

EFFICIENT AND SECURE KEY DISTRIBUTION  
PROTOCOL FOR WIRELESS SENSOR NETWORKS

Majid Alshammari

Under the Supervision of Prof. Khaled Elleithy

DISSERTATION

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE AND

ENGINEERING

THE SCHOOL OF ENGINEERING

UNIVERSITY OF BRIDGEPORT

CONNECTICUT

DECEMBER, 2018

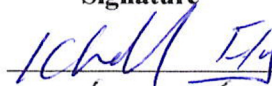
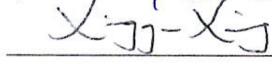

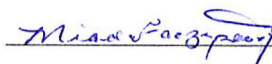

# EFFICIENT AND SECURE KEY DISTRIBUTION PROTOCOL FOR WIRELESS SENSOR NETWORKS

Majid Alshammari

Under the Supervision of Prof. Khaled Elleithy

## Approvals

### Committee Members

Name	Signature	Date
Prof. Khaled Elleithy		12/3/18
Prof. Xingguo Xiong		12/03/18
Prof. Junling Hu		12/3/18
Prof. Miad Faezipour		12, 3, 2018
Prof. Eman Abdel Fattah		12/3/18

### Ph. D. Program Coordinator

Prof. Khaled Elleithy		12/3/18
-----------------------	--	---------

### Chairman, Computer Science and Engineering Department

Prof. Ausif Mahmood		12-5-2018
---------------------	--	-----------

### Dean, School of Engineering

Prof. Tarek M. Sobh		12/5/2018
---------------------	--	-----------

EFFICIENT AND SECURE KEY DISTRIBUTION  
PROTOCOL FOR WIRELESS SENSOR NETWORKS

© Copyright by Majid Alshammari 2018

# EFFICIENT AND SECURE KEY DISTRIBUTION PROTOCOL FOR WIRELESS SENSOR NETWORKS

## ABSTRACT

Modern wireless sensor networks have adopted the IEEE 802.15.4 standard. This standard defines the first two layers, the physical and medium access control layers; determines the radio wave used for communication; and defines the 128-bit advanced encryption standard (AES-128) for encrypting and validating transmitted data. However, the standard does not specify how to manage, store, or distribute encryption keys. Many solutions have been proposed to address this problem, but the majority are impractical in resource-constrained devices such as wireless sensor nodes or cause degradation of other metrics. Therefore, we propose an efficient and secure key distribution protocol that is simple, practical, and feasible to implement on resource-constrained wireless sensor nodes. We conduct simulations and hardware implementations to analyze our work and compare it to existing solutions based on different metrics, such as energy consumption, storage overhead, key connectivity, replay attack, man-in-the-middle attack, and resiliency to node capture attack. Our findings show that the proposed protocol is secure and more efficient than other solutions.

## **DEDICATION**

*In memory of my father.*

*To my lovely mother, brothers and sisters, my wife, and my amazing son, for your  
love and support.*

## **ACKNOWLEDGEMENTS**

I would like to express my great appreciation to my advisor Prof. Khaled Elleithy for his continuous support of my Ph.D. study and research. I am very grateful for his guidance and valuable suggestions during this dissertation.

I would also like to thank committee members Prof. Xingguo Xiong, Prof. Junling Hu, and Prof. Miad Faezipour, as well as the external examiner Prof. Eman Abdel Fattah for their time and valuable comments.

Last but not the least, I would like to thank all of my family members for their love, support, and encouragement.

# TABLE OF CONTENTS

ABSTRACT . . . . .	iii
DEDICATION . . . . .	iv
ACKNOWLEDGEMENTS . . . . .	v
TABLE OF CONTENTS . . . . .	vi
LIST OF TABLES . . . . .	viii
LIST OF FIGURES . . . . .	ix
CHAPTER 1: INTRODUCTION . . . . .	1
1.1 Research Problem And Scope . . . . .	1
1.2 Motivation Behind The Research. . . . .	2
1.3 Contributions Of The Proposed Research. . . . .	3
CHAPTER 2: RELATED WORK. . . . .	5
CHAPTER 3: PROPOSED PROTOCOL . . . . .	13
3.1 Pre-Deployment Phase . . . . .	14
3.2 Key Distribution Phase . . . . .	15
3.3 Post-Key Distribution Phase . . . . .	17
3.4 Key Refreshment Phase . . . . .	19
CHAPTER 4: METHODOLOGY. . . . .	21
4.1 Experiment Design And Parameters. . . . .	21
4.2 Energy Consumption Of Wireless Sensor Node . . . . .	23
4.2.1 Energy Consumption Of The Transceiver . . . . .	24
4.2.2 Energy Consumption Of The Microcontroller . . . . .	26
4.3 Modeling Wireless Channel Effects . . . . .	26
4.4 Fast Modular Exponentiation Algorithm . . . . .	28
CHAPTER 5: FINDINGS AND ANALYSES . . . . .	29
5.1 Efficiency Analysis . . . . .	29
5.1.1 Energy Consumption . . . . .	29
5.1.2 Key Storage Overhead . . . . .	32
5.1.3 Key Connectivity. . . . .	34

5.2 Security Analysis . . . . .	38
5.2.1 Replay Attack . . . . .	38
5.2.2 Man-In-The-Middle Attack . . . . .	39
5.2.3 Node Capture Attack . . . . .	40
CHAPTER 6: FORMAL VERIFICATION . . . . .	45
6.1 Reachability And Secrecy . . . . .	45
6.2 Correspondence Assertions . . . . .	48
6.3 Observational Equivalence . . . . .	51
CHAPTER 7: CONCLUSION . . . . .	54
REFERENCES . . . . .	55



## LIST OF TABLES

Table 2.1 Evaluation Metrics. . . . .	7
Table 3.1 Notation for the Proposed Protocol. . . . .	13
Table 4.1 Experiment Parameters.. . . . .	22
Table 5.1 Energy consumption of each key distribution scheme . . . . .	31
Table 5.2 Key Storage overhead for each scheme. . . . .	32

## LIST OF FIGURES

Figure 2.1 Classification of key distribution schemes in WSNs. . . . .	6
Figure 3.1 Pre-deployment phase steps. . . . .	15
Figure 3.2 Key distribution phase steps. . . . .	16
Figure 3.3 Post-key distribution phase steps. . . . .	18
Figure 3.4 Key refreshment phase steps. . . . .	20
Figure 4.1 Our wireless sensor node model. . . . .	22
Figure 5.1 The magnitude of key storage overhead . . . . .	33
Figure 5.2 Modeling a WSN: <b>(a)</b> random deployment of sensor nodes; <b>(b)</b> wireless signal range of nodes' transceivers. . . . .	35
Figure 5.3 Key connectivity after implementing key distribution/establishment process for each scheme. <b>(a)</b> the proposed protocol; <b>(b)</b> scheme[35]; <b>(c)</b> scheme[44]; <b>(d)</b> scheme[45] ; <b>(e)</b> scheme[55] . . . . .	38
Figure 5.4 Schemes' resilience against node capture attacks. <b>(a)</b> The proposed protocol; <b>(b)</b> Scheme[35]; <b>(c)</b> Scheme[44]; <b>(d)</b> Scheme[45] ; <b>(e)</b> Scheme[55]. . . . .	44
Figure 6.1 Verification result of reachability and secrecy. . . . .	48
Figure 6.2 Verification result of authentication. . . . .	50

Figure 6.3 Verification result of observational equivalence. . . . . 53

# CHAPTER 1: INTRODUCTION

A wireless sensor network (WSN) is a network composed of resource-constrained devices with the ability to perform sensing and wireless communications, which are called wireless sensor nodes. WSN nodes link a spatial space or an object to a computing system for the purpose of monitoring, controlling, or targeting. The concept of WSNs was developed by the U.S. military [1–3]. Then, academic institutions began to improve upon this technology. These improvements led to advances in hardware and communication; for example, wireless sensor nodes became smaller in size and more cost-effective. Today, WSNs enable cyber-physical system (CPS) applications [4–10], and they have become a core technology in the Internet of things (IoT) [11–14]. WSNs have become rapidly involved in a variety of modern purposes, with applications in agriculture, the environment, health, home and commercial automation, the military and transportation [15–25].

## 1.1 Research Problem And Scope

Modern WSNs adopt the IEEE 802.15.4 standard, which specifies the physical layer and the medium access control (MAC) layer for low-rate wireless personal area networks (LR-WPANs). The standard also determines the radio frequency used for communication and provides four security services: access control, confidentiality, in-

egrity and replay protection. The MAC layer handles security for the IEEE 802.15.4 standard and defines the 128-bit advanced encryption standard (AES-128) for encrypting and validating transmitted data. Unfortunately, the standard does not specify how to manage, store, or distribute encryption keys [26]. Many solutions have been proposed to address this problem, but the majority are impractical in resource-constrained devices such as wireless sensor nodes or cause degradation of other metrics.

## **1.2 Motivation Behind The Research**

In the literature, many solutions have been proposed to address the key distribution problem, but the majority are impractical in resource-constrained devices such as wireless sensor nodes or cause degradation of other metrics. For example, some of these schemes applied asymmetric encryptions without a proper adjustment for resource-constrained devices. This type of schemes may be secure against some attacks, but it is inefficient in terms of energy and memory consumption. To circumvent this, some other schemes implemented symmetric encryption, but they too relied on energy and memory consuming techniques such as: storing several keys in each sensor node, engaging intermediary nodes, or exchanging numerous frames to find a common key between sensor nodes. Additional schemes adapted quantum cryptography for the key distribution, although quantum cryptography is not yet practical in resource-constrained devices.

Additionally, most of these schemes did not consider the energy consumed by the nodes transceivers, the energy consumed by the transmitters power output (TPO), and the energy consumption caused by wireless channel effects.

### 1.3 Contributions Of The Proposed Research

In this work, we propose an efficient and secure key distribution protocol for WSNs. We utilized the existing cryptographic primitives to design a protocol that is simple, practical and feasible to implement on resource-constrained devices such as wireless sensor nodes. The contributions of our work can be summarized as follows.

- We introduce a comprehensive classification for the main key distribution and key establishment schemes in WSNs. We classify the schemes into traditional key distribution schemes, including private-key-based schemes and public-key-based schemes, and quantum-based key distribution schemes, including those based on entanglement swapping and teleportation.
- We propose an efficient and secure key distribution protocol that is simple, practical and feasible to implement on resource-constrained devices such as wireless sensor nodes. Because data communication is responsible for most of a node's energy consumption [27], the proposed protocol utilizes the existing cryptographic primitives and leverages asymmetric encryption to achieve key distribution and node authentication in one step and using only one frame to avoid communication overhead. Moreover, the implementation of the proposed protocol adopts the following techniques: a fast modular exponentiation algorithm (described in Chapter 4, Section 4.4) and a short public exponent. These techniques speed up the node's data computation, resulting in lower energy consumption.
- We analyze and compare the proposed protocol against different types of schemes using various metrics, including energy consumption, key connectivity, storage

overhead, man-in-the-middle attack, replay attack and resiliency to node capture attack. Our methodology (described in Chapter 4) combines simulations, hardware implementations and practical models to calculate both the energy consumption of sensor nodes and the energy consumption caused by wireless channel effects.

- We visualize and analyze the key connectivity and the impact of node capture attack using a graph. We model a WSN as a graph and then implement the proposed protocol and the corresponding schemes on the graph to investigate their key connectivity and the impact of node capture attack on the key connectivity.
- We conduct a formal verification using an automatic cryptographic protocol verifier, ProVerif. We utilize ProVerif to prove the security and soundness of the proposed protocol in formal models. We verify the reachability and secrecy, correspondence assertions (authentication) and observational equivalences.

## CHAPTER 2: RELATED WORK

Key distribution schemes in WSNs have been comprehensively studied in the literature. The authors of [28–31] provided detailed surveys. However, in this study, we present a comprehensive overview of the existing key distribution and key establishment schemes in WSNs, which we classify into two domains. The first domain includes traditional key-based distribution schemes, which can be further classified into private-key-based and public-key-based schemes. Private-key-based schemes can be subcategorized into grid-based, polynomial-based, probabilistic, and exclusion basis system (EBS)-based schemes. Public-key-based key distribution schemes can be subcategorized based on an integer factorization problem (IFP) or on a discrete logarithm problem (DLP). The second domain includes quantum-based key distribution schemes, which can be further classified into entanglement-swapping-based and teleportation-based schemes. Figure 2.1 depicts this classification hierarchy, and Table 2.1 defines our evaluation metrics.



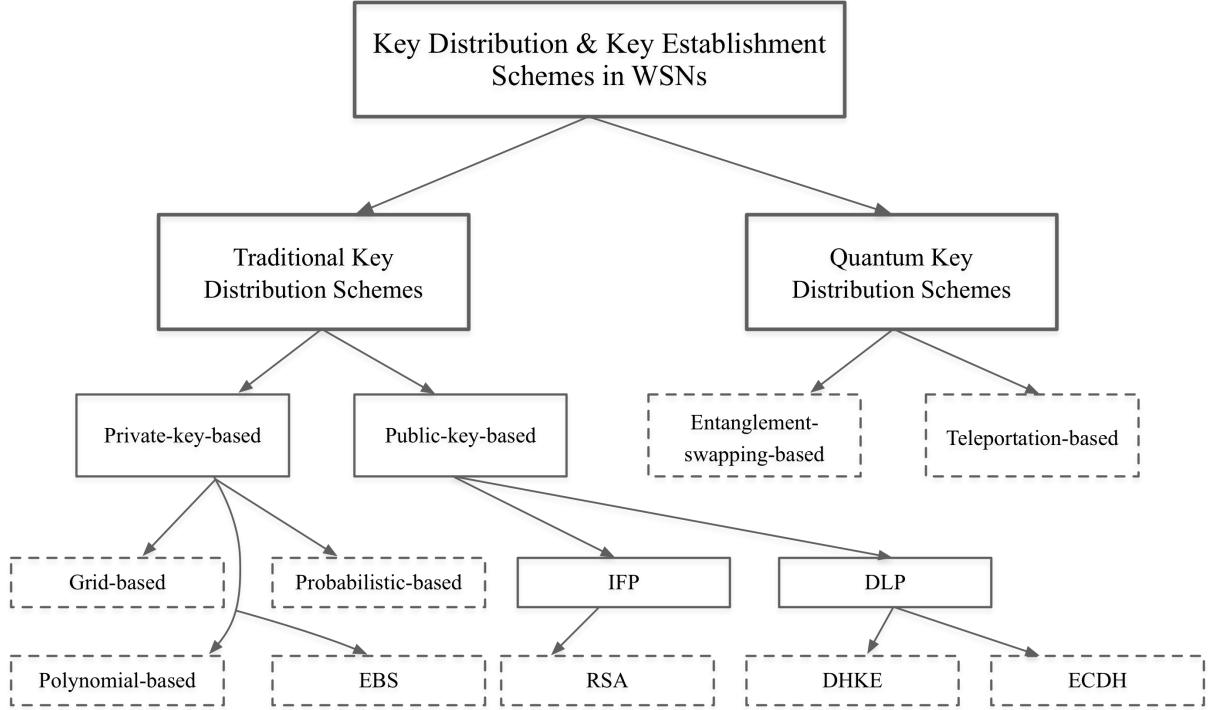


Figure 2.1: Classification of key distribution schemes in WSNs.

