operation in SOBAS where all nodes are synchronized loosely with the source node along the data delivery path to the sink. Forwarder nodes do not have the capability (memory) to remember previously discovered offset values nor malicious nodes.

- ***Stateful-No-reEncode***: In this mode, source nodes are synchronized loosely with the sink (end-to-end synchronization). Forwarder nodes were designed with the capability to remember offset values of other nodes that were already discovered. Also, they also remember a previously identified malicious node. False-classification of a healthy incoming packet as malicious may also occur in this mode.

- ***Stateful-Selective-reEncode***: In this mode, only some of the nodes are synchronized loosely with the source node along the data delivery path to the sink. Again, nodes have a capability to remember previously discovered offset values and malicious nodes.

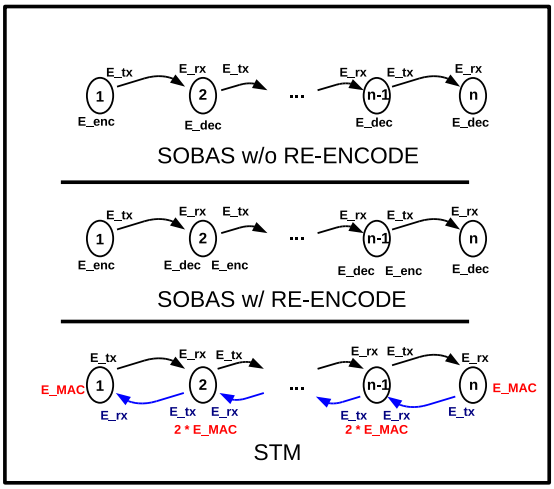

Figure 38: Illustrations of SOBAS and STM

- ***Stateful-Full-reEncode***: In this mode, all nodes are synchronized loosely with the source node along the data delivery path to the sink. Nodes have a capability to remember previously discovered offset values and malicious nodes.

5.4 Performance Evaluation

In this section we evaluate the effectiveness of the SOBAS protocol both via simulations and analysis. First, a comparative study considering other similar works is given. Second, using the realistic T_w value computed in the previous chapter (see 4.4) simulations results are presented to examine the energy efficiency of our scheme under normal operation and under attack.

5.4.1 Comparison with Other Schemes

In this sub-section, the energy performance of SOBAS is analytically compared with other relevant secure time synchronization studies in the literature. These works include Secure Transitive Multi-hop (STM) [56], Secure and Resilient Clock Synchronization (SRCS) [57], and Time-based Predeployed Secure Key Distribution (TPSKD) [52]. It should be emphasized that we are cognizant of the fact that these works aim

providing perfect synchronization in the network and that SOBAS provides loose-time synchronization. Thus, our comparison here is more hypothetical in nature to illustrate only the point that for some WSN applications employing a loose-time sychronization solution like SOBAS would be more energy efficient. In other words, further energy can be saved if nodes do not need perfect synchronization.

Each scheme is briefly summarized and the illustrations for each scheme are shown in Figures 37−38. Our comparison scenario is illustrated in these figures; it is assumed that node 1 synchronizes itself with node n, where n represents the number of nodes along a certain path. Each node goes through certain actions associated with the synch process and these actions are explained briefly below for each scheme. E_{tx} and E_{rx} are the energy consumption of sending and receiving a packet, while E_{enc}, E_{dec}, and E_{MAC} are the costs of encryption, decryption, and the message authentication code (MAC), respectively. Without loss of generality, we assumed that all the schemes would use the same type of cryptographic mechanisms unless specified otherwise by the referenced work. Since these works do not specify the type of cryptographic primitive operations in their respective publications, we have chosen MD5 for MAC operations and RC4 for encryption for simplicity and fairness. The real sensor implementation values for these crypto mechanisms are taken from [48, 49].

STM is one of the protocols proposed in [56]. It is used for the purpose of synchronizing nodes securely in a multi-hop fashion. In STM [56], node 1 first sends a notification for its desire to be synchronized with node n. This initial message is sent in the clear to node n. These messages are shown in the upstream direction and their associated costs are shown above the nodes. Then, after node n receives the notification message, it uses its static pairwise key that it shares with its downstream node *n-1* to create a MAC. Next, node n initiates a reverse path back to node 1 in the downstream direction to finalize the synch process with node 1. In addition to communication costs, each node creates a separate MAC for its downstream node and

verifies the MAC from the upstream node using static pairwise shared keys. These are shown in the reverse path below each node in Figure 38. In general, STM requires $4N$ messages for secure synchronization and has $N * 10\mu$s precision where N is the number of nodes.

SRCS is proposed in [57]. In this scheme it is assumed that source nodes would be equipped with global positioning system (GPS) devices, which may not be applicable in all WSN usage scenarios because "GPS is not suitable for sensor networks because of complexity and energy issues, cost efficiency, limited size, and so on" [76]. Similar to STM, the SRCS study also assumes unique static pairwise keys between the nodes. Furthermore, the study suggests several other predistribution schemes to create the static pairwise keys. However, as shown in Figure 38, creating the keys between the nodes using some other predistribution scheme also creates a burden on the sensors. In Figure 38, KeyMsg are the messages that are used for the key discovery process of the specific predistribution scheme as suggested by the scheme. In addition to communication costs, each KeyMsg has encryption and decryption costs on the sensors as shown in the figure. After pairwise keys are established among nodes, a synch message is sent from the source node 1 to other nodes in the upstream. This message is protected with an authentication mechanism using the pairwise keys. Associated communication costs and authentication costs are drawn on the nodes. Moreover, in SRCS, there are two operational modes, called level-based and diffusion-based clock synchronization. In general, to provide synchronization for 200 nodes, the SRCS scheme exchanges around 400-500 messages for level-based mode with an average of 2μs precision, whereas for diffusion-based mode, around 50000 messages are exchanged with an average of 6μs precision.

Finally, TPSKD [52] is essentially a time-based secure key pre-distribution scheme and it is not a synchronization protocol per se. Its main purpose is to create static pairwise keys between the nodes. It is included in our comparative analysis here as

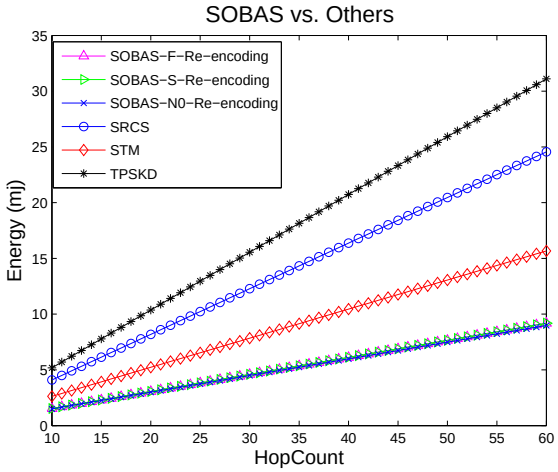

Figure 39: Comparison of SOBAS, STM, SRCS, and TPSKD.

it also uses time information to create keys. The scheme initially loads the sensors with fake clock drift values (Δ). These values are then exchanged by the nodes to create pairwise link keys in the clear. For our scenario, first pairwise keys are created by exchanging the Δ values among the nodes as shown in Figures 37−38. Then, an end-to-end path key is generated similar to creation of the link keys via KeyMsgs. All the associated communications and required cryptographic operations are shown on each sensor similar to the other schemes in this figure.

A comparison of each scheme in terms of their energy consumption is presented in Figure 39. The results are generated for one cycle of synchronization given the scenario from node 1 to node n while the hop count on the x-axis is varied as the value of n. As can be seen, SOBAS is very energy-efficient compared to other schemes. Furthermore, SOBAS uses time information to create a dynamic key as opposed to other schemes. This makes SOBAS more resilient to attacks. Its precision with today's sensor technology, as presented earlier in 4.4, is 7.24μs, which is better than STM and even better than SRCS. However, SRCS exchanges redundant messages and assumes a source-based GPS device, which may not be always available. Furthermore, it should be noted that in this comparative analysis, the energy cost of extra communications to GPS satellites, staying alive, and that to power GPS is neglected. SOBAS exhibits at

least twice better performance in energy consumption than the closest scheme, STM. If SOBAS had the same energy consumption level as STM (albeit not realistic with today's sensor technology), then it would be able to provide a better precision level of 3.62μs. In short, the other schemes exchange messages and use many static keys. SOBAS eliminates these from its design and is able to save energy and reduce the opportunity for attackers to intercept packets. However, again, SOBAS only provides loose synchronization, not perfect nor global network-wide synchronization. SOBAS is suitable for WSN applications where perfect synchronization is not needed.

5.4.2 Security and Energy Consumption Analysis

In this sub-section we evaluate the performance of the SOBAS protocol via simulations. We focus on the energy consumption of the SOBAS protocol while under attack.

5.4.2.1 Assumptions, Threat Model, and Simulation Parameters

We use the Georgia Tech Sensor Network Simulator (GTSNetS) [47], which is an event-based sensor network simulator with C++, to perform the analysis of the SOBAS protocol. The topology and the parameters used are given in Figure 40 and in Table 6.

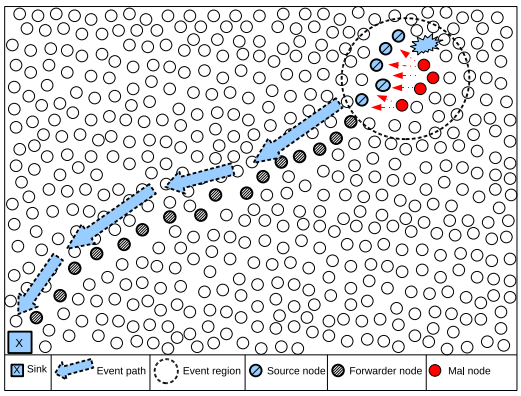

Figure 40: SOBAS simulation topology with GTSNetS.

Table 6: SOBAS Simulation Parameters

# of Nodes	500	SensSize	32 bytes	E_{ini}	5000 mJ	E_{dec}	3.3μJ
Area	1000x1000 m	RecvInterval	5s	E_{rx}	66.7μJ	E_{enc}	3.3μJ
Link Rate	250Kbps	SimTime	3000s	E_{tx}	59.6μJ	E_{mac}	8.6μJ
Range	75 m	#of Mal Node	(0..10)	E_{sens}	9μJ	E_{sa}	11.4μJ
# of Healthy Nodes	10	T_w	16	Time Offset	U$[-3,+3]\mu$s	Voltage	3V

Nodes were located randomly in the deployment region and on average, source nodes were 25−35 hops away from the sink. The energy costs for different operations in the table are computed based on the values given in [61, 16]. However, the costs for encryption and decryption operations are computed based on the the reported values of the implementation of RC4 [48] on real sensor devices. E_{tx}, E_{rx}, and E_{sens} are the energy consumption of sending, receiving a packet and sensing an event, while E_{enc}, E_{dec}, and E_{mac} are the costs of encryption, decryption, and the message authentication code, respectively. We use 16 as the value of S_w as found in Section 4.4. Due to the broadcast nature of the wireless medium used in WSNs, attackers may try to eavesdrop, intercept, or inject false messages. In this paper we mainly consider the false injection and eavesdropping of messages from an outside malicious node; hence similar to [56, 65], the insider attacks are outside the scope of this paper. In our attack scenario, the total number of healthy source nodes that collect the event information and send it toward the sink is assumed to be fixed, whereas the number of malicious nodes are increased over time. Letting i be the number of healthy source nodes and j be the number of malicious nodes, in our attack scenario, $j \leq i$, where $i = n$ and $n > 0$. The malicious sensors are randomly located inside the event collection region. We use in-line filtering to remove the malicious data as in [1, 2, 3, 13]. Throughout this work, the following additional assumptions are made: each node has its local clock and its drift value from the real clock is generated using a uniform distribution between -3 and +3 μs similar to [57]. The Directed Diffusion routing protocol [41] is used, but others such as [46] can also be used. According to specifics of Directed Diffusion, after the sink asks for data via *interest* messages, a

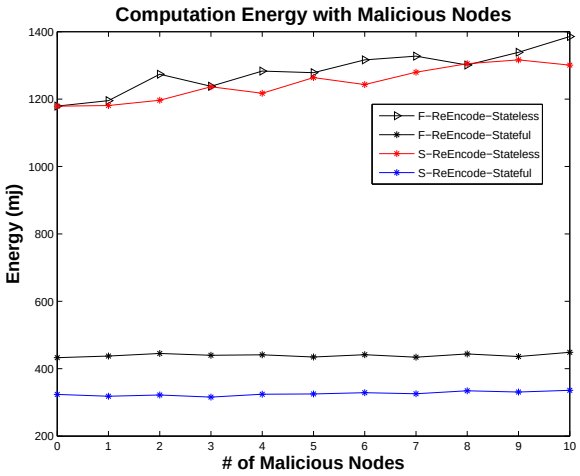

Figure 41: Computation cost under an attack scenario.

routing path is established from the sources in the event region to the sink. Thus, we assume that the path is fixed during the delivery of a particular sensed event report. Sensors are assumed to have the same communication ranges and may have different initial battery supplies. Finally, the simulation results presented in the figures are the average of 50 simulation runs for a specific analyzed parameter.

5.4.2.2 Simulation Results for Security and Energy Consumption

As mentioned in Section 5.3, there are three different modes for the CRYPT and two operational modes for the TKM modules. Modes for the CRYPT module are *No-reEncode*, *Selective-reEncode*, and *Full-reEncode* modes, whereas for the TKM module they are *Stateless* and *Stateful*. Figures 41−43 show the results for the attack scenario considering 4 different modes of operations except for the No-reEncode mode[3]. The x-axis represents the number of malicious nodes inside the region and y-axis respresents the energy consumption in mJ. As seen from the figures, the computation cost (i.e., E_{enc}, E_{dec}) for SOBAS-Selective-reEncode is less than that of SOBAS-Full-reEncode when Stateless mode is used. This is the direct result of the encrypting, decrypting,

[3]Note that this mode is specifically not included in the analysis because a healthy packet may be classified as malicious depending on the time difference between the local times of the source and the far-away node (see Section 4.3.3).