Grid-based schemes address a WSN of size  $n$  as an  $\sqrt{n} \cdot \sqrt{n}$  matrix or grid. In this subcategory, each node in the WSN is assigned to a unique intersection  $(i, j)$  in the grid [32–34]. An early example was called Peer Intermediaries for Key Establishment in Sensor Networks (PIKE) [35]. PIKE represents a sensor network of size  $n$  by an  $\sqrt{n} \cdot \sqrt{n}$  matrix and uses some sensor nodes as trusted intermediaries for key distribution. Each sensor has an  $ID$  in the form of  $(x, y)$  based on its position in the matrix. Moreover, each node is loaded with a pairwise secret key shared only with each node in the two sets:

$$(i, y) \forall i \in \{1, 2, 3, \dots, \sqrt{n} - 1\} \quad (2.1)$$

$$(x, j) \forall j \in \{1, 2, 3, \dots, \sqrt{n} - 1\}. \quad (2.2)$$

Table 2.1: Evaluation Metrics.

	<b>Metric</b>	<b>Definition</b>
Efficiency	Energy consumption	The amount of energy consumed during the key distribution/key establishment process.
	Storage overhead	The memory required to store keys or keys materials.
	Key connectivity	The percentage of available links in a WSN, calculated as the number of secured links divided by the total links.
Security	Replay attack	The ability of an adversary to replay any of the corresponding frames.
	Man-in-the-middle attack	The ability of an adversary to impersonate any sensor node or sink node.
	Resiliency to node capture attack	The impact percentage of a node capture attack on WSN key connectivity, calculated as the number of compromised links over the number of secured links.

Keys are deployed such that in any pair  $A$  and  $B$ , at least one node  $C$  exists that shares a pairwise key with both  $A$  and  $B$ . However, this approach suffers from key dependency because one inoperable or missing node would impact the network connectivity. Additionally, the search process for intermediary nodes consumes a large amount of energy because it involves sending many frames to other sensor nodes searching for a node that shares a pairwise key. Moreover, this approach requires each sensor node to store  $2(\sqrt{n-1})$  keys.

Polynomial-based schemes depend on storing polynomials on wireless sensor

nodes that are used for key generation. In [36], this process was described as a threshold scheme  $(R, K)$ . The threshold is a shared security scheme that divides a message into  $K$  parts, where  $R$  is the minimum number of parts required to reconstruct the original message. The author of [37], proposed a sharing security scheme that used a polynomial equation in a finite field to construct a threshold scheme. In this scheme, an arbitrary polynomial of degree  $R - 1$  was generated in the following form:

$$(ax^{R-1} + bx^{(R-1)-1} + \dots + m) \text{ mod } p, \quad (2.3)$$

where  $p$  is a public prime number that is greater than the coefficients,  $m$  is the message, and  $a$  and  $b$  are randomly chosen coefficients. Many other forms of polynomial-based key distribution schemes have been proposed [38–43]. In [44], the authors proposed a polynomial-based key management scheme consisting of three phases. In the first phase, sensor nodes discover the neighboring nodes and elect a cluster head (CH). In the second phase, when a sensor node wants to acquire a secure communication channel through the base station (BS), it sends a registration request. The third phase generates a triple key. Subsequently, when a sensor node wants to communicate with the CH and BS, the key is calculated based on a given polynomial and coefficients in a finite group. In this approach, sensor nodes consume large amounts of energy during the discovery and cluster-electing phases. Additionally, key connectivity is affected by node capture attacks.

Probabilistic-based schemes rely on the probability of two sensor nodes sharing a common key to establish a communication link. In [45], the authors proposed a probabilistic key-based management protocol. This scheme consists of three phases:

key predistribution, shared-key discovery and path-key establishment. During the key predistribution phase, a large pool of keys  $p$  and their identifiers are generated. Then, a random set of keys  $k$  and their identifiers are drawn out of the key pool  $p$  to form a key ring for each sensor node based on the following formula:

$$P_{key} = 1 - \left( \frac{((P - k)!)^2}{((P - 2k)!P!)} \right), \quad (2.4)$$

where  $P_{key}$  is the probability of two nodes sharing a common key. Next, the key rings are loaded into each of the sensor node memories, and the key identifiers are saved on a trusted controller node along with the associated sensor identifiers. In the shared-key discovery phase, two sensor nodes can discover a shared key by broadcasting a list of their key rings. Additionally, the two sensor nodes can hide the key sharing patterns by broadcasting a list,  $li = \{\alpha || Eki(\alpha) || i = 1, \dots, k\}$ , for every key on the key ring, where  $\alpha$  is a challenge. The ability of the receiver to decrypt  $Eki(\alpha)$  will reveal the challenge  $\alpha$  and then establish a shared key with the sender. In the path-key establishment phase, a path key is assigned to each pair of sensor nodes that do not share a key but are connected to other sensor nodes at the end of the shared-key discovery phase. However, this approach consumes large amounts of energy because finding a common key between two sensor nodes requires broadcasting many frames. Additionally, storing the key ring requires large amounts of memory, especially when the probability of sensor nodes sharing a common key is certain.

EBS-based schemes are a combinatorial formulation of key management as proposed by [46]. EBS is denoted by  $EBS(n, k, m)$  (where  $n$ ,  $k$ , and  $m$  are positive integers), and it is a collection  $\Gamma$  of subset of  $[1, n]$  such that for every integers  $t \in [1, n]$ , the fol-

lowing two properties hold:

1.  $t$  is in most  $k$  subsets in  $\Gamma$ .
2. there is exactly  $m$  subset in  $\Gamma$  such that  $A_i, \dots, A_m$  that  $t$  is excluded by a union of  $m$  subsets in  $\Gamma$ ,  $\bigcup_{i=1}^m A_i$  is  $[1, n] - t$ .

Many schemes based on EBS have been proposed [47–53]. For example, the authors of [54] proposed a key generation scheme based on a system of linear equations that utilizes EBS for key management. This scheme generates associated keys and then combines them with EBS for key management. The key management consists of three phases: pre-deployment, shared-key discovery/path-key establishment, and key redistribution. In the pre-deployment phase, a system of linear equations with two variables is used to generate the unique solution/secret keys  $T(x_s, y_s)$ :

$$E^{(2)} = \begin{cases} E_1(x, y) : a_{1,1}x + a_{1,2}y + b_1 = 0 \\ \vdots \\ E_n(x, y) : a_{n,1}x + a_{n,2}y + b_n = 0. \end{cases} \quad (2.5)$$

To figure out the unique solution, a sensor node needs two equations. However, since any equation/line of the system can be calculated using two different points on the line, these two points can be used instead of the equations to form one key. Therefore, each sensor node requires at least two keys, called the node's key rings. Corresponding IDs also are established and distributed with these keys along to the nodes. In the shared-key discovery/path-key establishment phase, nodes find the common keys by broadcasting the keys' IDs. In the key redistribution phase, the exhausted nodes are replaced without changing the keys. Extending the network requires the new nodes to choose one of

the key rings or add a new line (equation) that will pass through the secret keys unique solution/secret keys  $T(x_s, y_s)$ . However, this approach is vulnerable to node capture attacks because compromising one node will reveal the secret keys.

Public-key-based schemes are another class of the traditional key distribution schemes in WSNs. Practical public-key-based schemes depend on two major families of problems: IFPs such as the Rivest–Shamir–Adleman (RSA) cryptosystem and DLPs such as Diffie-Hellman key exchange (DHKE) and elliptic curve Diffie-Hellman (ECDH). The authors of [55] discussed using public infrastructure such as RSA to improve the security of WSNs. The study considered the WSN topology as a set of wirelessly connected sensor nodes that report collected data to the base station. However, wireless sensor nodes are resource-constrained devices, and a straightforward implementation of IFP or DLP without appropriate modifications is energy and memory intensive. In [56], the authors proposed a public-key-based key distribution scheme using ECDH. The scheme consists of two phases: a predeployment phase and a postdeployment phase. In the predeployment phase, sensor nodes are configured with elliptic curve (EC) parameters and the basepoint  $G$ . Then,  $\alpha_n$  is generated to calculate  $P_n = \alpha_n G$  for all  $n$  nodes. Next,  $\alpha_n$  is stored in each corresponding node, and all  $P_n$  are stored in the sink node. In the postdeployment phase, the sink creates a new secret key  $b$  and calculates its public key  $Q = bG$ . The public key is then broadcast to all sensor nodes. Each sensor node calculates a new key  $k_n = \alpha_n Q$ , whereas the sink calculates a new key  $k_n = bP_n$ . The downside of this approach is that each node must store all the EC parameters such as the field over which the curve is defined, the  $\alpha$  and  $b$  values that defined the curve and the generator point  $G$ .

Quantum-based schemes rely on the laws of physics and require special hard-

ware. Nevertheless, many authors have proposed solutions. For example, entanglement swapping was adopted in [57]. This scheme utilizes a third party, called a base station, to perform entanglement swapping among sensor nodes. Each sensor node shares  $n$  qubits with the base station, and the base station shares  $m$  qubits with each sensor node. When two sensor nodes  $x$  and  $y$  are separately entangled with the base station, the base station performs entanglement swapping, allowing sensor node  $x$  and sensor node  $y$  to become entangled. As another example, the authors of [58] proposed a scheme that includes Einstein-Podolsky-Rosen (EPR)-pair distribution and quantum authentication. This scheme allows a sensor node to teleport quantum information to any other sensor node in the network. The entanglement-swapping-based and teleportation-based schemes rely on quantum cryptography, which has been proven in prior literature to be secure unless the laws of physics have been defeated. However, these schemes require entangled qubits to function, and applying entangled qubits in resource-constrained devices such as WSNs is not yet practical with existing technology.

## CHAPTER 3: PROPOSED PROTOCOL

The proposed protocol includes four phases: *a pre-deployment phase, a key distribution phase, a post-key distribution phase, and a key refreshment phase.* Table 3.1 presents the notations used to describe the proposed protocol.

*Table 3.1: Notation for the Proposed Protocol.*

Notation	Description
$y \leftarrow x$	$y$ is generated by $x$ .
$x \stackrel{\text{def}}{=} y$	$x$ is defined as $y$ .
$x := y$	$y$ is assigned to $x$ .
$H()$	One-way hash function.
$\parallel$	Concatenation.
$\xrightarrow{\text{send}} [x]$	Sending message $x$ .
$\xleftarrow{\text{recv}} [x]$	Receiving message $x$ .
$E_k(y)$	$y$ is encrypted with $k$ .
$E_k^\perp(y)$	$y$ is encrypted with $k$ using algorithm $\perp$ .
$f()$	Function to compare or verify.
$P()$	Probability function.
$F : A \mapsto B$	Function maps $A$ to $B$ .
$P$	Plaintext.



$C$	Cipher text.
$Id$	Node identification.
$T$	Timestamp.
$D$	Data.

---

### 3.1 Pre-Deployment Phase

The pre-deployment phase of the protocol consists of five offline steps. The first step is the generation of an asymmetric key pair  $\{K_P, K_R\} \leftarrow RSA_{gen}$ , where  $RSA_{gen}$  is the RSA key generation algorithm. In the second step,  $K_P$  is defined as a sink node key,  $AK_{sink}$ , and  $K_R$  is defined as the key for the sensor nodes,  $AK_{nodes}$ . In the third step,  $AK_{sink}$  is loaded into the sink node, and  $AK_{nodes}$  is loaded into the sensor nodes. The fourth step involves the generation of a random local key for each sensor node as follows:  $K_{local_i} \xleftarrow{R} key_{gen}\{0, 1\}^{128}$ , where  $key_{gen}\{0, 1\}^{128}$  is a random key generation algorithm with a key space of  $\{0, 1\}^{128}$ ; the key space is a uniform distribution such that  $\forall K_{local_i} \in \{0, 1\}^{128}$ , and the probability of each key is  $P(K_{local_i}) = \frac{1}{|\{0, 1\}^{128}|}$ . In the fifth step,  $K_{local_i}$  is loaded into the corresponding sensor node<sub>*i*</sub> and into the sink nodes. Figure 3.1 shows pre-deployment phase steps.

### Pre-deployment Phase Steps

$$\{K_P, K_R\} \leftarrow RSA_{gen}$$

$$K_P \stackrel{\text{def}}{=} AK_{sink} \text{ and } K_R \stackrel{\text{def}}{=} AK_{nodes}$$

$$Sink \text{ node} := AK_{sink} \text{ and } Sensor \text{ nodes} := AK_{nodes}$$

$$K_{local_i} \stackrel{R}{\leftarrow} key_{gen}\{0, 1\}^{128}, \forall K_{local_i} \in \{0, 1\}^{128} \Rightarrow P(K_{local_i}) = \frac{1}{|\{0, 1\}^{128}|}$$

$$Sensor \text{ node}_i := K_{local_i} \text{ and } Sink \text{ node} := K_{local_i}$$

Figure 3.1: Pre-deployment phase steps.

## 3.2 Key Distribution Phase

After deploying the sensor nodes, the sink node generates a random complementary key  $K_{compl} \leftarrow Ran_{gen}\{0, 1\}^{128}$ , computes its hash value  $Tag \leftarrow H(K_{compl})$ , and calculates a timestamp  $T$ . The sink node then encrypts these values using its asymmetric key  $AK_{sink}$ . After this task is complete, the sink node sends the cipher to neighboring sensor nodes as follows:

$$\xrightarrow{\text{send}} \left[ C \leftarrow E_{AK_{sink}} \left( K_{compl} || Tag \leftarrow H(K_{compl}) || T \right) \right].$$

These neighbors forward the cipher to their neighbors in a multihop fashion until all of the sensor nodes have

$$\text{received the cipher } \xleftarrow{\text{recv}} \left[ C \leftarrow E_{AK_{sink}} \left( K_{compl} || Tag \leftarrow H(K_{compl}) || T \right) \right].$$

Because each sensor node is loaded with the asymmetric key  $AK_{nodes}$  in

the pre-deployment phase, a sensor node<sub>i</sub> can decrypt the cipher as follows:

$$P \leftarrow D_{AK_{nodes}} \left( C \leftarrow E_{AK_{sink}} \left( K_{compl} || Tag \leftarrow H(K_{compl}) || T \right) \right)$$

and verifies the timestamp  $f_{verify}(T)$  based on a predefined threshold. If the timestamp

exceeds the threshold, the sensor node<sub>i</sub> rejects the cipher. Otherwise, the sensor node<sub>i</sub>

hashes the complementary key  $Tag' \leftarrow H(K_{compl})$  and compares it with the received hash  $f_{compare}(Tag, Tag')$  to ensure it has not received a modified complementary key  $K_{compl}$ . If a mismatch is found, the sensor node<sub>i</sub> rejects the cipher; otherwise, the sensor node<sub>i</sub> produces its unique key by XORing the complementary key and its local key as follows:  $K_{unique_i} = K_{compl} \oplus K_{local_i}$ . Figure 3.2 shows key distribution phase steps.