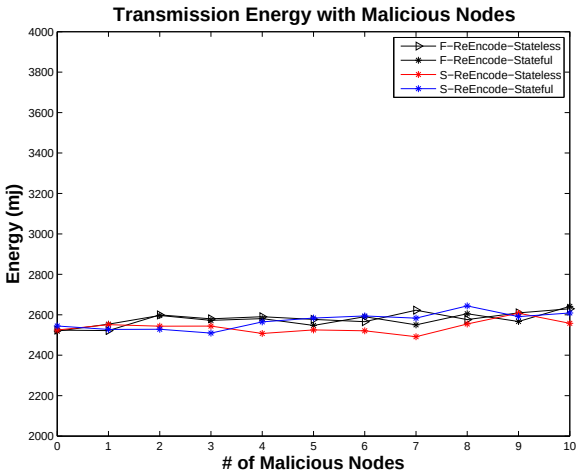

Figure 42: Transmissions cost under an attack scenario.

and re-encrypting of packets at every hop in the network for SOBAS-Full-reEncode. The cost of one encryption computation is saved by forwarding the original packet in SOBAS-Selective-reEncode while selectively re-encoding packets over some nodes along the data delivery path. The Selective-reEncode mode of operation in the crypto module may be suitable for further limited networking deployments. Nonetheless, SOBAS-Full-reEncode is more suited for networks with more networking resources available. From the security stand-point, we see that as the number of malicious nodes increases inside the network, nodes spend more computation energy in the Stateless mode. This happens because the number of nodes who use all their key-trial attempts and ultimately classifies a packet as malicious, increases with the increased malicious traffic. Furthermore, as seen from Figure 41 it is possible to save a significant amount of energy if statefull mode of operation is utilized. However, the stateful mode would depend on the availability of more storage on sensors.

As for the transmission costs (i.e., E_{tx},E_{rx}), all modes are about the same because in all modes the same number of packets are transmitted or received. Moreover, analyzing the results for the total energy consumption, we see that the total energy consumption in the network exhibits a similar behavior as the computation costs. This is because the overall energy consumption is greatly dominated by the computation

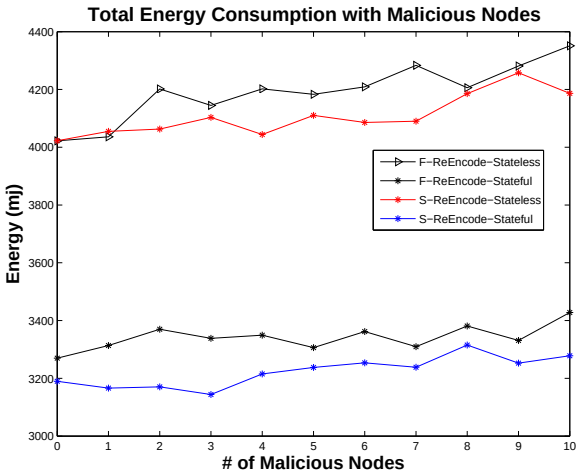

Figure 43: Total energy consumption under an attack scenario.

costs.

5.4.2.3 Simulation Results for Key-trials

As explained earlier, when a sensor receives a packet from another sensor, it tries to find the time-based key value associated with this packet used by the sender when encrypting the packet before sending. However, the total trial attempts is limited by the value of synch window T_w, not to exhaust the resources onboard the sensor. For this, we have found and used in our simulations a feasible value for T_w (16) given for today's sensor technology (see 4.4). With this in mind, it is also interesting to look at how many key-trial attempts on average that sensors uses in the simulations. Thus, in this part of the simulation results, we discuss the average number of key-trial attempts by a sensor when attempting to decrypt the packet when both stateful and stateless operational modes are considered. We observe that nodes do not use all the attempts with both modes; for the stateless mode of operation, the highest point for our attack scenario was around 6.9 (Figure 44). In general, we see that nodes with SOBAS-Selective-reEncode use more key-trials than SOBAS-Full-reEncode in the stateless mode. This is mainly due to the fact that refreshing a packet with a new key based on the current local time at a node decreases the effort of the next hop

node. On the other hand, comparing the stateful and stateless operational modes, one can easily observe that the number of key-trial attempts with stateful operation is significantly smaller than that of stateless operation. This is the direct result of sensor's ability to remember individual offset values for each sender sensor and malicious nodes. Keeping states makes the effort of the receiver sensor easier when it tries to find the correct key for the sender.

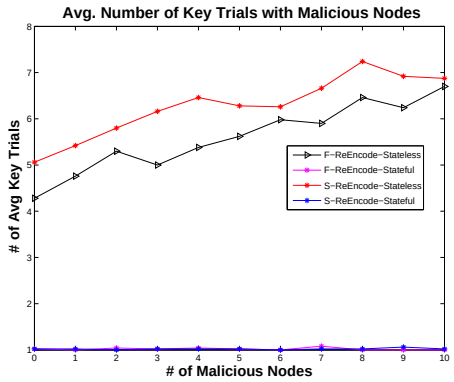

Figure 44: SOBAS Avg. key trials.

In fact, one implication of these results for both the operational modes is that since nodes do not use all of their key trial attempts, the remaining effort can be utilized to increase the clock precision value. For instance, if on average half of the T_w is used with a certain ϕ, then $2 * \phi$ clock precision can be achieved by increasing the effective size of the T_w. Therefore, in our future work, we will study further opportunities for increasing the clock precision by investigating unused key-trial attempts at a node. For instance, a node can adaptively increase its T_w.

5.5 Impact of Selective Re-Encoding

In this section we analyze the impact of the selective-reEncoding operation in the network by investigating the false-positive rate in the system.

The impact of selective-reEncoding was studied by considering reEncodig operation at different hops along the data delivery path to the sink. Specifically, three

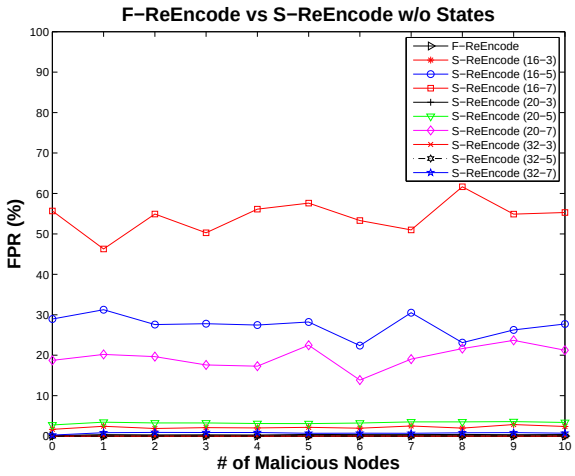

Figure 45: Stateless SOBAS FPR analysis under an attack scenario.

different cases were analyzed: reEncoding at every 3rd, 5th, and 7th nodes. Moreover, as discussed earlier (see Section 4.4), given the technical capabilities of sensors today, the value of T_w is 16. However, with near-term improvements in technology, further $T_w s$ that can provide better precision values are possible. Hence, we also include in our analysis different values for $T_w s$ as 16, 20, and 32. For all these different cases, further simulations were carried out. In Figures 45 and 46, the x-axis represents the number of malicious nodes inside the event region and y-axis respresents the false-positive-rate (FPR) in the system.