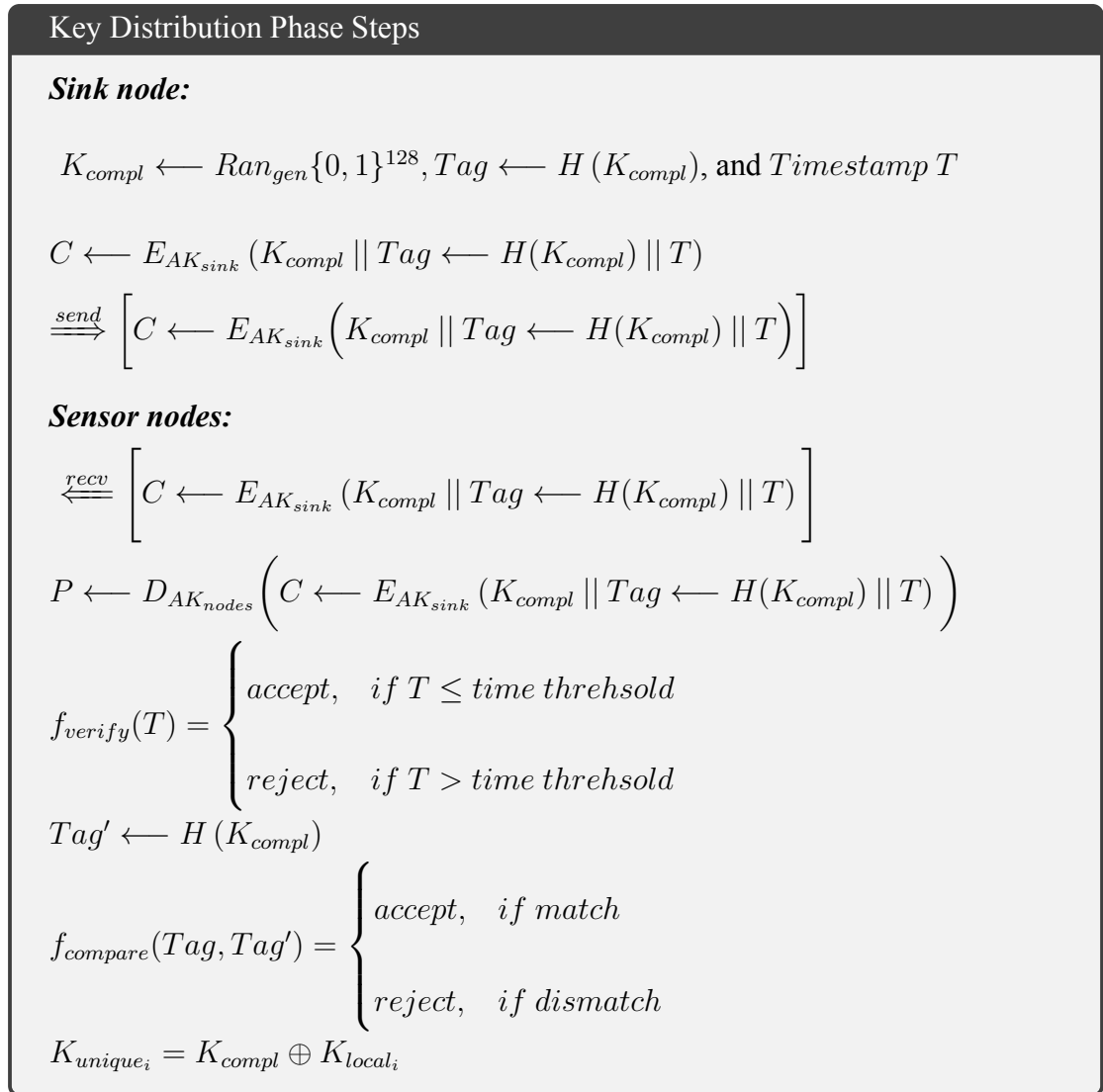


Figure 3.2: Key distribution phase steps.

### 3.3 Post-Key Distribution Phase

After establishing the key distribution phase, the sensor nodes have already produced their unique keys. Therefore, when sensor node<sub>*i*</sub> wants to transmit data  $D$  to the sink node, it uses its unique key  $K_{unique_i}$  to encrypt the data  $D$ , its identity  $Id_i$ , and a timestamp  $T$ ; then, it concatenates the cipher with another copy of its identity  $Id'_i$  and sends both to the sink node. This process is described as follows:  $\xrightarrow{send} \left[ Id'_i \parallel C \leftarrow E_{K_{unique_i}}^\perp (D \parallel Id_i \parallel T) \right]$ , where  $\perp$  is a probabilistic encryption algorithm. Because this study is concerned with key distribution, the sensor node<sub>*i*</sub> can use any secure probabilistic encryption algorithm. When the sink node receives the following cipher  $\xleftarrow{recv} \left[ Id'_i \parallel C \leftarrow E_{K_{unique_i}}^\perp (D \parallel Id_i \parallel T) \right]$ , it uses the concatenated node  $Id'_i$  to find the corresponding  $K_{unique_i}$  as follows:  $F : Id'_i \mapsto K_{unique_i}$ . Then, the sink node decrypts the cipher and compares the identities  $f_{compare}(Id, Id')$ , to ensure that the appropriate  $K_{unique_i}$  is used and that the cipher is received from an authorized node. If a match is found, the sink node verifies the timestamp  $T$ ,  $f_{verify}(T)$ , based on a predefined threshold. When  $T$  is less than or equal to the threshold, the sink node accepts the sensor data  $D$ ; otherwise, it rejects the sensor data  $D$ . Figure 3.3 shows post-key distribution phase steps.

## Post-key Distribution Phase Steps

### **Sensor nodes:**

Sensor data  $D$ , Node $_i$  identity  $Id_i$ , and Timestamp  $T$

$$C \leftarrow E_{K_{unique_i}}^\perp (D \parallel Id_i \parallel T)$$

$$\xrightarrow{\text{send}} \left[ Id'_i \parallel C \leftarrow E_{K_{unique_i}}^\perp (D \parallel Id_i \parallel T) \right]$$

### **Sink node:**

$$\xleftarrow{\text{recv}} \left[ Id'_i \parallel C \leftarrow E_{K_{unique_i}}^\perp (D \parallel Id_i \parallel T) \right]$$

$$F : Id'_i \mapsto K_{unique_i}$$

$$P \leftarrow D_{K_{unique_i}}^\perp \left( C \leftarrow E_{K_{unique_i}}^\perp (D \parallel Id_i \parallel T) \right)$$

$$f_{compare}(Id, Id') = \begin{cases} \text{accept,} & \text{if match} \\ \text{reject,} & \text{if mismatch} \end{cases}$$

$$f_{verify}(T) = \begin{cases} \text{accept,} & \text{if } T \leq \text{time threhsold} \\ \text{reject,} & \text{if } T > \text{time threhsold} \end{cases}$$

Figure 3.3: Post-key distribution phase steps.

### 3.4 Key Refreshment Phase

In the key refreshment phase, the sink node generates a new random complementary key  $K_{compl_{new}} \leftarrow \text{Ran}_{gen}\{0, 1\}^{128}$ , computes its hash value  $Tag \leftarrow H(K_{compl_{new}})$ , and calculates a timestamp  $T$ . The sink node then encrypts these values using its asymmetric key  $AK_{sink}$  and sends the cipher to the neighboring sensor nodes as follows:  $\xrightarrow{send} \left[ C \leftarrow E_{AK_{sink}} \left( K_{compl_{new}} \parallel Tag \leftarrow H(K_{compl_{new}}) \parallel T \right) \right]$ . These neighbors forward the cipher to their neighbors in a multihop fashion until all of the sensor nodes have received the cipher  $\xleftarrow{recv} \left[ C \leftarrow E_{AK_{sink}} \left( K_{compl_{new}} \parallel Tag \leftarrow H(K_{compl_{new}}) \parallel T \right) \right]$ . Because the sensor nodes are loaded with the asymmetric key  $AK_{nodes}$  in the pre-deployment phase, a sensor node<sub>*i*</sub> can decrypt the cipher as follows:  $P \leftarrow D_{AK_{nodes}} \left( C \leftarrow E_{AK_{sink}} \left( K_{compl_{new}} \parallel Tag \leftarrow H(K_{compl_{new}}) \parallel T \right) \right)$  and verifies the timestamp  $f_{verify}(T)$ , based on a predefined threshold. If  $T$  exceeds the threshold, the sensor node<sub>*i*</sub> rejects the cipher; otherwise, the node hashes the new complementary key  $Tag' \leftarrow H(K_{compl_{new}})$  and compares it with the received hash  $f_{compare}(Tag, Tag')$  to ensure it has not received a modified new complementary key  $K_{compl_{new}}$ . If a mismatch is found, the sensor node<sub>*i*</sub> rejects the cipher; otherwise, it produces its new unique key by XORing the new complementary key and its local key as follows:  $K_{unique_{i_{new}}} = K_{compl_{new}} \oplus K_{local_i}$ . Figure 3.4 shows key refreshment phase steps.

## Key Refreshment Phase Steps

### **Sink node:**

$K_{compl_{new}} \leftarrow Ran_{gen}\{0, 1\}^{128}, Tag \leftarrow H(K_{compl_{new}})$ , and *Timestamp T*

$C \leftarrow E_{AK_{sink}}(K_{compl_{new}} || Tag \leftarrow H(K_{compl_{new}}) || T)$

$\xrightarrow{send} \left[ C \leftarrow E_{AK_{sink}}(K_{compl_{new}} || Tag \leftarrow H(K_{compl_{new}}) || T) \right]$

### **Sensor nodes:**

$\xleftarrow{recv} \left[ C \leftarrow E_{AK_{sink}}(K_{compl_{new}} || Tag \leftarrow H(K_{compl_{new}}) || T) \right]$

$P \leftarrow D_{AK_{nodes}} \left( C \leftarrow E_{AK_{sink}}(K_{compl_{new}} || Tag \leftarrow H(K_{compl_{new}}) || T) \right)$

$$f_{verify}(T) = \begin{cases} \text{accept,} & \text{if } T \leq \text{time threhsold} \\ \text{reject,} & \text{if } T > \text{time threhsold} \end{cases}$$

$Tag' \leftarrow H(K_{compl_{new}})$

$$f_{compare}(Tag, Tag') = \begin{cases} \text{accept,} & \text{if match} \\ \text{reject,} & \text{if mismatch} \end{cases}$$

$K_{unique_{i_{new}}} = K_{K_{compl_{new}}} \oplus K_{local_i}$

Figure 3.4: Key refreshment phase steps.

## CHAPTER 4: METHODOLOGY

### 4.1 Experiment Design And Parameters

The experimental design includes two parts: simulation and hardware implementation. In the first part, we utilize the OPNET Modeler to design a model for a wireless sensor node and then used this model to conduct simulations for a network of 200 wireless sensor nodes, as shown in Figure 4.1. Our sensor node model calculates the energy consumption of a node's transceiver, including both the TPO and the energy consumption caused by wireless channel effects, based on the models described in Section 4.2 and Section 4.3, respectively.

In the second part, because the energy consumption by a node's microcontroller cannot be simulated, we implement the proposed protocol and the compared schemes on a real microcontroller and measure the time these schemes require to perform key distribution/establishment processes. Then, we calculate the energy consumption of the node's microcontroller based on the model described in Section 4.2.

Moreover, in this experiment, the transceiver parameters of our sensor node model are based on XBee transceiver S1 [59], and the microcontroller used in the implementation is an Atmega328p [60] (clocked at 16 MHz). Table 4.1 shows these parameters.



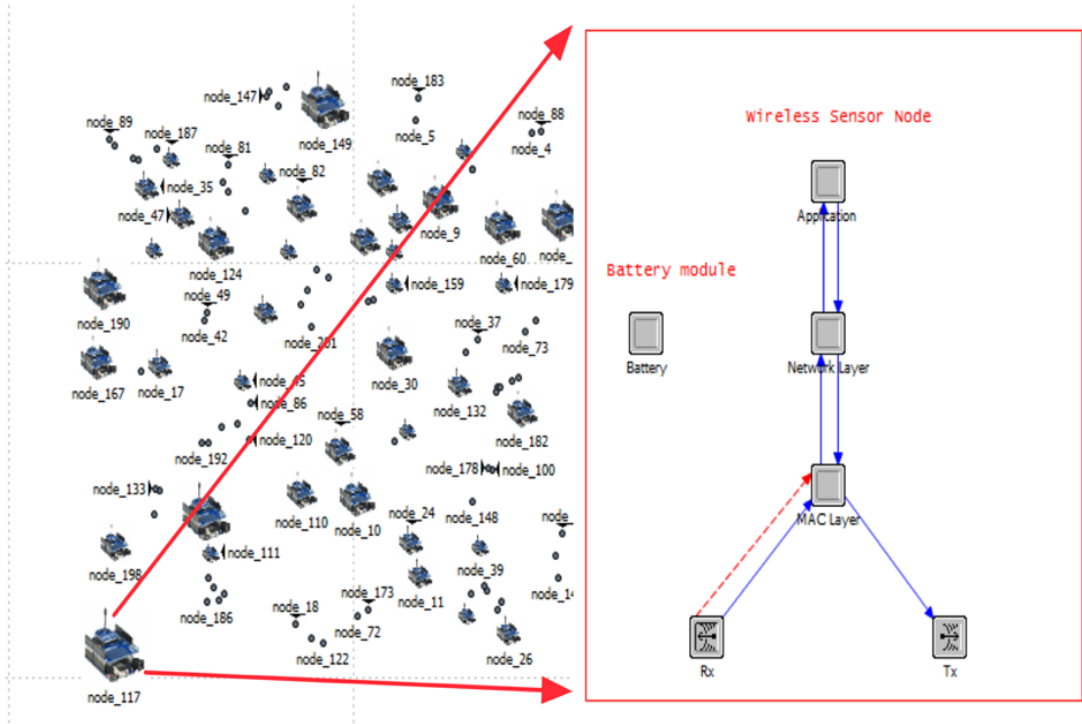


Figure 4.1: Our wireless sensor node model.

Table 4.1: Experiment Parameters.

Description	Parameters	Values
<b>Channel</b>	Data Rate	250 kbps
	Frame Size	1024 bits
	Transmission power	0 dBm
	Modulation	bpsk
	Receiver Sensitivity	-92 dBm
<b>Transceiver</b>	Tx Current draw	45 mA @ 3.3 VDC
	Rx Current draw	50 mA @ 3.3 VDC
<b>Microcontroller</b>	Microcontroller	3.2 mA @ 4.5 V
<b>Power Requirements</b>	Tx power consumption	148.5 mW
	Rx power consumption	165 mW

## 4.2 Energy Consumption Of Wireless Sensor Node

The energy consumption of a wireless sensor node is the total amount of energy consumed by the node's transceiver, microcontroller, and sensors, as indicated in equation (4.1). Because our research considers key distribution security and efficiency regardless of the network's application, we ignore the energy consumed by the sensors because that consumption is based on sensor applications regardless of the underlying key distribution/establishment algorithm. Thus, in this study, the energy consumption of a wireless sensor node is calculated by equation (4.2):

$$E_{sensor\ node} = E_{transceiver} + E_{microcontroller} + E_{sensor} \quad (4.1)$$

$$E_{sensor\ node} = E_{transceiver} + E_{microcontroller}. \quad (4.2)$$

The energy consumption is generally calculated by equation (4.3):

$$Energy(J) = Power(W) \cdot Time(S), \quad (4.3)$$

where  $Power(W)$  is calculated by equation (4.4):

$$Power(W) = Voltage(V) \cdot Current(A). \quad (4.4)$$

Based on equation (4.3), we calculate the energy consumption of nodes' transceivers and microcontrollers. The following subsections describe these calculations.

### 4.2.1 Energy Consumption Of The Transceiver

The energy consumption of a node's transceiver  $E_{transceiver}$ , given in equation (4.2), is the energy consumed by the node's transmitter  $E_{Tx}$  and its receiver  $E_{Rx}$ . However, the energy consumption of the node's transmitter  $E_{Tx}$  is the total energy consumed by the TPO  $E_{TxTPO}$  and the transmitter electronics  $E_{Txelec}$ , which can be found by the following equation:

$$E_{Tx} = E_{TxTPO} + E_{Txelec}, \quad (4.5)$$

where the energy consumed by  $E_{TxTPO}$  is found by equation (4.6):

$$E_{TxTPO} = Tx_{poweroutput} \cdot Tx_{time}, \quad (4.6)$$

where  $Tx_{poweroutput}$  is the TPO, and it depends on the transceiver module (described in Section 4.1).  $Tx_{time}$  is the time that a node's transmitter takes to send one frame and is found by equation (4.7):

$$Tx_{time} = \left( \frac{Fr_{Size}}{Tx_{DR}} \right), \quad (4.7)$$

where  $Fr_{Size}$  is the frame size, and  $Tx_{DR}$  is the data rate of the transmitter. The transmitter data rate depends on the transceiver module (described in Section 4.1).

The energy consumed by the transmitter electronics  $E_{Tx_{elec}}$  can be calculated as follows:

$$E_{Tx_{elec}} = Tx_{time} \cdot ((Tx_{elec_{current}} \cdot Tx_{elec_{voltage}}) + (MA_{Tx_{current}} \cdot MA_{Tx_{voltage}})), \quad (4.8)$$

where  $Tx_{time}$  is found as shown in equation (4.7).  $Tx_{elec_{current}}$  and  $Tx_{elec_{voltage}}$  are the respective current and voltage required by the transmitter's electronics, and  $MA_{Tx_{current}}$  and  $MA_{Tx_{voltage}}$  are the respective current and voltage required by a node's microcontroller to run the transmitter.

The energy consumption of the node's receiver  $E_{Rx}$  can be calculated as follows:

$$E_{Rx} = Rx_{time} \cdot ((Rx_{current} \cdot Rx_{voltage}) + (MA_{Rx_{current}} \cdot MA_{Rx_{voltage}})), \quad (4.9)$$

where  $Rx_{time}$  is the time that a node's receiver requires to receive one frame, which can be found by equation (4.10).  $Rx_{current}$  and  $Rx_{voltage}$  are the respective current and voltage required by a node's receiver, and  $MA_{Rx_{current}}$  and  $MA_{Rx_{voltage}}$  are the respective current and voltage required by a node's microcontroller to run the receiver.

$$Rx_{time} = \left( \frac{Fr_{Size}}{Rx_{DR}} \right), \quad (4.10)$$

where  $Fr_{Size}$  is the frame size, and  $Rx_{DR}$  is the data rate of the receiver. The receiver's data rate depends on the transceiver module (described in Section 4.1).

### 4.2.2 Energy Consumption Of The Microcontroller

The energy consumption of a node's microcontroller,  $E_{microcontroller}$ , presented in equation (4.2), represents the energy consumed by a node's microcontroller,  $E_{MA}$ , and can be found using the following equation:

$$E_{MA} = MA_{Time_{operate}} \cdot MA_{current} \cdot MA_{voltage}, \quad (4.11)$$

$MA_{Time_{operate}}$  is the time required by the microcontroller to execute the core algorithm of a key distribution/establishment process.  $MA_{current}$  and  $MA_{voltage}$  are the respective current and voltage required by the microcontroller.

### 4.3 Modeling Wireless Channel Effects

When our sensor node model receives a frame, it calculates the received power  $P_{Rx}$  of the frame as follows:

$$P_{Rx} = P_{Tx} \cdot G_{Tx} \cdot G_{Rx} \cdot P_{Loss}, \quad (4.12)$$

where  $P_{Tx}$  is the transmitter power,  $G_{Tx}$  is the transmitter antenna gain,  $G_{Rx}$  is the receiver antenna gain, and  $P_{Loss}$  is the path loss in free space, which can be found by the following equation:

$$P_{Loss} = \left( \frac{\lambda}{4\pi D} \right)^2, \quad (4.13)$$

where  $\lambda$  is the wavelength, and  $D$  is the distance.

If the received power is less than the receiver sensitivity, our model discards the frame because the receiver is unable to decode it. However, if the received power is greater than or equal to the receiver's sensitivity, our model calculates the noise in each frame caused by interference from other frames and then calculates the signal-to-noise ratio (SNR) based on equation (4.14):

$$SNR = 10 \log_{10} \left( \frac{P_{Rx}}{N_{Intern}} \right). \quad (4.14)$$

Then, our model calculates the bit error rate (BER) in the received frame, which is calculated based on the chosen modulation curve and the SNR. Finally, it calculates the percentage of errors in the received frame by the following equation:

$$F_{Errors} = \left( \frac{BER}{F_{size}} \right), \quad (4.15)$$

where  $F_{size}$  is the frame size. If the percentage of errors in the received frame  $F_{Errors}$  exceeds a specified threshold, the model discards the frame.

## 4.4 Fast Modular Exponentiation Algorithm

---

**Algorithm:** Square-and-multiply

---

**Input** : base  $a$ ; mod  $n$ ; binary representation of exponent  $e$

$[e_t, e_{t-1}, \dots, e_0]_2$

**Output:**  $a^e \bmod n$

```
1  $b \leftarrow 1$ 
2 for ( $i = t$ ;  $i > 0$ ;  $i --$ ) do
3    $b = b \cdot b \bmod n$ 
4   if ( $e_i == 1$ ) then
5      $b = b \cdot a \bmod n$ 
6   end
7 end
8 return  $b$ 
```

---

The proposed protocol relies on modular exponentiation in the key distribution phase. The running time for an ordinary modular exponentiation algorithm is exponential with the magnitude of the exponent  $e$  (linear complexity); thus the algorithm is not a practical. To make the proposed protocol practical in resource-constrained nodes, we utilized the square-and-multiply algorithm, which has a polynomial running time in the length of  $e$  (logarithmic complexity).

## CHAPTER 5: FINDINGS AND ANALYSES

In this section, we analyze the efficiency and security of the proposed protocol in comparison to the schemes proposed in [35], [44], [45], and [55]. These analyses are based on the metrics presented in Table 2.1.

### 5.1 Efficiency Analysis

#### 5.1.1 Energy Consumption

Table 5.1 compares the energy consumption required by the proposed protocol and the corresponding schemes to perform a key distribution/establishment process between two nodes. (Because some schemes perform key distribution and others conduct key establishment, we refer to both terms as the “key distribution/establishment process.”). The experimental design and parameters are described in Chapter 4, Section 4.1.

The first part of the table includes eight rows to quantify the energy consumed by the nodes’ transceivers. The first row, “Th.N.F.Tx,” represents the theoretical number of frames sent by a node’s transmitter when performing a key distribution/establishment process without modeling wireless channel effects. The second row, “Av.N.F.Tx,” shows the average number of frames sent after modeling the wireless channel effects. The third row, “N.F.Rx,” shows the number of frames that a node’s receiver receives



during the key distribution/establishment process. If a scheme involves exchanged frames for node discovery or clustering before the key distribution/establishment process, the fourth row, “N.F.ND.C,” represents the number of those frames. The fifth row, “T.Tx,” shows the time the node’s transmitter requires to send the frames. The sixth row, “T.Rx,” shows the time the node’s receiver requires to receive the frames. The seventh row, “E.TPO,” shows the energy consumed by the transmitter power output (TPO). The eighth row, “E.TRX,” shows the total energy consumed by the nodes’ transceivers.

The second part of the table quantifies the energy consumption of the nodes’ microcontrollers and includes eight rows. The first row, “S.C.K,” represents the search complexity for finding a common key, where  $n$  is the number of nodes,  $P$  is the key pool size, and  $P_C$  is the probability that two nodes share a key. The next six rows show the time a node’s microcontroller requires to perform the various operations required to complete the key distribution/establishment process. The “T.MA.K” row shows the time a node’s microcontroller requires to find a common key; “T.MA.X” is the time a node’s microcontroller requires to perform an XOR operation; “T.MA.H” is the time a node’s microcontroller requires to perform a hashing operation; “T.MA.E” is the time a node’s microcontroller requires for encryption; and “T.MA.D” is the time a node’s microcontroller requires for decryption. When a scheme involves polynomial evaluation, “T.MA.PE” is the time a node’s microcontroller requires for polynomial evaluation. The eighth row, “E.MA,” shows the total energy consumed by the nodes’ microcontrollers.

The last part of the table, “T.C.E,” shows the total energy consumption of the proposed protocol and the corresponding schemes.

Table 5.1: Energy consumption of each key distribution scheme

Descriptions	Schemes		Our Protocol	Scheme[35]	Scheme[44]	Scheme[45]	Scheme[55]
	Parameters						
The parameters that contribute to the energy consumption of nodes' transceivers.	Th.N.F.Tx		1	27	6	95	2
	Av.N.F.Tx		3	39	13	120	4
	N.F.Rx		1	27	6	95	2
	N.F.ND.C		NA	2	201	NA	NA
	T.Tx		12.29 ms	159.74 ms	876.54 ms	491.52 ms	16.38 ms
	T.Rx		4.10 ms	110.59 ms	847.87 ms	389.12 ms	8.19 ms
	E.TPO		0.01 mJ	0.16 mJ	0.88 mJ	0.49 mJ	0.02mJ
	E.TRX		2.75 mJ	46.02 mJ	295.77 mJ	150.37 mJ	4.15 mJ
	C.S.K		$\mathcal{O}(1)$	$\mathcal{O}(2 \cdot \sqrt{n-1})$	$\mathcal{O}(1)$	$\mathcal{O}(\sqrt{-P \cdot \log(1 - P_c)})^1$	$\mathcal{O}(n)$
	T.MA.K		NA	10.08 ms	NA	89.31 ms	170.02 ms
T.MA.X		0.35 ms	NA	NA	NA	NA	
T.MA.H		177.82 ms	NA	NA	NA	NA	
T.MA.E		982 ms	$t^2$	NA	NA	982 ms	
T.MA.D		1502.90 ms	$t^2$	NA	NA	1502.90 ms	
T.MA.PE		NA	NA	1909.83 ms	NA	NA	
E.MA		38.35 mJ	0.15 mJ	27.50 mJ	1.29 mJ	38.23 mJ	
Total energy consumption		41.10 mJ	$46.17 + 2t^2$ mJ	323.28 mJ	151.65 mJ	42.39 mJ	

<sup>1</sup> The key pool size  $P$  is equal to the number of sensor nodes multiplied by 10, and  $P_c$  is equal to 0.99 %

<sup>2</sup> Here,  $t$  denotes an unknown time; the time cannot be determined because the scheme involves encryption and decryption during the key distribution process, and the source did not specify the type of encryption and decryption algorithm used.

As shown in Table 5.1, the nodes' transceivers consume the lowest amount of energy in the proposed protocol, followed by scheme[55], at  $2.75 \text{ mJ}$  and  $4.15 \text{ mJ}$ , respectively; whereas, the nodes' transceivers consume the largest amount of energy in scheme[44], followed by scheme[45] and scheme[35], at  $295.77 \text{ mJ}$ ,  $150.37 \text{ mJ}$ , and  $46.02 \text{ mJ}$ , respectively.

In contrast, the nodes' microcontrollers consume the lowest amount of energy in scheme[35], followed by scheme[45] and scheme[44], at  $0.15 \text{ mJ}$ ,  $1.29 \text{ mJ}$ , and  $27.50 \text{ mJ}$ , respectively, and the largest amount of energy consumed is in the proposed protocol, followed by scheme[55], at  $38.35 \text{ mJ}$  and  $38.23 \text{ mJ}$ , respectively.

However, the total energy consumption in the proposed protocol is  $41.10 \text{ mJ}$ , which is the lowest relative to the total energy consumed by each of the corresponding schemes. This result is intuitive because the proposed protocol is designed to perform key distribution based on efficient data computation rather than data communication.

### 5.1.2 Key Storage Overhead

Table 5.2: Key Storage overhead for each scheme.

Scheme	Key Storage Overhead
The proposed protocol	3
Scheme[35]	$2 \cdot \sqrt{n-1}$
Scheme[44]	6
Scheme[45]	$\sqrt{-P \cdot \log(1 - P_c)}^1$
Scheme[55]	$n$

$n$  represents the number of nodes.

<sup>1</sup> The key pool size  $p$  is equal to the number of sensor nodes multiplied by 10, and  $P_c$  is equal to 0.99 %.

Table 5.2 illustrates the key storage overhead of the proposed protocol and the corresponding schemes. In the pre-deployment phase of the proposed protocol, each sensor node stores two keys,  $AK_{nodes}$  and  $K_{local}$ . Then, in the key distribution phase, each sensor node prepares its unique key,  $K_{unique}$ . Therefore, the proposed protocol has the lowest key storage overhead for each sensor node compared to the corresponding schemes. The logarithmic graph presented in Figure 5.1 shows the magnitude of the key storage overhead as the number of sensor nodes increases. The graph clearly shows that the proposed protocol is advantageous because it requires the fewest keys compared to other schemes.