One general observation is that the more frequent the packets are re-encoded the smaller the rate of the FPR is in the network. This occurs because frequent refreshing a packet with a new key makes it easier to find the key at the next hop node. This situation also explains the superior performance of Full-reEncoding over other configurations. Also, with near-term improvements in technology, it is possible to achieve small FPR values, which are even close to values in Full-reEncoding mode of operation. Finally, overall, we observe similar behaviors for both stateless and stateful operations.

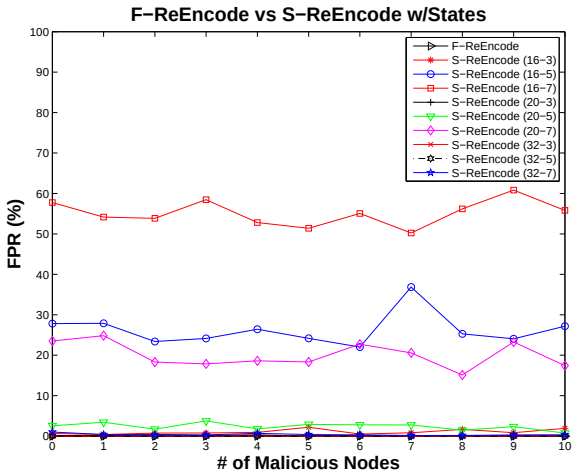

Figure 46: Stateful SOBAS FPR analysis under an attack scenario.

5.6 Benefits and Limitations

In this section, we list some of the appealing features of the SOBAS protocol and its limitations.

- *Dynamic*: Although nodes are pre-loaded with initial IV values, SOBAS is not a static key-based scheme because the keys change dynamically based on local clock increments on nodes and nodes do not go through the phase of key discovey like other static key-based schemes. Rather, it is a dynamic protocol where keys are changed over time. This makes SOBAS more applicable for many WSN applications that are hard to recover and replace after the initial deployment.
- *Resilient*: The dynamic nature of SOBAS makes it resilient against attackers because each message is encrypted with a different clock value as the key.
- *No synch messages*: The network is synchronized as events occur; separate synch control messages are not sent in the network.
- *No reference point*: No reference point is used in SOBAS to synchronize the nodes. This increases the overall security of the network while it eliminates the problem of a single-point-of-failure.

- *No key exchange*: In SOBAS, key values are not exchanged between the nodes. Nodes can intelligently discover keys through the mechanics of the scheme.
- *Simple and robust*: SOBAS can easily operate on sensors as its operations are not complicated.
- *No extra hardware*: SOBAS does not require extra hardware like GPS for synchronization. It utilizes the default hardware of the sensors as GPS is not suitable for sensor networks because of complexity and energy issues, cost efficiency, limited size, and so on [76].
- *Flexible architecture*: Since the key generation step in SOBAS is unbundled from other security mechanisms, dynamically created keys can be fed into any encryption mechanism (e.g., DES encryption). Thus, SOBAS provides a flexible architecture depending on the desired security level in the network.
- *Independent of underlying protocol*: SOBAS can be implemented with any underlying routing or MAC protocol.

Several limitations of the SOBAS scheme are as follows:

- SOBAS only aims providing loose time synchronization, not a perfect synchronization among the nodes as opposed to other schemes like [65, 56, 57].
- SOBAS is able to achieve our main goals to synchronize events at the sink as energy-efficient, precise, and surreptitious as possible. The protocol decreases the number of possible opportunities for malicious entities by reducing the number of control messages exchanged (completely eliminating that for synchronization). However, given the technical capabilities of sensor devices today, SOBAS is able to provide 7.24μs clock precision. This will improve as sensor technology improves.
- Similar to [56, 65], only outsider threats are addressed in this work. Our future work will address insider attacks in an energy efficient manner.

CHAPTER VI

CONCLUSIONS AND FUTURE WORK

In this thesis, efficient and secure communication frameworks have been developed for WSN applications. Motivated by the downsides of current dynamic key management and en-route-filtering schemes, the fact that the communication cost is the most dominant factor in a sensor's energy consumption [14, 15], and further building upon the concept of sharing a dynamic cryptic credentials, security to sensor-based applications was addressed using a new approach. As opposed to other "chatty" dynamic key management and en-route filtering schemes, the focus was on eliminating specific control messages for keying or rekeying in the network so that some of the energy savings from transmission cost could be utilized for the computation of local security operations. The following four areas are investigated under this thesis and conclusions for each of them are described in the following subsections:

- Designing Secure Protocols for Wireless Sensor Networks
- Virtual Energy-Based Encryption and Keying (VEBEK) protocol for Wireless Sensor Networks
- TIme-Based Dynamic Keying and En-Route Filtering (TICK) for Wireless Sensor Networks
- Secure SOurce-BASed Loose Time Synchronization (SOBAS) for Wireless Sensor Networks

6.1 Research Contributions

6.1.1 Designing Secure Protocols for Wireless Sensor Networks

Both WSNs and the security for WSNs research fields have matured over the years. Furthermore, optimization of the limited resources has motivated new research directions in the field. In Chapter 2, considering the established concepts and new directions, general principles of designing secure WSN protocols were presented for researchers. Specifically, the desired security services, i.e., confidentiality, authentication, integrity, access control, availability, and nonrepudiation, and their necessity from the WSN perspective was reviewed. Several valuable suggestions for protocols builders were determined and listed.

6.1.2 Virtual Energy-Based Encryption and Keying (VEBEK) protocol for Wireless Sensor Networks

As emphasized multiple times previously within the context of this thesis, communication is very costly for WSNs and for certain WSN applications. Independent of the goal of saving energy, it may be very important to minimize the exchange of messages (e.g., military scenarios). To address these concerns, in Chapter 3, a secure communication framework for WSNs called ***V***irtual ***E***nergy-***B***ased ***E***ncryption and ***K***eying (VEBEK) was developed. It was the first protocol in this thesis that is based on the idea of sharing a dynamic cryptic credential and the residual virtual energy of the sensor was used intelligently as a dynamic cryptic credential.

In comparison with other key management schemes, VEBEK has the following benefits: (1) it does not exchange control messages for key renewals and is therefore able to save more energy and is less chatty; (2) it uses one key per message so successive packets of the stream use different keys - making VEBEK more resilient to certain attacks (e.g., replay attacks, brute-force attacks, masquerade attacks); and (3) it unbundles key generation from security services, providing a flexible modular architecture that allows for an easy adoption of different key-based encryption

or hashing schemes. Moreover, VEBEK is an effective dynamic key-based en-route filtering scheme.

VEBEK's feasibility and performance were evaluated through both theoretical analysis and simulations. The results showed that different operational modes of VEBEK (I and II) could be configured to provide optimal performance in a variety of network configurations depending largely on the application of the sensor network. The energy performance of our framework with other en-route malicious data filtering schemes was also presented in the chapter. Our results showed that VEBEK performed better (in the worst case between 60%−100% improvement in energy savings) than others while providing support for communication error handling, which was not the focus of earlier studies.