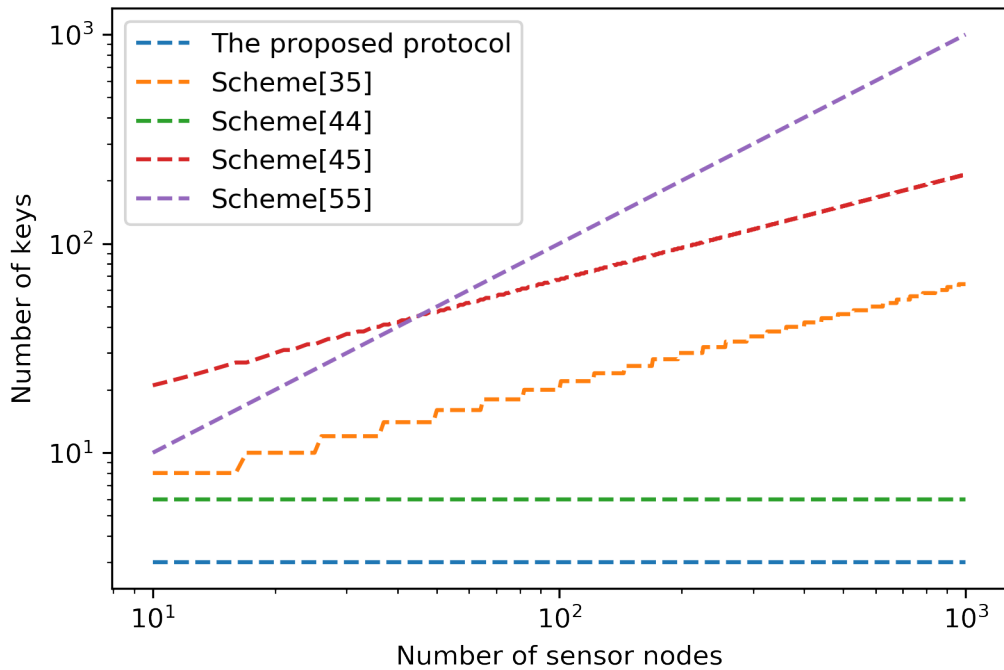


Figure 5.1: The magnitude of key storage overhead

### 5.1.3 Key Connectivity

To evaluate the key connectivity in each scheme, we model the entire WSN with a graph in which the vertices represent wireless sensor nodes and the edges represent links. Therefore, Figure (5.2-a) shows a random deployment of 200 nodes over an area of 1000 ft · 1000 ft. In Figure (5.2-b), the black links indicate the wireless signal range of the nodes' transceivers without applying either the proposed protocol or the corresponding schemes. The wireless signal range is based on the nodes' transceiver modules (described in Chapter 4, Secion 4.1).

Figure 5.3 shows the implementation of the proposed protocol and the corresponding schemes on the WSN shown in Figure (5.2-b). However, when two sensor nodes share a common key or key material within the same wireless range, the black links are converted to green links. In contrast, the red links indicate nodes that do not share a common key or any key materials. This modeling shows the key connectivity of each scheme, which can be defined as follows:

$$\frac{\textit{Secured links}}{\textit{Total number of links}} \cdot 100, \quad (5.1)$$

where the term "*Secured links*" includes any link between two nodes that share a common key or key materials, and "*Total number of links*" counts all links in the WSN. Figure (5.3-a), (5.3-c) and (5.3-e) show that the key connectivity is certain because nodes in these schemes are designed to have either a shared common key or key materials that lead to 100 % key connectivity. In contrast, the key connectivities in (5.3-b) and (5.3-d) reach only 87 % and 99 %, respectively.

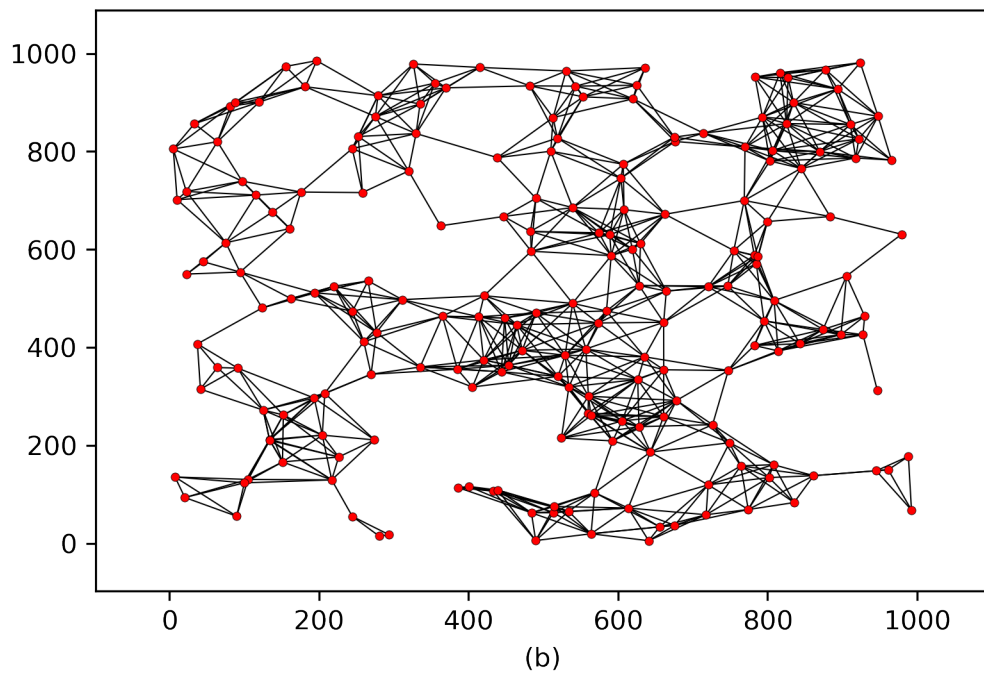
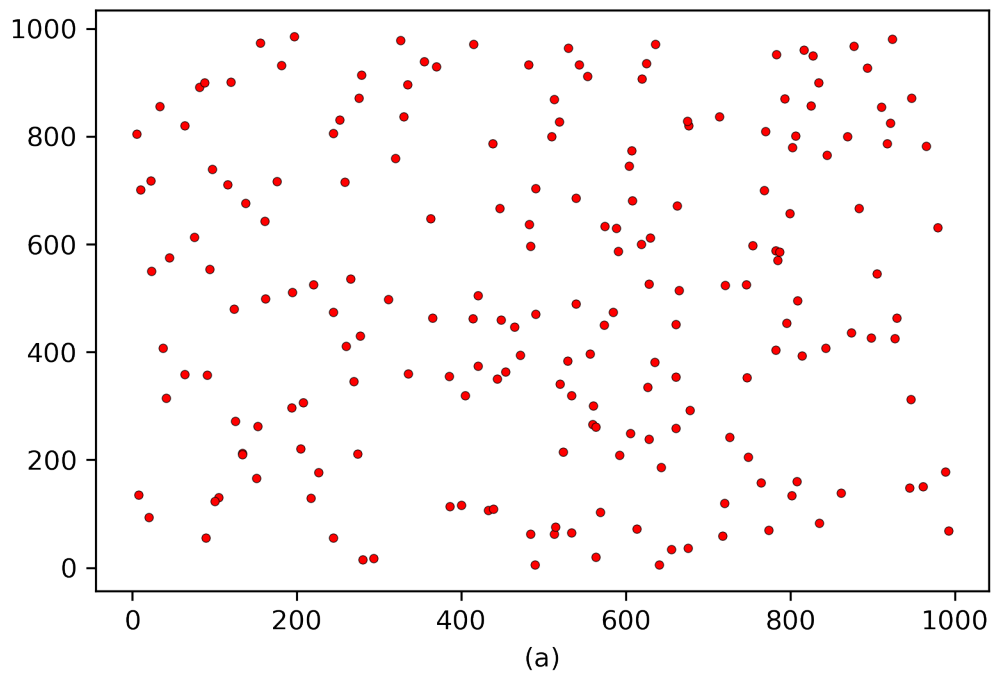
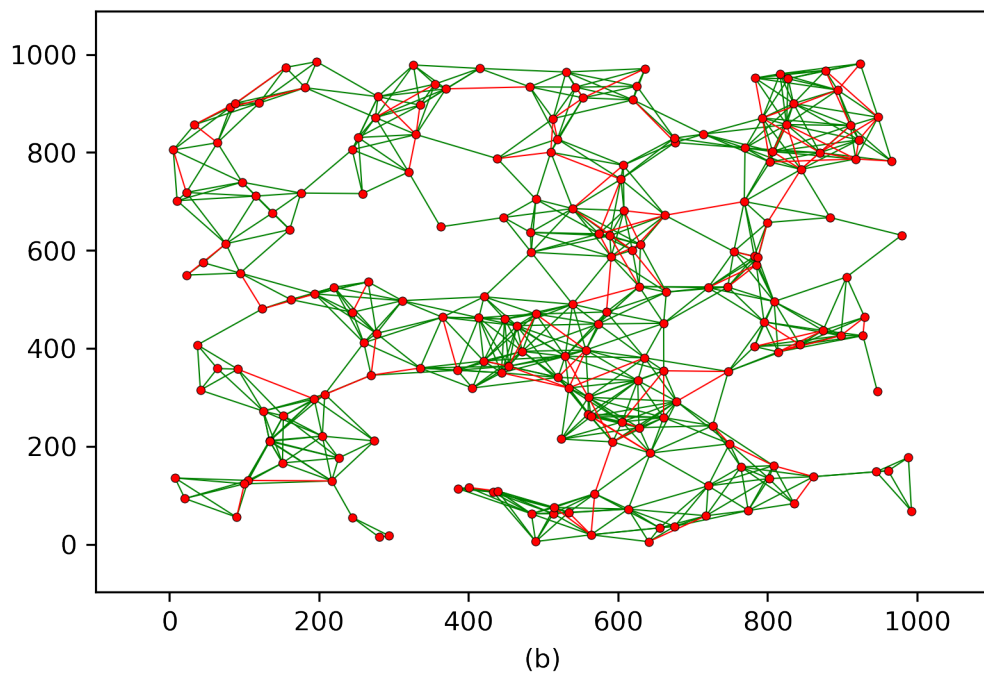
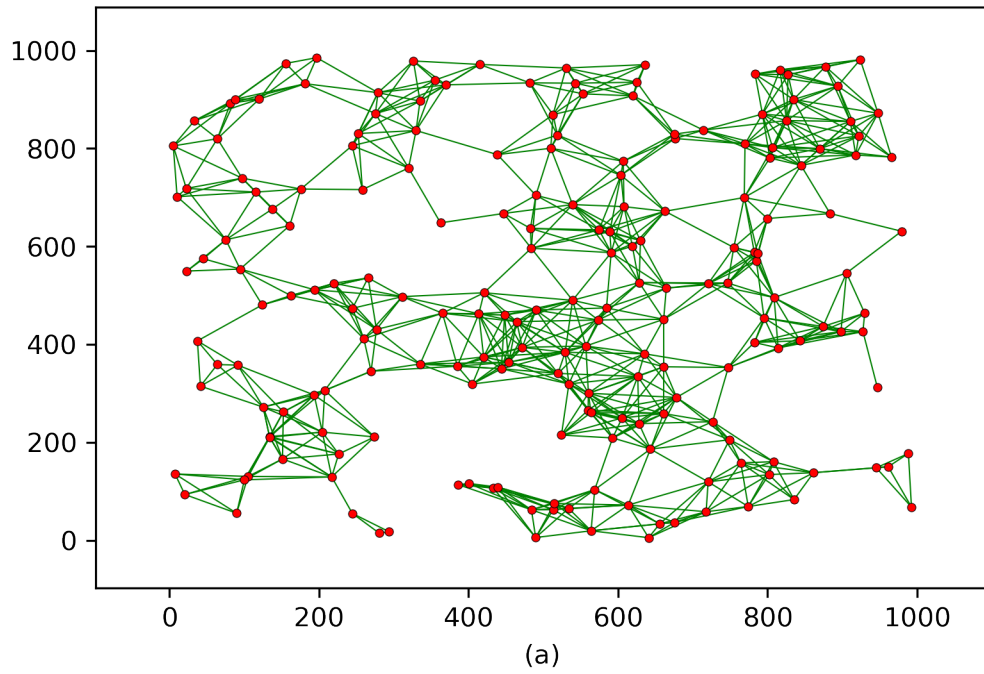
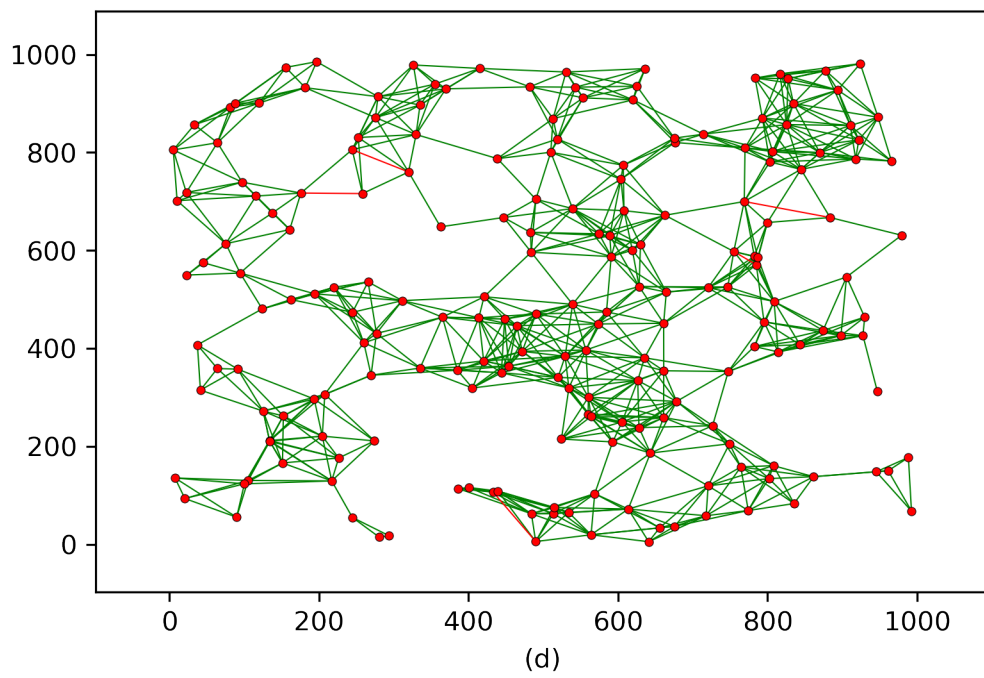
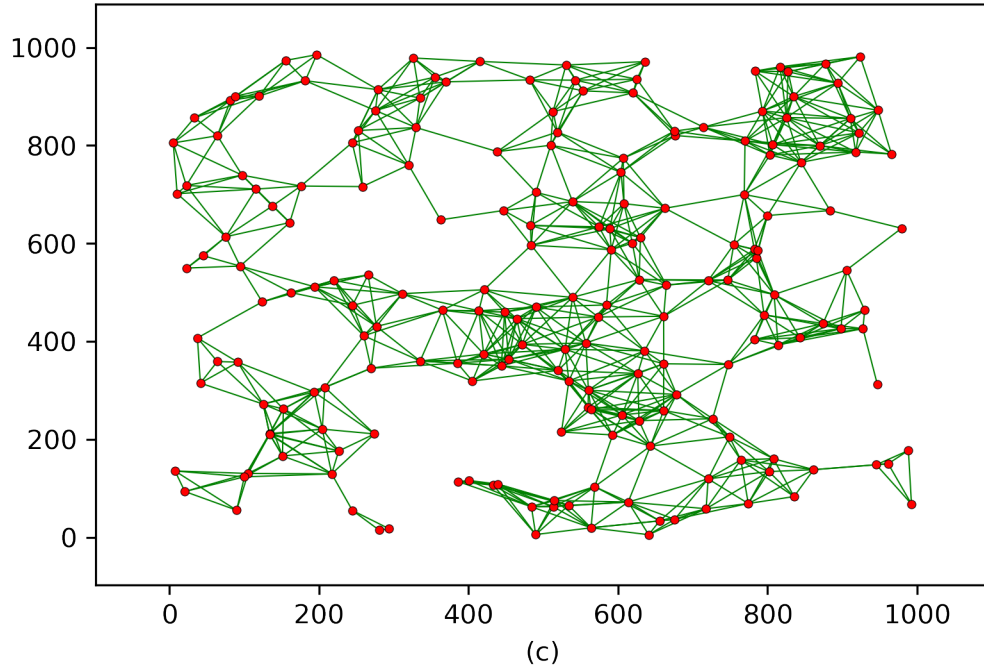


Figure 5.2: Modeling a WSN: (a) random deployment of sensor nodes; (b) wireless signal range of nodes' transceivers.







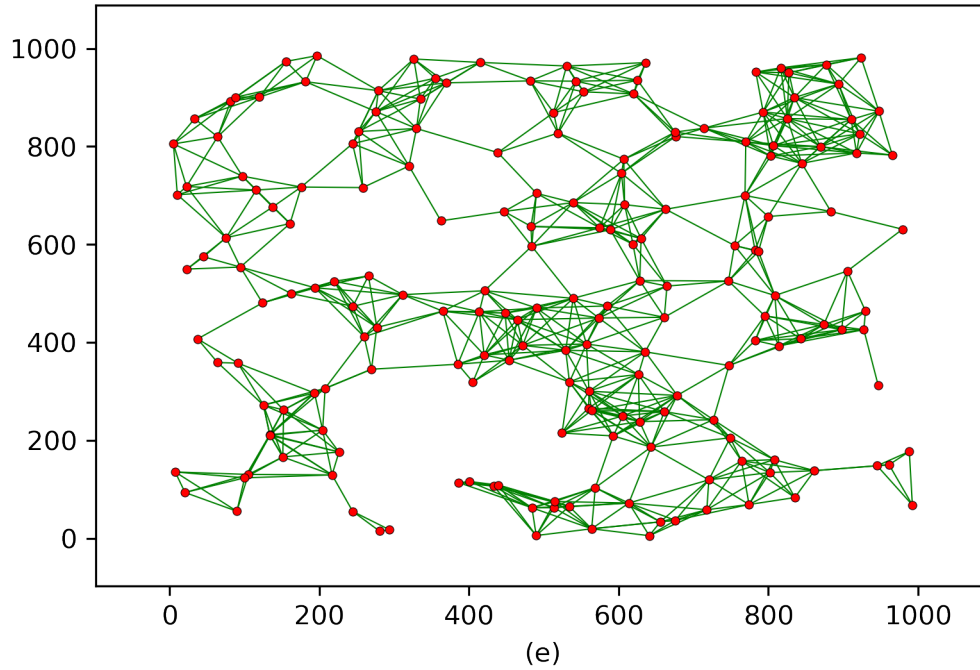


Figure 5.3: Key connectivity after implementing key distribution/establishment process for each scheme. (a) the proposed protocol; (b) scheme[35]; (c) scheme[44]; (d) scheme[45]; (e) scheme[55]

## 5.2 Security Analysis

### 5.2.1 Replay Attack

In a replay attack, an adversary captures a copy of exchanged frames to resend them later to the receiver for a deceptive purpose. The goal is for the receiver to believe that the resent messages are new messages; however, the receiver receives old information. This type of attack cannot be performed against the proposed protocol because a timestamp  $T$  is employed as a countermeasure in all three on-line phases: key distribution, post-key distribution and key refreshment. In the key distribution and key refreshment phases, the timestamp  $T$  is appended to each complementary key sent to the sensor nodes:

$$\xrightarrow{\text{send}} \left[ C \leftarrow E_{AK_{sink}} \left( K_{compl} \parallel Tag \leftarrow H(K_{compl}) \parallel T \right) \right]$$

$$\xrightarrow{\text{send}} \left[ C \leftarrow E_{AK_{sink}} \left( K_{compl_{new}} \parallel Tag \leftarrow H(K_{compl_{new}}) \parallel T \right) \right].$$

In the post-key-distribution phase, each sensor node appends a timestamp  $T$  to each frame sent to the sink node.

$$\xrightarrow{\text{send}} \left[ Id'_i \parallel C \leftarrow E_{K_{unique_i}}^\perp (D \parallel Id_i \parallel T) \right].$$

This timestamp allows the sink node to validate whether the received data are replayed data. However, none of the corresponding schemes implement a countermeasure against replay attacks.

### 5.2.2 Man-In-The-Middle Attack

During a man-in-the-middle attack, an adversary secretly intercepts frames from the sender and likely modifies them. Then, the adversary resends the frames to the receiver. This process occurs without the knowledge of the sender and receiver; therefore, both parties assume that they are communicating directly with one another. However, the proposed protocol is secure against man-in-the-middle attacks because in the pre-deployment phase, an asymmetric key pair is generated, renamed  $AK_{sink}$  and  $AK_{nodes}$ , and loaded into the sink node, and the sensor nodes, respectively.

$$\{K_P, K_R\} \leftarrow RSA_{gen}$$

$$K_P \stackrel{\text{def}}{=} AK_{sink} \text{ and } K_R \stackrel{\text{def}}{=} AK_{nodes}.$$

Thus, during the key distribution and key refreshment phases, when the sink node encrypts the complementary key with  $AK_{sink}$ :

$$C \leftarrow E_{AK_{sink}}(K_{compl} || Tag \leftarrow H(K_{compl}) || T).$$

Only the sensor nodes are able to decrypt the cipher because they already possess one key of the asymmetric key pair.

Additionally, in the post-key distribution phase, each sensor node encrypts data with its unique key that is only in the possession of the sensor node and the sink node.

$$C \leftarrow E_{K_{unique_i}}^\perp(D || Id_i || T).$$

Therefore, an adversary cannot impersonate either the sink node or any sensor node in the proposed protocol. Scheme [55] is vulnerable to man-in-the-middle attacks; however, the other compared schemes are not subjected to this type of attack.

### 5.2.3 Node Capture Attack

To investigate the resilience of the proposed protocol and the corresponding schemes against node capture attacks, node capture attacks must be launched on the key connectivity of each scheme. Therefore, as shown in Figure 5.4, we mount node

capture attacks on the key connectivity of each scheme presented in Figure 5.3. Thus, Figure 5.4 shows each scheme's resilience against node capture attacks. The impact of a node capture attack can be defined as follows:

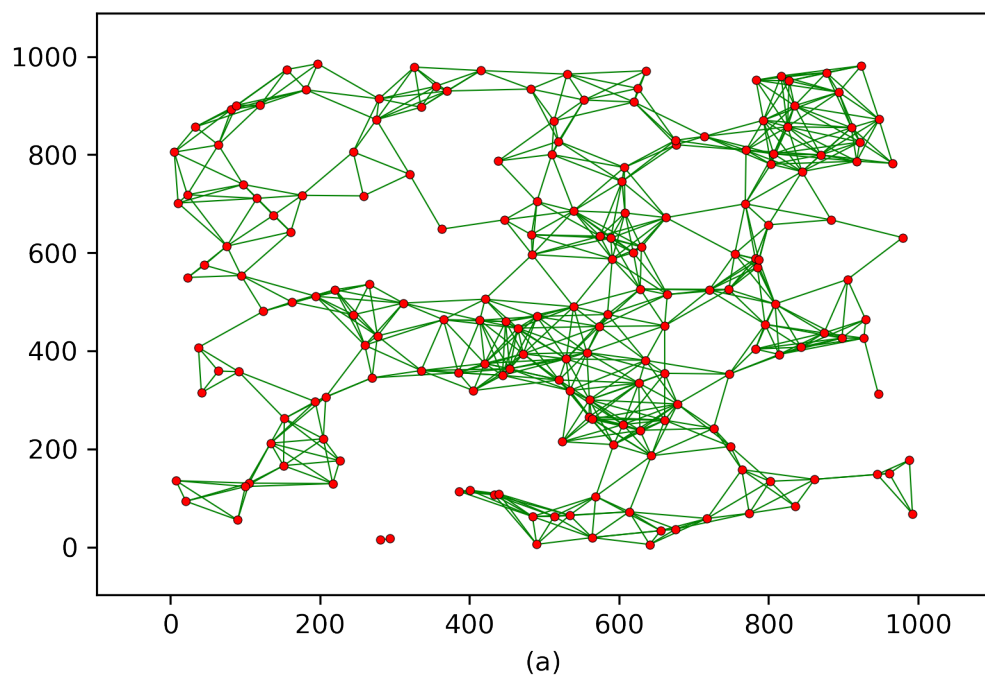
$$\frac{\textit{Compromised links}}{\textit{Total number of secured links}} \cdot 100, \quad (5.2)$$

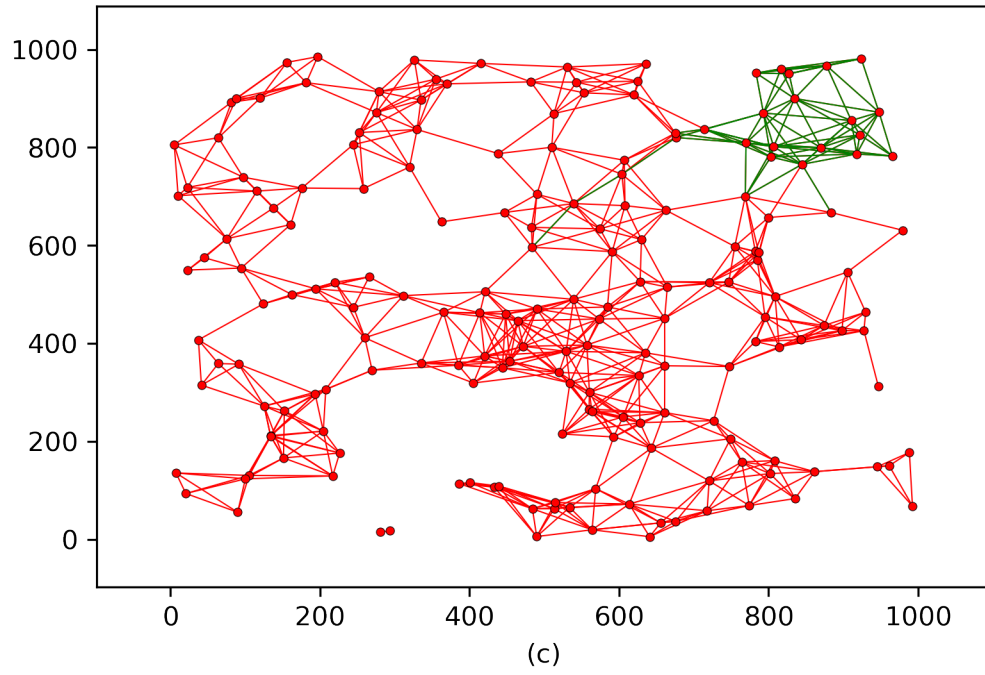
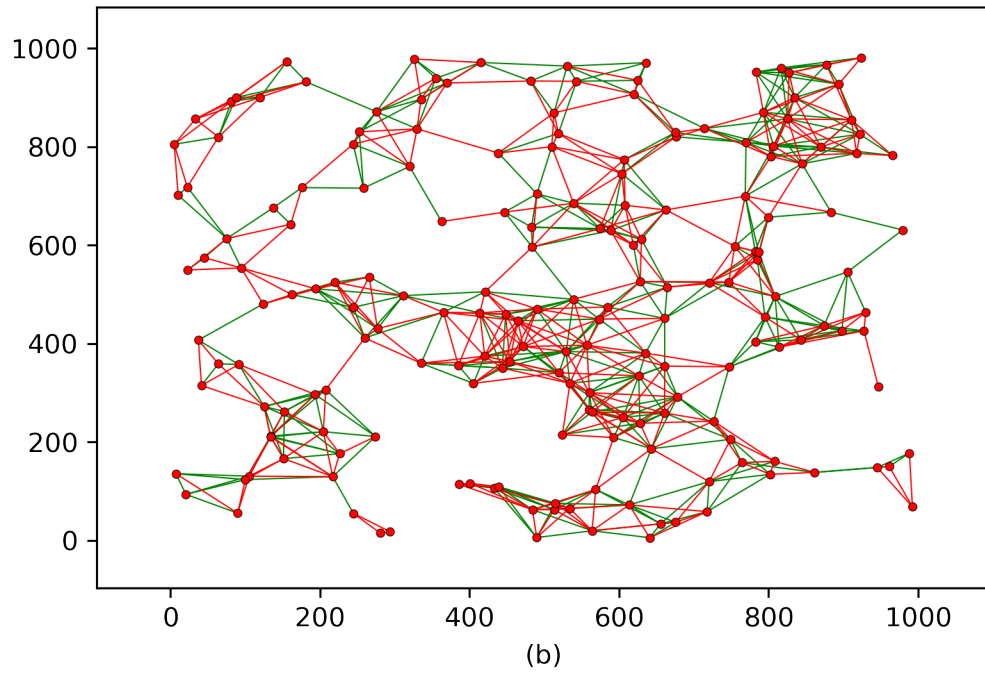
where “*Compromised links*” indicates the number of links that are compromised after a random number of sensor nodes have been captured, and “*Total number of secured links*” counts any link between two nodes that share a common key or key materials. However, each scheme has a different design; thus, for fairness, we assume the following:

- An adversary is able to physically capture 5 % of the sensor nodes randomly (i.e., 10 sensor nodes in the example network).
- Because capturing the sink node will compromise any given WSN, in Assumption 1, node capture does not include the sink node.
- Capturing a sensor node reveals all the data that node contains. For example, if the captured sensor node contains data that reveal information about other nodes' common keys or keys materials, those keys are also compromised.