6.1.3 TIme-Based Dynamic Keying and En-Route Filtering (TICK) for Wireless Sensor Networks

TICK is the second protocol developed based on the idea of sharing a dynamic cryptic credential and presented in Chapter 4. In TICK, nodes use their local time values as the dynamic cryptic credential. Again as in VEBEK, the focus was on eliminating sending explicit keying or rekeying messages as opposed to current "chatty" schemes. More specifically, a one-time dynamic key based on local time is used to encrypt each message. The receiving nodes use their local time to intelligently decode the timing key of the source node. As time progresses, the subsequent transmissions use different time values. TICK is also an effective dynamic en-route filtering mechanism, where the malicious data is filtered from the network. Both analytical and simulation results verified the feasibility of the TICK scheme and presented that TICK was more energy-efficient than other comparable schemes.

6.1.4 Secure SOurce-BASed Loose Time Synchronization (SOBAS) for Wireless Sensor Networks

The SOBAS protocol was presented in Chapter 5 and was developed as a derivative of the TICK protocol. In SOBAS, instead of synchronizing each sensor globally (as done in approaches providing perfect synchronization), the primary objective was to ensure that each source node was synchronized loosely with the sink and/or nodes along the data delivery path such that event reports generated by the sink were ordered properly. Further, as in TICK, the clock-based dynamically changing keys to the encryption operation prevents malicious nodes from injecting false timing packets into the network.

The feasibility of the SOBAS scheme was verified by both analytical and simulation results. SOBAS achieves the main goal of synchronizing events loosely at the sink and at the data delivery path as quick, as accurate, and as surreptitious as possible. The SOBAS protocol is able to provide 7.24μs clock precision on the data delivery path given today's sensor technology. Although it is not plausable to compare perfect synchronization and with loose synchronization directly, it should be noted that this value (7.24μs) is even better than other perfect synchronization schemes (not including ones with GPS devices), and is at least twice better in energy efficiency than other perfect synchronization schemes. Given the unique on demand or periodic event-fetching nature of WSN applications, the approach of loose time synchronization is well-suited for the characteristics of many WSN applications, where utmost silence is necessary (like in military scenarios), as SOBAS is not "chatty".

6.2 Future Research Directions

Several areas of interest exist for future work. Each is described below.

- ***Unbundling of security services for other resource-limited devices***:

As stated in Chapter 2 and [6], one of the primary motivations for this thesis work was to present the security needs of resource-limited devices, and in particular WSNs. It is not only important to know that all other resource-limited devices (e.g., PDAs, smart phones, intelligent motes, actuators, etc.) have unique energy consumption profiles, but also important to accept energy-efficient and light-secure solutions as required fundamental building blocks of the systems that are going to be designed. To achieve this, the concepts and approaches presented in this thesis can easily be adapted for the needs of other resource-limited devices. One promising direction is to consider unbundling of security services and implement them in a layered-fashion. We believe further improvements can be accomplished by unbundling some of the unnecassray security services, which may be contrary to most of the established principles.

- ***Addressing insider threats in an energy-efficient manner for resource-limited devices***: Insider threats is of great concern in all aspects of the security research. However, this issue poses even more challenging cases for already resource-limited devices. Hence, energy-efficient solutions for resource-limited devices must be investigated.

- ***Extension of ideas into Cyber-Physical Systems-oriented applications***: Cyber-Physical Systems (CPS) [5] are envisioned to be consisting of large-scale interconnected systems of heterogeneous components (e.g., sensors) interacting with their pyhsical environments. Specifically, CPS applications entail how humans and/or smart networking devices interact with and control the physical world around them. The protocols built for sensor-based CPS applications should be able to adapt to different environmental conditions, to be compatable with other CPS applications, to be re-useable and sustainable in

different CPS platforms/deployments, and most importantly to function properly even when surrounded by untrusted entities. Given the limited resources onboard the sensor devices, these specific needs require a modular, flexible, and energy-efficient sensor network communication architecture. Dynamic cryptic credential-based solutions can be tailored for the needs of CPS applications.

APPENDIX A

RC4 STREAM CIPHER

In this chapter, RC4 [30, 58] encryption scheme is explained briefly to provide a background for all the protocols introduced in this thesis. However, it should again be noted that the use of RC4 is not stipulated in any of the protocols designed in this thesis. The framework for the protocols is designed with modular architecture; therefore, it provides flexibility to adopt other encryption mechanisms as well.

RC4 is a variable key-size stream cipher developed by Ronald Rivest for RSA security. As of writing of this thesis, RC4 is one of most commonly used stream ciphers in the literature. It is effectively used in SSL/TLS, standards for securing communications between web browsers and servers, and in the IEEE 802.11 wireless LAN standard.

There are three steps involved in RC4 operations: *Initialization, initial-permutation*, and *key-stream generation*. In the initialization step, a variable key-length of 1 to 256 bytes is used to initialize a 256-byte state vector, S, or state-bytes. A key-array of size 256, K, is also created in this step. In the initial-permutation step, state-byte vector entries are permuted based on the values of K. In the final step, a byte k is generated from S, by choosing one of 256 entries. With each value of k, the entries in S are once again permuted. The algorithm for RC4 is given in Algorithm 5.

RC4 is a byte-oriented cipher and runs faster in software [30]. Hence, it would be a good choice for WSN applications where a stream of data is transferred from the sources to the sink. Furthermore, in the protocols in this thesis, in order to prevent any differantial cryptanalysis on the cipher [58], different keys with sizes of at least 128 bits are assumed to be used for each packet transmitted in the network.

Algorithm 5 RC4 Encryption – *Src:* [58]

1: $l \leftarrow Key - length$
2: // **Initial-state step**
3: **for** $i = 0$ to 255 **do**
4: $S[i] \leftarrow i$
5: $K[i] \leftarrow \text{Key}[i \bmod l]$
6: **end for**
7: $j \leftarrow 0$
8: // **Initial-permutation step**
9: **for** $i = 0$ to 255 **do**
10: $j \leftarrow (j + S[i] + K[i]) \bmod 256$
11: $swap(S[i], S[j])$
12: **end for**
13: $i \leftarrow 0$; $j \leftarrow 0$
14: // **Stream-key generation step**
15: **while** more bytes to encrypt **do**
16: $i \leftarrow (i + 1) \bmod 256$
17: $j \leftarrow (j + S[i]) \bmod 256$
18: $swap(S[i], S[j])$
19: $k \leftarrow S[(S[i] + S[j]) \bmod 256]//k = \text{one-byte key}$
20: $C \leftarrow P \oplus k$ $//C = \text{Cipher}$; $P = \text{Plaintext}$
21: **end while**

REFERENCES

[1] Z. Yu and Y. Guan, "A dynamic en-route scheme for filtering false data injection in wireless sensor networks," *Proc. of IEEE INFOCOM*, pp. 1–12, April 2006.

[2] F. Ye, H. Luo, S. Lu, and L. Zhang, "Statistical en-route filtering of injected false data in sensor networks," *IEEE JSAC*, vol. 23, no. 4, pp. 839–850, April 2005.