In Figure (5.4-a), the key connectivity of the proposed protocol does not change after 10 sensor nodes are captured, which indicates that the proposed protocol is secure against node capture attacks. However, the key connectivities of the networks in (5.4-b) and (5.4-c) are decreased by 51 % and 75 %, respectively (assuming the network in (5.4-c) has 4 clusters and that some of the compromised nodes are located in 3 different

clusters). The key connectivity in (5.4-d) is only slightly affected, decreasing by only 6 %. In contrast, the network in (5.4-e) is heavily impacted, and the key connectivity is decreased by 100 %.





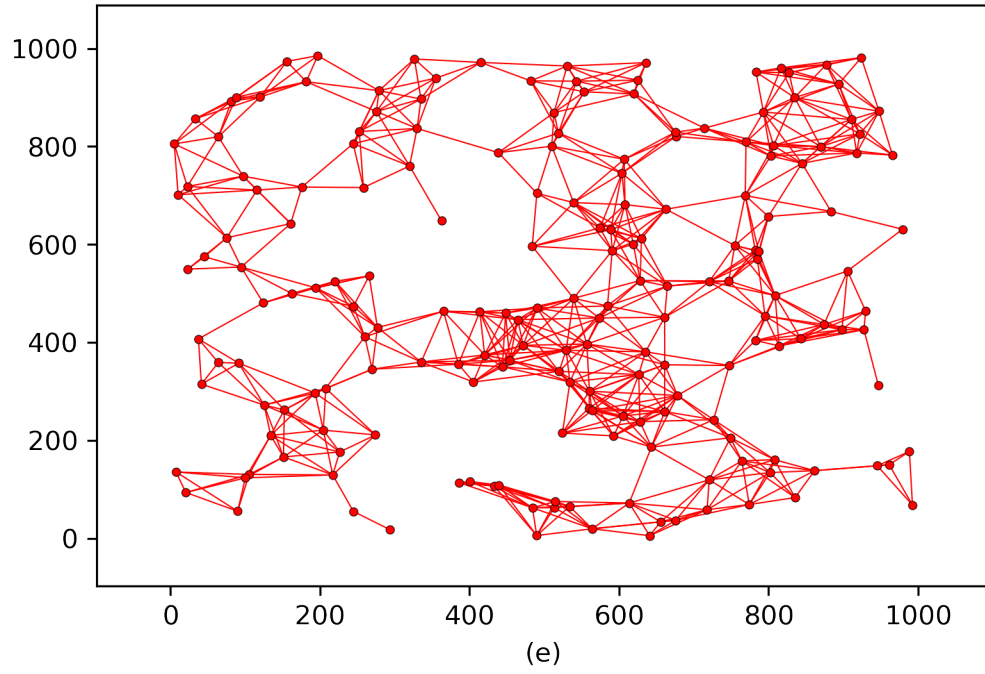
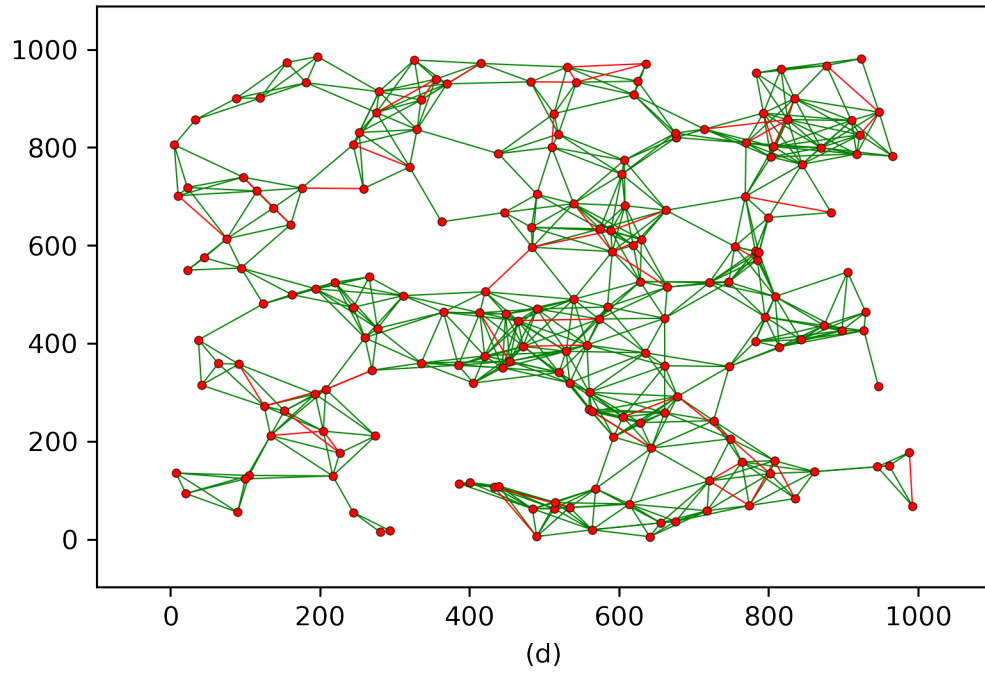


Figure 5.4: Schemes' resilience against node capture attacks. (a) The proposed protocol; (b) Scheme[35]; (c) Scheme[44]; (d) Scheme[45] ; (e) Scheme[55]

## CHAPTER 6: FORMAL VERIFICATION

To formally prove the security and soundness of the proposed protocol, we utilize ProVerif, an automatic cryptographic protocol verifier. ProVerif is a powerful tool for automatically analyzing the security of cryptographic protocols and verifying them in a formal model.

In this section, we present the verification results pertaining to reachability and secrecy, correspondence assertions (authentication), and observational equivalences.

### 6.1 Reachability And Secrecy

ProVerif provides proof of reachability and secrecy properties by investigating the reachability of a term  $x$  by an adversary  $\mathcal{A}$ . Based on the results, the secrecy of  $x$  can be assessed with respect to the modeled protocol. In the proposed protocol, we test whether sensor data “sensorData” are available to  $\mathcal{A}$ . Figure 6.1 shows the complete verification result. The result concludes, “RESULT not attacker(sensorData[ ]) is true”, meaning that “sensorData” is unreachable, and an attack cannot be conducted against the protocol successfully.



## Reachability and Secrecy

Process:

```
{1}new senkey: prkey;
{2}let sinkey: pukey = pk(senkey) in
{3}new sensor1LocKey: localK;
{4}new sensor2LocKey: localK;
{5}new uK: uniquKey;
{6}insert sinkt(sensorA,sensor1LocKey);
{7}insert sinkt(sensorB,sensor2LocKey);
(
  {8}!
  {9}new k_118: compKey;
  {10}let tag: compKey = h(k_118) in
  {11}new t: time;
  {12}new r_119: Padding;
  {13}let c: bitstring = internal_aenc((k_118,tag,t),sinkey,r_119) in
  {14}out(ch, c);
  {15}in(ch, (senIdX: sensors,y_120: bitstring));
  {23}get sinkt(=senIdX,slocl: localK) in
  {16}let SenUniqueKey': uniquKey = xor(slocl,k_118) in
  {17}let (AuthIden: sensors,tz: time,s2: bitstring) =
sdecrypt(y_120,SenUniqueKey') in
  {18}if (senIdX = AuthIden) then
```

```

    {19}event acceptsSink(tz);

    {20}new sit: time;

    {21}if (tz = sit) then

    {22}event termSink(AuthIden)
  )|(
    {24}!

    {25}in(ch, x_121: bitstring);

    {26}let (cK: compKey,tagX: compKey,tx: time) = decrypt(x_121,senkey) in

    {27}new snt: time;

    {28}if (tx = snt) then

    {29}let tag2: compKey = h(cK) in

    {30}if (tagX = tag2) then

    {31}event acceptsSensor(cK,tx);

    {32}let SenUniqueKey: uniquKey = xor(sensor1LocKey,cK) in

    {33}new ty: time;

    {34}new r_122: Randomness;

    {35}let c': bitstring =
internal_senc((sensorA,ty,sensorData),SenUniqueKey,r_122) in

    {36}out(ch, (sensorA,c'));

    {37}event termSensor(sensorA)
  )
  – Query not attacker(sensorData[ ])

  Completing...

```

```
Starting query not attacker(sensorData[ ])
```

```
RESULT not attacker(sensorData[ ]) is true.
```

Figure 6.1: Verification result of reachability and secrecy.

## 6.2 Correspondence Assertions

In ProVerif, authentication can be modeled using a sequence of events defined as correspondence assertions. We apply a sequence of events to verify the authentication of the sink node and the complementary key to the sensor nodes and the authentication of the sensor nodes and encrypted data to the sink node. Figure 6.2 shows the complete verification result. The verification confirms that the proposed protocol achieves successful authentication.

### Correspondence Assertions

Process:

```
{1}new senkey: prkey;
```

```
{2}let sinkey: pukey = pk(senkey) in
```

```
{3}new sensor1LocKey: localK;
```

```
{4}new sensor2LocKey: localK;
```

```
{5}new uK: uniquKey;
```

```
{6}insert sinkt(sensorA,sensor1LocKey);
```

```
{7}insert sinkt(sensorB,sensor2LocKey);
```

```
(
```

```

{8}!

{9}new k_124: compKey;

{10}let tag: compKey = h(k_124) in

{11}new t: time;

{12}new r_125: Padding;

{13}let c: bitstring = internal_aenc((k_124,tag,t),sinkey,r_125) in

{14}out(ch, c);

{15}in(ch, (senIdX: sensors,y_126: bitstring));

{23}get sinkt(=senIdX,slocl: localK) in

{16}let SenUniqueKey': uniquKey = xor(slocl,k_124) in

{17}let (AuthIden: sensors,tz: time,s2: bitstring) =
sdecrypt(y_126,SenUniqueKey') in

{18}if (senIdX = AuthIden) then

{19}event acceptsSink(tz);

{20}new sit: time;

{21}if (tz = sit) then

{22}event termSink(AuthIden)

)|(

{24}!

{25}in(ch, x_127: bitstring);

{26}let (cK: compKey,tagX: compKey,tx: time) = decrypt(x_127,senkey) in

{27}new snt: time;

{28}if (tx = snt) then

```

```

{29}let tag2: compKey = h(cK) in
{30}if (tagX = tag2) then
{31}event acceptsSensor(cK,tx);
{32}let SenUniqueKey: uniquKey = xor(sensor1LocKey,cK) in
{33}new ty: time;
{34}new r_128: Randomness;
{35}let c': bitstring =
internal_senc((sensorA,ty,sensorData),SenUniqueKey,r_128) in
  {36}out(ch, (sensorA,c'));
  {37}event termSensor(sensorA)
)
– Query inj-event(termSink(y_130)) ==> inj-event(acceptsSink(x_129))
Completing...
Starting query inj-event(termSink(y_130)) ==> inj-event(acceptsSink(x_129))
RESULT inj-event(termSink(y_130)) ==> inj-event(acceptsSink(x_129)) is true.
– Query inj-event(termSensor(z)) ==> inj-event(acceptsSensor(x_696,y_697))
Completing...
Starting query inj-event(termSensor(z)) ==>
inj-event(acceptsSensor(x_696,y_697))
RESULT inj-event(termSensor(z)) ==> inj-event(acceptsSensor(x_696,y_697))
is true.

```

Figure 6.2: Verification result of authentication.

### 6.3 Observational Equivalence

In applied  $\pi$ -calculus terminology, two processes  $p_1$  and  $p_2$  have observational equivalence ( $p_1 \approx p_2$ ) when they are indistinguishable through observation. ProVerif can prove observational equivalence such as strong secrecy, in which an adversary  $\mathcal{A}$  cannot distinguish when a cipher changes. We leverage this feature to prove that the proposed protocol is semantically secure and that the  $\mathcal{A}$  cannot learn anything from the cipher. Figure 6.3 shows the complete verification result. The analysis showses that the sensor data “sensorData” in the proposed protocol are observationally equivalent and that the  $\mathcal{A}$  cannot distinguish when they change because the data are encrypted by a probabilistic algorithm, as described in the post-key distribution phase.

#### Observational Equivalence

Process:

```
{1}new senkey: prkey;  
{2}let sinkey: pukey = pk(senkey) in  
{3}new sensor1LocKey: localK;  
{4}new sensor2LocKey: localK;  
{5}new uK: uniquKey;  
{6}insert sinkt(sensorA,sensor1LocKey);  
{7}insert sinkt(sensorB,sensor2LocKey);  
(  
  {8}!  
  {9}new k_124: compKey;
```

```

{10}let tag: compKey = h(k_124) in

{11}new t: time;

{12}new r_125: Padding;

{13}let c: bitstring = internal_aenc((k_124,tag,t),sinkey,r_125) in

{14}out(ch, c);

{15}in(ch, (senIdX: sensors,y_126: bitstring));

{23}get sinkt(=senIdX,sloc1: localK) in

{16}let SenUniqueKey': uniquKey = xor(sloc1,k_124) in

{17}let (AuthIden: sensors,tz: time,s2: bitstring) =
sdecrypt(y_126,SenUniqueKey') in

{18}if (senIdX = AuthIden) then

{19}event acceptsSink(tz);

{20}new sit: time;

{21}if (tz = sit) then

{22}event termSink(AuthIden)
)|(
{24}!

{25}in(ch, x_127: bitstring);

{26}let (cK: compKey,tagX: compKey,tx: time) = decrypt(x_127,senkey) in

{27}new snt: time;

{28}if (tx = snt) then

{29}let tag2: compKey = h(cK) in

{30}if (tagX = tag2) then

```

```

{31}event acceptsSensor(cK,tx);

{32}let SenUniqueKey: uniquKey = xor(sensor1LockKey,cK) in

{33}new ty: time;

{34}new r_128: Randomness;

{35}let c': bitstring =

internal_senc((sensorA,ty,sensorData),SenUniqueKey,r_128) in

{36}out(ch, (sensorA,c'));

{37}event termSensor(sensorA)

)

– Non-interference sensorData

Completing...

RESULT Non-interference sensorData is true (bad not derivable).

```

*Figure 6.3: Verification result of observational equivalence.*



## CHAPTER 7: CONCLUSION

In this work, we propose a practical key distribution protocol that can be implemented above the IEEE 802.15.4 standard to secure the wireless communication of resource-constrained sensor nodes. We utilized existing cryptographic primitives to design a protocol that maintains a tradeoff between efficiency and security. We conducted simulation, hardware implementations, and modeling to compare the proposed protocol to the existing solutions. Moreover, we conducted formal verifications to prove the soundness and the security of our proposed protocol. The proposed protocol provides low energy and memory consumption, certain key connectivity, and security against: replay attack, man-in-the-middle attack, and node capture attack. The overall results show that the proposed protocol is more efficient and secure than the corresponding schemes.

Future work includes examining more advanced methods to enhance the energy consumption of the proposed protocol. For example, using more efficient algorithms to perform data computation. Furthermore, investigating additional types of attacks against the proposed protocol to increase its security.