[3] C. Kraub, M. Schneider, K. Bayarou, and C. Eckert, "Stef: A secure ticket-based en-route filtering scheme for wireless sensor networks," *The 2nd Int. Conf. on Availability, Reliability and Security (ARES)*, pp. 310–317, April 2007.

[4] A. S. Uluagac, R. Beyah, and J. Copeland, "Virtual energy-based encryption and keying (vebek) for wireless sensor networks," *Accepted to Appear in IEEE Transactions on Mobile Computing,*, 2010.

[5] M. Iqbal and H. B. Lim, "A cyber-physical middleware framework for continuous monitoring of water distribution systems," in *Proc. of the 7th ACM SenSys*, 2009, pp. 401–402.

[6] S. Uluagac, C. Lee, R. Beyah, and J. Copeland, "Designing secure protocols for wireless sensor networks," *Lecture Notes in Computer Science, Wireless Algorithms, Systems, and Applications (WASA).*, vol. 5258, pp. 503–514, 2008.

[7] A. S. Uluagac, R. A. Beyah, and J. A. Copeland, "TIme-Based dynamic keying and en-route filtering (TICK) for wireless sensor networks," in *Submitted to IEEE Globecom 2010 - Communication & Information System Security*, Miami, Florida, USA.

[8] S. Uluagac, R. Beyah, and J. Copeland, "Secure source-Based loose time Synchronization (sobas) for wireless sensor networks," Communications Systems Center, School of ECE, Georgia Institute of Technology, Tech. Rep., 2009. [Online]. Available: http://users.ece.gatech.edu/~selcuk/sobas-csc-techreport.pdf

[9] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: A survey," *Computer Networks (Elsevier) Journal*, vol. 38, no. 4, pp. 393–422, Mar. 2002.

[10] L. Eschenauer and V. D. Gligor, "A key-management scheme for distributed sensor networks," in *Proc. of IEEE Symposium on Security and Privacy*, 2002, pp. 41–47.

[11] M. Eltoweissy, M. Moharrum, and R. Mukkamala, "Dynamic key management in sensor networks," *IEEE Communications Magazine*, vol. 44, no. 4, pp. 122–130, April 2006.

[12] Y. Xiao, V. K. Rayi, B. Sun, X. Du, F. Hu, and M. Galloway, "A survey of key management schemes in wireless sensor networks," *Comput. Commun.*, vol. 30, no. 11-12, pp. 2314–2341, 2007.

[13] H. Hou, C. Corbett, Y. Li, and R. Beyah, "Dynamic energy-based encoding and filtering in sensor networks," in *Proc. of the IEEE MILCOM*, October 2007.

[14] G. J. Pottie and W. J. Kaiser, "Wireless integrated network sensors," *Commun. ACM*, vol. 43, no. 5, pp. 51–58, 2000.

[15] R. Roman, C. Alcaraz, and J. Lopez, "A survey of cryptographic primitives and implementations for hardware-constrained sensor network nodes," *Mobile Networks and Applications, Springer*, vol. 12, no. 4, pp. 231–244, August 2007.

[16] Xbow, "Crossbow technology," 2008. [Online]. Available: http://www.xbow.com/

[17] I. F. Akyildiz, M. C. Vuran, and O. B. Akan, "A cross layer protocol for wireless sensor networks," in *Proc. CISS '06*, Princeton, NJ, March 2006.

[18] A. Perrig, J. Stankovic, and D. Wagner, "Security in wireless sensor networks," in *Communications of the ACM*, vol. 47, no. 6, June 2004, pp. 53–57.

[19] C. Karlof and D. Wagner, "Secure routing in wireless sensor networks: Attacks and countermeasures," *Elsevier's AdHoc Networks Journal, Special Issue on Sensor Network Applications and Protocols*, vol. 1, no. 2-3, pp. 293–315, September 2003.

[20] F. Hu and N. K. Sharma, "Security considerations in ad hoc sensor networks," *Elsevier's AdHoc Networks Journal*, vol. 3, no. 1, pp. 69–89, January 2005.

[21] A. D. Wood and J. A. Stankovic, "Denial of service in sensor networks," *IEEE Computer*, vol. 35, no. 10, pp. 54–62, Oct. 2002.

[22] T. Roosta, S. Shieh, and S. Sastry, "Taxonomy of security attacks in sensor networks," in *The First IEEE International Conference on System Integration and Reliability Improvements*, Hanoi, Vietnam, Dec 2006.

[23] R. D. Pietro, L. Mancini, A. Mei, A. Panconesi, and J. Radhakrishnan, "How to design connected sensor networks that are provably secure," *Securecomm*, pp. 89–100, 2006.

[24] X. Wang, S. Chellappan, W. Gu, W. Yu, and D. Xuan, "Policy-driven physical attacks in sensor networks: modeling and measurement," in *IEEE Wireless Communications and Networking Conference (WCNC 2006)*, vol. 2. IEEE, April 2006, pp. 671– 678.

[25] I. Khalil, S. Bagchi, and C. Nina-Rotaru, "Dicas: Detection, diagnosis and isolation of control attacks in sensor networks," *Securecomm*, vol. 0, pp. 89–100, 2005.

[26] J. Deng, R. Han, and S. Mishra, "Countermeasures against traffic analysis attacks in wireless sensor networks," in *Security and Privacy for Emerging Areas in Communications Networks (SecureComm)*. New York, NY, USA: IEEE, September 2005, pp. 113–124.

[27] M. Manzo, T. Roosta, and S. Sastry, "Time synchronization attacks in sensor networks," in *SASN '05: Proceedings of the 3rd ACM workshop on Security of Ad Hoc and Sensor Networks*. ACM Press, 2005, pp. 76–88.

[28] Y. W. Law, L. van Hoesel, J. Doumen, P. Hartel, and P. Havinga, "Energy-efficient link-layer jamming attacks against wireless sensor network mac protocols," in *SASN '05: Proceedings of the 3rd ACM workshop on Security of Ad Hoc and Sensor Networks*. New York, NY, USA: ACM Press, 2005, pp. 76–88.

[29] A. Gabrielli, L. Mancini, S. Setia, and S. Jajodia, "Securing topology maintenance protocols for sensor networks: Attacks and countermeasures," in *Security and Privacy for Emerging Areas in Communications Networks (SecureComm)*. New York, NY, USA: IEEE, September 2005, pp. 101–102.

[30] W. Stallings, *Cryptography and Network Security: Principles and Practices (3rd edition)*. Prentice Hall, 2003.

[31] E. Shi and A. Perrig, "Designing secure sensor networks," *IEEE Wireless Communications*, vol. 11, no. 6, pp. 38– 43, December 2004.

[32] I.-T. R. X.800, "Security architecture for open systems interconnection for ccitt applications," 1991.

[33] Y. W. Law, J. Doumen, and P. Hartel, "Survey and benchmark of block ciphers for wireless sensor networks," *ACM Trans. Sen. Netw.*, vol. 2, no. 1, pp. 65–93, 2006.

[34] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler, "Spins: security protocols for sensor networks," *Kluwer Wireles Networks*, vol. 8, no. 5, pp. 521–534, 2002.

[35] C. Karlof, N. Sastry, and D. Wagner, "Tinysec: A link layer security architecture for wireless sensor networks," in *ACM SenSys 2004*, November 2004.

[36] Y. Zhou, Y. Zhang, and Y. Fang, "Access control in wireless sensor networks," *Elsevier's AdHoc Networks Journal*, vol. 5, no. 1, pp. 3–13, January 2007.

[37] C. Vu, R. Beyah, and Y. Li, "A composite event detection in wireless sensor networks," in *Proc. of the IEEE IPCCC*, April 2007.

[38] M. Ma, "Resilience of sink filtering scheme in wireless sensor networks," *Elsevier Comput. Commun.*, vol. 30, no. 1, pp. 55–65, 2006.

[39] J. Hyun and S. Kim, "Low energy consumption security method for protecting information of wireless sensor networks," *LNCS Advanced Web and Network Technologies, and Applications*, vol. 3842, pp. 397–404, 2006.

[40] S. Zhu, S. Setia, S. Jajodia, and P. Ning, "An interleaved hop-by-hop authentication scheme for filtering of injected false data in sensor networks," in *Proc. of IEEE Symposium on Security and Privacy*, 2004.

[41] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: A scalable and robust communication paradigm for sensor networks," in *Proc. of ACM MOBICOM*, August 2002, pp. 56–67.

[42] A. Perrig, R. Szewczyk, V. Wen, D. Cullar, and J. Tygar, "Spins: Security protocols for sensor networks," in *Proc. of MOBICOM'01*, 2001.

[43] M. Luk, G. Mezzour, A. Perrig, and V. Gligor, "Minisec: A secure sensor network communication architecture," *6th International Symposium on Information Processing in Sensor Networks (IPSN)*, pp. 479–488, April 2007.

[44] M. Zorzi and R. Rao, "Geographic random forwarding (geraf) for ad hoc and sensor networks: multihop performance," *Mobile Computing, IEEE Transactions on*, vol. 2, no. 4, pp. 337–348, Oct.-Dec. 2003.

[45] M. Vuran and I. Akyildiz, "Cross-layer analysis of error control in wireless sensor networks," *Proc. of IEEE SECON*, vol. 2, pp. 585–594, Sept. 2006.

[46] K. Akkaya and M. Younis, "A survey on routing protocols for wireless sensor networks," *Elsevier Ad Hoc Networks Journal*, vol. 3, pp. 325–349, May 2005.

[47] G. T. S. N. Simulator, "Gtsnets," 2007.

[48] R. V. et al., "Encryption overhead in embedded systems and sensor network nodes: modeling and analysis," in *Proc. of ACM CASES '03*, 2003, pp. 188–197.

[49] M. Passing and F. Dressler, "Experimental performance evaluation of cryptographic algorithms on sensor nodes," Oct. 2006, pp. 882–887.

[50] D. of Defense, *Department of Defense Dictionary of Military and Associated Terms, Joint Publication 1-02*, 2001.

[51] B. Nguyen and R. Rom, "Communication services under emcon," in *Proc. of the ACM SIGCOMM Conference*, 1986, pp. 275–281.

[52] J. Jeong and Z. Haas, "Predeployed secure key distribution mechanisms in sensor networks: current state-of-the-art and a new approach using time information," *IEEE Wireless Communications*, vol. 15, no. 4, pp. 42–51, Aug. 2008.

[53] D. Scott, "Relying on time synchronization for security in ad hoc networks," in *Proc. of 43rd ACM Southeast Conference*, March 2005.

[54] A. Boukerche and D. Turgut, "Secure time synchronization protocols for wireless sensor networks," *IEEE Wireless Communications*, vol. 14, no. 5, pp. 64–69, October 2007.

[55] K. Sun, P. Ning, and C. Wang, "Tinysersync: secure and resilient time synchronization in wireless sensor networks," in *CCS '06: Proceedings of the 13th ACM conference on Computer and communications security.* New York, NY, USA: ACM, 2006, pp. 264–277.

[56] S. Ganeriwal, C. Pöpper, S. Čapkun, and M. B. Srivastava, "Secure time synchronization in sensor networks," *ACM Trans. Inf. Syst. Secur.*, vol. 11, no. 4, pp. 1–35, 2008.

[57] K. Sun, P. Ning, and C. Wang, "Secure and resilient clock synchronization in wireless sensor networks," *IEEE JSAC*, vol. 24, no. 2, pp. 395–408, Feb. 2006.

[58] B. A. Forouzan, *Cryptography & Network Security (1st edition).* McGraw-Hill, 2007.

[59] S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing-sync protocol for sensor networks," in *Proc. of the ACM SenSys*, 2003, pp. 138–149.

[60] O. Akan and I. Akyildiz, "Event-to-Sink Reliable Transport in Wireless Sensor Networks," *IEEE/ACM Transactions on Networking*, vol. 13, no. 5, pp. 1003–1017, Oct. 2005.

[61] *MSP430x1xx Family User's Guide Rev. F*, Texas Instruments, November 2008. [Online]. Available: http://www.ti.com/msp430

[62] *CC2420DataSheet, 2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver Rev. B*, Chipcon Products from Texas Instruments, November 2008. [Online]. Available: focus.ti.com/lit/ds/symlink/cc2420.pdf

[63] B. Sundararaman, U. Buy, and A. D. Kshemkalyani, "Clock synchronization for wireless sensor networks: a survey," *Elsevier's AdHoc Networks Journal*, vol. 3, no. 3, pp. 281–323, May 2005.

[64] W. Su and I. F. Akyildiz, "Time-diffusion synchronization protocol for wireless sensor networks," *IEEE/ACM Trans. Netw.*, vol. 13, no. 2, pp. 384–397, 2005.

[65] S. Ganeriwal, S. Čapkun, C.-C. Han, and M. B. Srivastava, "Secure time synchronization service for sensor networks," in *Proc. of the ACM workshop on Wireless security (WiSe)*, 2005, pp. 97–106.

[66] H. Song, S. Zhu, and G. Cao, "Attack-resilient time synchronization for wireless sensor networks," *Ad Hoc Networks*, vol. 5, pp. 112–125, 2005.

[67] F. Stann and J. Heidemann, "RMST: Reliable data transport in sensor networks," in *Proceedings of IEEE SNPA 2003*, Anchorage, Alaska, USA, May 2003, pp. 102–112.

[68] Q. Li and D. Rus, "Global clock synchronization in sensor networks," *IEEE Transactions on Computers*, vol. 55, no. 2, pp. 214–226, Feb. 2006.

[69] C. Riechmann, "An x.400 message transfer protocol for emcon network situations," in *Proc. of the IEEE MILCOM*, vol. 1, Nov 1997, pp. 444–449 vol.1.

[70] R. Roman, C. Alcaraza, and J. Lopeza, "The role of wireless sensor networks in the area of critical information infrastructure protection," *Information Security Technical Report, Elsevier*, vol. 11, no. 1, pp. 24–31, Aug. 2007.

[71] H. Oztarak, A. Yazici, D. Aksoy, and R. George, "Multimedia processing in wireless sensor networks," in *Proc. of the Innovations in Information Technology Conference*, Nov. 2007, pp. 78–82.