## REFERENCES

- [1] C.-Y. Chong and S. P. Kumar, “Sensor networks: evolution, opportunities, and challenges,” *Proceedings of the IEEE*, vol. 91, no. 8, pp. 1247–1256, 2003.
- [2] G. J. Pottie and W. J. Kaiser, “Wireless integrated network sensors,” *Communications of the ACM*, vol. 43, no. 5, pp. 51–58, 2000.
- [3] K. Sohraby, D. Minoli, and T. Znati, *Wireless sensor networks: technology, protocols, and applications*. John Wiley & Sons, 2007.
- [4] S. Zeadally and N. Jabeur, *Cyber-physical System Design with Sensor Networking Technologies*. Institution of Engineering and Technology, 2016.
- [5] F.-J. Wu, Y.-F. Kao, and Y.-C. Tseng, “From wireless sensor networks towards cyber physical systems,” *Pervasive and Mobile computing*, vol. 7, no. 4, pp. 397–413, 2011.
- [6] P. K. Misra, L. Mottola, S. Raza, S. Duquennoy, N. Tsiftes, J. Hoglund, and T. Voigt, “Supporting cyber-physical systems with wireless sensor networks: An outlook of software and services,” *Journal of the Indian Institute of Science*, vol. 93, no. 3, pp. 463–486, 2013.

- [7] C. Lu, A. Saifullah, B. Li, M. Sha, H. Gonzalez, D. Gunatilaka, C. Wu, L. Nie, and Y. Chen, “Real-time wireless sensor-actuator networks for industrial cyber-physical systems,” *Proceedings of the IEEE*, vol. 104, no. 5, pp. 1013–1024, 2016.
- [8] C.-Y. Lin, S. Zeadally, T.-S. Chen, and C.-Y. Chang, “Enabling cyber physical systems with wireless sensor networking technologies,” *International Journal of Distributed Sensor Networks*, vol. 8, no. 5, p. 489794, 2012.
- [9] C. Chen, J. Yan, N. Lu, Y. Wang, X. Yang, and X. Guan, “Ubiquitous monitoring for industrial cyber-physical systems over relay-assisted wireless sensor networks,” *IEEE Transactions on Emerging Topics in Computing*, vol. 3, no. 3, pp. 352–362, 2015.
- [10] W. Abbas, A. Laszka, and X. Koutsoukos, “Resilient wireless sensor networks for cyber-physical systems,” *Cyber-Physical System Design with Sensor Networking Technologies*; Zeadally, S., Jabeur, N., Eds, pp. 239–267, 2016.
- [11] L. Mainetti, L. Patrono, and A. Vilei, “Evolution of wireless sensor networks towards the internet of things: A survey,” in *IEEE 19th International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*. IEEE. Croatia, 2011, pp. 1–6.
- [12] M. T. Lazarescu, “Design of a wsn platform for long-term environmental monitoring for iot applications,” *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 3, no. 1, pp. 45–54, 2013.

- [13] M. Kocakulak and I. Butun, “An overview of wireless sensor networks towards internet of things,” in *IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE. Las Vegas, NV, USA, 2017, pp. 1–6.
- [14] A. Flammini and E. Sisinni, “Wireless sensor networking in the internet of things and cloud computing era,” *Procedia Engineering*, vol. 87, pp. 672–679, 2014.
- [15] L. Wan, G. Han, L. Shu, N. Feng, C. Zhu, and J. Lloret, “Distributed parameter estimation for mobile wireless sensor network based on cloud computing in battlefield surveillance system,” *IEEE Access*, vol. 3, pp. 1729–1739, 2015.
- [16] C. A. Trasviña-Moreno, R. Blasco, Á. Marco, R. Casas, and A. Trasviña-Castro, “Unmanned aerial vehicle based wireless sensor network for marine-coastal environment monitoring,” *Sensors*, vol. 17, no. 3, p. 460, 2017.
- [17] A. B. Noel, A. Abdaoui, T. Elfouly, M. H. Ahmed, A. Badawy, and M. S. Shehata, “Structural health monitoring using wireless sensor networks: A comprehensive survey,” *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1403–1423, 2017.
- [18] J.-R. Lin, T. Talty, and O. K. Tonguz, “A blind zone alert system based on intravehicular wireless sensor networks,” *IEEE Transactions on Industrial Informatics*, vol. 11, no. 2, pp. 476–484, 2015.
- [19] T. Liang and Y. J. Yuan, “Wearable medical monitoring systems based on wireless networks: A review,” *IEEE Sensors Journal*, vol. 16, no. 23, pp. 8186–8199, 2016.

- [20] Z. Iqbal, K. Kim, and H.-N. Lee, "A cooperative wireless sensor network for indoor industrial monitoring." *IEEE Trans. Industrial Informatics*, vol. 13, no. 2, pp. 482–491, 2017.
- [21] V. Bapat, P. Kale, V. Shinde, N. Deshpande, and A. Shaligram, "Wsn application for crop protection to divert animal intrusions in the agricultural land," *Computers and Electronics in Agriculture*, vol. 133, pp. 88–96, 2017.
- [22] J. Aponte-Luis, J. A. Gómez-Galán, F. Gómez-Bravo, M. Sánchez-Raya, J. Alcina-Espigado, and P. M. Teixido-Rovira, "An efficient wireless sensor network for industrial monitoring and control," *Sensors*, vol. 18, no. 1, p. 182, 2018.
- [23] D. Antolín, N. Medrano, B. Calvo, and F. Pérez, "A wearable wireless sensor network for indoor smart environment monitoring in safety applications," *Sensors*, vol. 17, no. 2, p. 365, 2017.
- [24] K. S. Adu-Manu, C. Tapparello, W. Heinzelman, F. A. Katsriku, and J.-D. Abdulai, "Water quality monitoring using wireless sensor networks: Current trends and future research directions," *ACM Transactions on Sensor Networks (TOSN)*, vol. 13, no. 1, p. 4, 2017.
- [25] E. Aguirre, P. Lopez-Iturri, L. Azpilicueta, A. Redondo, J. J. Astrain, J. Villadangos, A. Bahillo, A. Perallos, and F. Falcone, "Design and implementation of context aware applications with wireless sensor network support in urban train transportation environments," *IEEE Sensors Journal*, vol. 17, no. 1, pp. 169–178, Jan 2017.

- [26] I. . W. Group *et al.*, “Ieee standard for local and metropolitan area networks—part 15.4: Low-rate wireless personal area networks (lr-wpans),” *IEEE Std*, vol. 802, pp. 4–2011, 2011.
- [27] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, “Wireless sensor networks: a survey,” *Computer networks*, vol. 38, no. 4, pp. 393–422, 2002.
- [28] J. Zhang and V. Varadharajan, “Wireless sensor network key management survey and taxonomy,” *Journal of Network and Computer Applications*, vol. 33, no. 2, pp. 63–75, 2010.
- [29] K.-A. Shim, “A survey of public-key cryptographic primitives in wireless sensor networks,” *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 577–601, 2016.
- [30] P. Mahajan and A. Sardana, “Key distribution schemes in wireless sensor networks: Novel classification and analysis,” in *Advances in Computing and Information Technology*. Springer. Berlin-Heidelberg, Germany, 2012, pp. 43–53.
- [31] S. Bala, G. Sharma, and A. K. Verma, “A survey and taxonomy of symmetric key management schemes for wireless sensor networks,” in *Proceedings of the CUBE International Information Technology Conference*. ACM. Pune, India, 2012, pp. 585–592.
- [32] Y. Liao, C. Lei, and A. Wang, “A robust grid-based key predistribution scheme for sensor networks,” in *2009 Fourth International Conference on Innovative Computing, Information and Control (ICICIC)*. IEEE. Kaohsiung, Taiwan, 2009, pp. 760–763.

- [33] N. X. Quy, V. Kumar, Y. Park, E. Choi, and D. Min, "A high connectivity pre-distribution key management scheme in grid-based wireless sensor networks," in *2008 International Conference on Convergence and Hybrid Information Technology*. IEEE. Busan, South Korea, 2008, pp. 35–42.
- [34] N.-C. Wang, Y.-L. Chen, and H.-L. Chen, "An efficient grid-based pairwise key predistribution scheme for wireless sensor networks," *Wireless personal communications*, vol. 78, no. 2, pp. 801–816, 2014.
- [35] H. Chan and A. Perrig, "Pike: peer intermediaries for key establishment in sensor networks," in *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, vol. 1. IEEE, Miami, FL, USA, 2005, pp. 524–535.
- [36] B. Schneier, *Applied cryptography: protocols, algorithms, and source code in C*. john wiley & sons, 2007.
- [37] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, Nov. 1979. [Online]. Available: <http://doi.acm.org/10.1145/359168.359176>
- [38] H. Tong-sen, C. Deng, and T. Xian-zhong, "An enhanced polynomial-based key establishment scheme for wireless sensor networks," in *2008 International Workshop on Education Technology and Training 2008 International & 2008 International Workshop on Geoscience and Remote Sensing*. IEEE. Shanghai, China, 2008, pp. 809–812, vol. 2.
- [39] X. Li and J. Shen, "A novel key pre-distribution scheme using one-way hash chain and bivariate polynomial for wireless sensor networks," in *2009 3rd International*

*Conference on Anti-counterfeiting, Security, and Identification in Communication.*  
IEEE. Hong Kong, China, 2009, pp. 575–580.

- [40] H. Ito, A. Miyaji, and K. Omote, “Rpok: A strongly resilient polynomial-based random key pre-distribution scheme for multiphase wireless sensor networks,” in *2010 IEEE Global Telecommunications Conference GLOBECOM 2010*. IEEE. Miami, FL, USA, 2010, pp. 1–5.
- [41] H. Dai and H. Xu, “An improved polynomial-based key predistribution scheme for wireless sensor networks,” in *2008 International Conference on Communications, Circuits and Systems*. IEEE. Fujian, China, 2008, pp. 424–428.
- [42] F. Delgosha, E. Ayday, and F. Fekri, “Mkps: A multivariate polynomial scheme for symmetric key-establishment in distributed sensor networks,” in *Proceedings of the 2007 International Conference on Wireless Communications and Mobile Computing*, ser. IWCMC '07. ACM. Honolulu, Hawaii, USA, 2007, pp. 236–241.
- [43] A. K. Das and I. Sengupta, “An effective group-based key establishment scheme for large-scale wireless sensor networks using bivariate polynomials,” in *2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE '08)*. IEEE. Bangalore, India, 2008, pp. 9–16.
- [44] E. Baburaj *et al.*, “Polynomial and multivariate mapping-based triple-key approach for secure key distribution in wireless sensor networks,” *Computers & Electrical Engineering*, vol. 59, pp. 274–290, 2017.
- [45] L. Eschenauer and V. D. Gligor, “A key-management scheme for distributed sensor networks,” in *Proceedings of the 9th ACM Conference on Computer and Com-*



- munications Security*, ser. CCS '02. ACM. Washington, DC, USA, 2002, pp. 41–47.
- [46] L. Morales, I. H. Sudborough, M. Eltoweissy, and M. H. Heydari, “Combinatorial optimization of multicast key management,” in *Proceedings of the 36th Annual Hawaii International Conference on System Sciences*. IEEE. Hawaii, USA, Jan 2003, pp. 324 – 332.
- [47] M. F. Younis, K. Ghumman, and M. Eltoweissy, “Location-aware combinatorial key management scheme for clustered sensor networks,” *IEEE transactions on parallel and distributed systems*, vol. 17, no. 8, pp. 865–882, 2006.
- [48] B. Ying, D. Makrakis, H. T. Mouftah, and W. Lu, “Dynamic combinatorial key pre-distribution scheme for heterogeneous sensor networks,” in *FTRA International Conference on Secure and Trust Computing, Data Management, and Application*. Springer. Loutraki, Greece, 2011, pp. 88–95.
- [49] J. Wang, L. Zheng, L. Zhao, and D. Tian, “Leach-based security routing protocol for wsns,” in *Advances in Computer Science and Information Engineering*. Springer. Zhengzhou, China, 2012, pp. 253–258.
- [50] M. K.-u.-R. R. Syed, H. Lee, S. Lee, and Y.-K. Lee, “Muqami+: a scalable and locally distributed key management scheme for clustered sensor networks,” *annals of telecommunications - annales des télécommunications*, vol. 65, no. 1, pp. 101–116, Feb 2010.

- [51] M. Moharrum, M. Eltoweissy, and R. Mukkamala, “Dynamic combinatorial key management scheme for sensor networks,” *Wireless Communications and Mobile Computing*, vol. 6, no. 7, pp. 1017–1035, 2006.
- [52] M. A. Moharrum and M. Eltoweissy, “A study of static versus dynamic keying schemes in sensor networks,” in *Proceedings of the 2Nd ACM International Workshop on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks*. ACM. Montreal, Quebec, Canada, 2005, pp. 122–129.
- [53] C.-C. Lo, C.-C. Huang, and S.-W. Chen, “An efficient and scalable ebs-based batch rekeying scheme for secure group communications,” in *Military Communications Conference*. IEEE. Boston, Massachusetts, USA, Oct 2009, pp. 1–7.
- [54] F. Zhan, N. Yao, Z. Gao, and G. Tan, “A novel key generation method for wireless sensor networks based on system of equations,” *Journal of Network and Computer Applications*, vol. 82, pp. 114–127, 2017.
- [55] Z. Yu, “The scheme of public key infrastructure for improving wireless sensor networks security,” in *2012 IEEE International Conference on Computer Science and Automation Engineering*. IEEE. Zhangjiajie, China, 2012, pp. 527–530.
- [56] A. Chung and U. Roedig, “Efficient key establishment for wireless sensor networks using elliptic curve diffie-hellman,” in *Proceedings of the 2nd European Conference on Smart Sensing and Context (EUROSSC2007)*. Kendal, UK, 2007.
- [57] N. Nagy, M. Nagy, and S. G. Akl, “Quantum wireless sensor networks,” in *The 7th International Conference on Unconventional Computing*. Springer. Vienna, Austria, 2008, pp. 177–188.

- [58] J. Li and C. Yang, “Quantum communication in distributed wireless sensor networks,” in *2009 IEEE 6th International Conference on Mobile Adhoc and Sensor Systems*. IEEE, Macau, China, 2009, pp. 1024–1029.
- [59] Sheet, X.P.D, “Xbee-datasheet,” available online: <https://www.sparkfun.com/datasheets/Wireless/Zigbee/XBee-Datasheet.pdf> (accessed on 3 May 2018).
- [60] Atmel, “Atmega328p datasheet,” available online: [http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A-48PA-88A-88PA-168A-168PA-328-328P\\_datasheet\\_Summary.pdf](http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A-48PA-88A-88PA-168A-168PA-328-328P_datasheet_Summary.pdf) (accessed on 5 June 2018).