[72] O. Chipara, C. Brooks, S. Bhattacharya, C. Lu, R. Chamberlain, G.-C. Roman, and T. Bailey, "Reliable real-time clinical monitoring using sensor network technology," in *American Medical Informatics Association (AMIA) Annual Symposium*, Nov. 2009.

[73] F. Sivrikaya and B. Yener, "Time synchronization in sensor networks: a survey," *IEEE Network*, vol. 18, no. 4, pp. 45–50, July-Aug. 2004.

[74] S. Mizanur Rahman, N. Nasser, and T. Taleb, "Secure timing synchronization for heterogeneous sensor network using pairing over elliptic curve," *Wireless Communications and Mobile Computing*, 2009.

[75] K. B. Rasmussen, S. Capkun, and M. Cagalj, "Secnav: secure broadcast localization and time synchronization in wireless networks," in *Proc. of the ACM MobiCom*, 2007, pp. 310–313.

[76] F. Ren, C. Lin, and F. Liu, "Self-correcting time synchronization using reference broadcast in wireless sensor network," *IEEE Wireless Communications*, vol. 15, no. 4, pp. 79–85, Aug. 2008.

[77] P. Sommer and R. Wattenhofer, "Gradient clock synchronization in wireless sensor networks," in *Proc. of the International Conference on Information Processing in Sensor Networks (IPSN)*, April 2009, pp. 37–48.

[78] S. Yoon, C. Veerarittiphan, and M. L. Sichitiu, "Tiny-sync: Tight time synchronization for wireless sensor networks," *ACM Trans. Sen. Netw.*, vol. 3, no. 2, p. 8, 2007.

[79] Y. Zeng, B. Hu, and S. Liu, "Vector Kalman Filter Using Multiple Parents for Time Synchronization in Multi-Hop Sensor Networks," in *Proc. of the IEEE SECON*, 2008, pp. 413–421.

[80] B. A. Forouzan, *Data Communications and Networking (4th edition).* McGraw-Hill, 2007.

[81] H. Yoshida and T. Mori, "Development of precision underwater positioning system," in *Proc. of the Underwater Technology and Workshop on Scientific Use of Submarine Cables and Related Technologies Symposium*, April 2007, pp. 217–222.

[82] J. W. Youngberg, *Method for extending GPS to underwater applications.* United States Patent 5119341, 1992.

VITA

Arif Selçuk Uluağaç was born on June 8, 1975 in Izmir, Turkey. He received his B.Sc. in Computer Engineering from the Turkish Naval Academy, Istanbul, Turkey, and M.Sc. degrees in Electrical and Computer Engineering (ECE) from Carnegie Mellon University, Pittsburgh, PA, USA, in 1997 and 2002, respectively. He also holds an M.Sc. in Information Security from the School of Computer Science at Georgia Institute of Technology, Atlanta, GA, USA.

He received his Doctor of Philosophy (Ph.D.) degree in ECE from Georgia Tech on June 8, 2010. During his PhD, he was a graduate research assistant (GRA) in the Communications Systems Center Laboratory at Georgia Tech and has several publications in the areas of security and networks.

Over the last 2 years of his PhD, he taught courses related to computer communications, computer security, and network security in the department of Electrical and Computer Engineering Technology (ECET) at Southern Polytechnic State University as an adjunct instructor while he was a Ph.D. student at Georgia Tech. Furthermore, in an earlier teaching experience as a graduate teaching assistant (GTA) at Georgia Tech, he received the 2007 Outstanding ECE Graduate Teaching Assistant Award from the School of ECE. He is a member of IEEE, ACM, and ASEE.

PUBLICATIONS

** A. S. Uluagac, R. A. Beyah, Y.Li, and J. A. Copeland, "Virtual Energy-Based Encryption and Keying (VEBEK) for Wireless Sensor Networks," ***IEEE Transactions on Mobile Computing***, Vol. 9, No. 7, pp. 994-1007, 2010

* C. P. Lee, A. S. Uluagac, K.Fairbanks, and J. A. Copeland, The Design of NetSecLab: A Small Competition-Based Network Security Lab, Accepted to appear in the ***IEEE Transactions on Education***.

** A. S. Uluagac, C. P. Lee, R. A. Beyah, and J. A. Copeland, "Designing Secure Protocols for Wireless Sensor Networks," in Wireless Algorithms, Systems, and Applications, *Lecture Notes in Computer Science 5258*, Yingshu Li et al. (Eds.), Springer-Verlag, pp. 503-514, 2008.

* A. S. Uluagac and J. M. Peha, "IP Multicast over Cable TV Networks," in Group Communications and Charges, *Lecture Notes in Computer Science 2816*, B. Stiller et al. (Eds.), Springer-Verlag, pp. 168-180, 2003.

* A.Selcuk Uluagac, Roma Kane, Siddharth Joshi, R. A. Beyah, and J. A. Copeland, "Analysis of Varying AS Path Lengths from the Edge of the Network," Proceedings of the *IEEE International Communications Conference (ICC)*, Cape Town, South Africa, May 2010

* A. S. Uluagac, T. Fallon, W. Thain, and J. A. Copeland, "Development of Undergraduate Network Security Labs with Open Source Tools," Proceedings of the *American Society for Engineering Education (ASEE)*, Annual Conference of Composition and Exhibition, Austin, Texas, June 2009.

* A. S. Uluagac and D. Williams, Building Hardware-based Low-Cost Experimental DSP Learning Modules, Proceedings of the *American Society for Engineering Education (ASEE)*, Annual Conference of Composition and Exhibition, Pittsburgh, PA, June 2008.

** A. S. Uluagac, R. A. Beyah, and J. A. Copeland, *TI*me-Based Dynami*C K*eying and En-Route Filtering (TICK) for Wireless Sensor Networks, Submitted to *IEEE GLOBECOM-2010* (under review).

** A. S. Uluagac, R. A. Beyah, and J. A. Copeland, *S*ecure S*O*urce-*BA*sed Loose Time *S*ynchronization (SOBAS) for Wireless Sensor Networks, in preparation for the ***IEEE Transactions on Parallel and Distributed Systems (TPDS)***.

INDEX

A

Access Control, 18
An Analysis of the Rekeying Cost for WSNs, 30
Authentication, 16
availability, 20

B

Bridging concept, 44

C

Communication Model for WSNs, 10
Confidentiality, 12
Cyber-physical systems (CPS), 1

D

Dynamic cryptic credential, 2

F

VEBEK Forwarding Module, 41

I

Integrity, 17

N

Nonrepudiation, 19

R

RC4, 113

S

Semantics of VEBEK, 33
SOBAS, 82
Spatio-Temporal Security, 67

T

Target-Based Threat Model for WSNs, 11
Threat Model for WSNs, 10
TICK, 63
TICK CRYPT Module, 69
TICK FFWD Module, 71
TICK TKM Module, 68
Time Uncertainty, 72

U

Uluağaç, Arif Selçuk , 122

V

VEBEK, 25
VEBEK Crypto Module, 38
VEBEK Module, 34
VEBEK-I, 46
VEBEK-II, 48

W

Watching concept, 36
Watching list, 36
When to employ which service, 21