sensors

MDPI

Article

Efficient and Secure Key Distribution Protocol for Wireless Sensor Networks

Majid R. Alshammari * and Khaled M. Elleithy

Department of Computer Science and Engineering, University of Bridgeport, 126 Park Ave, Bridgeport, CT 06604, USA; elleithy@bridgeport.edu

* Correspondence: maalsham@my.bridgeport.edu; Tel.: +1-203-889-6542

Received: 1 September 2018; Accepted: 16 October 2018; Published: 21 October 2018

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 the transmitted data. However, the standard does not specify how to manage, store, or distribute the 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.

Keywords: key distribution; wireless sensor networks; resource-constrained nodes

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. Due to the low cost and flexibility of WSNs, they are employed in a variety of applications such as agriculture, the environment, health, home and commercial automation, the military and transportation [1–11] and have recently become a promising technology for Internet of Things (IoT) applications [12–15]. 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 standarad also determines the radio frequency used for communication and provides four security services: access control, confidentially, integrity 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 [16]. 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.

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. This work extends our preliminary work introduced in [17] by improving its efficiency and security. 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 [18], 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 Appendix A.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 Appendix A.1) 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.

The remainder of this paper is organized as follows: The next section introduces a classification for WSN key distribution schemes and work related to the proposed protocol. Section 3 describes the proposed protocol. Section 4 presents our findings and analyses. Section 5 describes the formal verification of our proposed protocol, and Section 6 concludes the paper.

2. Related Work

Key distribution schemes in WSNs have been comprehensively studied in the literature. The authors of [19–22] 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 and probabilistic 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 1 depicts this classification hierarchy, and Table 1 defines our evaluation metrics.

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 [23–25]. An early example was called Peer Intermediaries for Key Establishment in Sensor Networks (PIKE) [26]. 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:

$$(i, y) \; \forall \; i \in \{1, 2, 3, ..., \sqrt{n-1}\} \tag{1}$$

$$(x, j) \; \forall \; i \in \{1, 2, 3, ..., \sqrt{n-1}\}. \tag{2}$$

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 $\left(\sqrt{n-1}\right)$ keys.

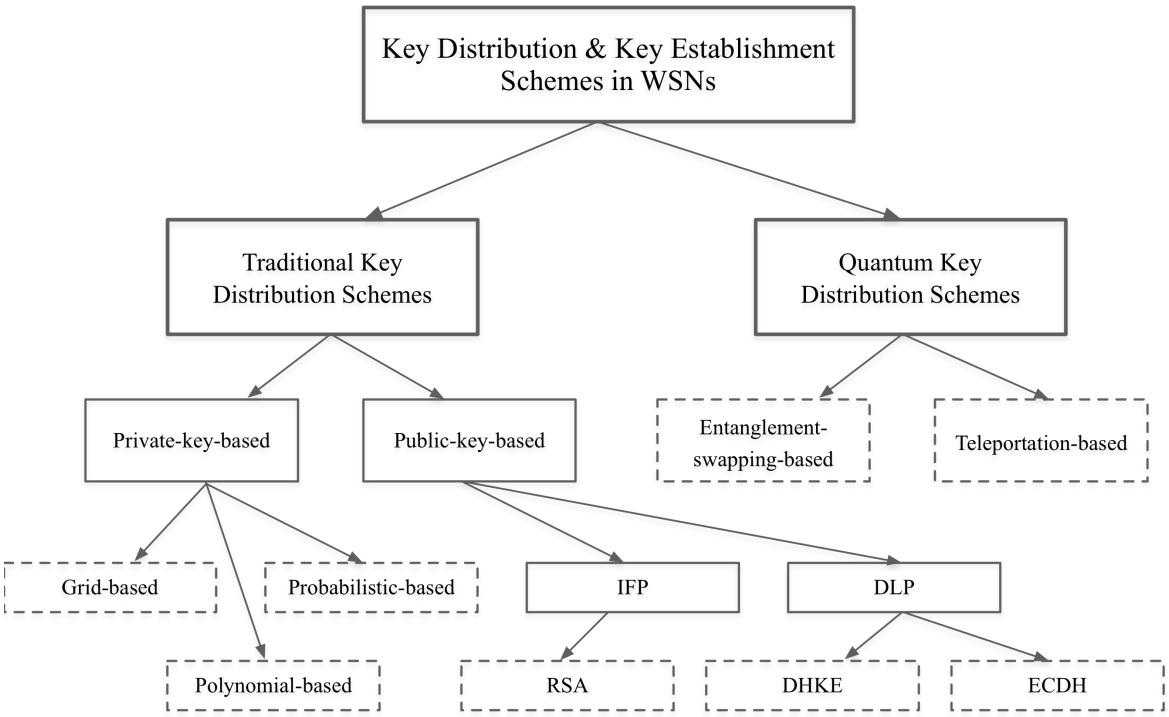

Figure 1. Classification of key distribution schemes in WSNs.

Table 1. Evaluation Metrics.

Metric		Definition
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.

Polynomial-based schemes depend on storing polynomials on wireless sensor nodes that are used for key generation. In [27], 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 [28], 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:

$$\left(ax^{R-1} + bx^{(R-1)-1} + ... + m\right) mod\ p, \tag{3}$$

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 [29–34]. In [35], 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 [36], 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:

$$P_{key} = 1 - \left(\frac{((P-k)!)^2}{((P-2k)!\,P!)}\right), \tag{4}$$

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, $li = \{\alpha||Eki(\alpha)||i = 1, ..., k\}$, for every key on the key ring, where α is a challenge. The ability of the receiver to decrypt $Eki(\alpha)$ will reveal the challenge α 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.

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 [37] 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 [38], 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, α_n is generated to calculate $P_n = \alpha_n G$ for all n nodes. Next, α_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 α and b values that defined the curve and the generator point G.

Quantum-based schemes rely on the laws of physics and require special hardware. Nevertheless, many authors have proposed solutions. For example, entanglement swapping was adopted in [39]. 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 [40] 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.

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 2 presents the notations used to describe the proposed protocol.

Table 2. Notation for the Proposed Protocol.

Notation	Description
$y \longleftarrow x$	y is generated by x.
$x \stackrel{def}{=} y$	x is defined as y.
$x := y$	y is assigned to x.
$H(\,)$	One-way hash function.
$\|\|$	Concatenation.
$\stackrel{send}{\Longrightarrow} [x]$	Sending message x.
$\stackrel{recv}{\Longleftarrow} [x]$	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

Pre-deployment Phase Steps

$\{K_P, K_R\} \longleftarrow RSA_{gen}$
$K_P \overset{def}{=} AK_{sink}$ and $K_R \overset{def}{=} AK_{nodes}$
Sink node := AK_{sink} and *Sensor nodes* := AK_{nodes}
$K_{local_i} \overset{R}{\longleftarrow} key_{gen}\{0,1\}^{128}, \forall K_{local_i} \in \{0,1\}^{128} \Rightarrow P\left(K_{local_i}\right) = \frac{1}{|\{0,1\}^{128}|}$
Sensor node$_i$:= K_{local_i} and *Sink node* := K_{local_i}

The pre-deployment phase of the protocol consists of five offline steps. The first step is the generation of an asymmetric key pair $\{K_P, K_R\} \longleftarrow RSA_{gen}$, 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:$K_{local_i} \overset{R}{\longleftarrow} key_{gen}\{0,1\}^{128}$, 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 $P\left(K_{local_i}\right) = \frac{1}{|\{0,1\}^{128}|}$. In the fifth step, K_{local_i} is loaded into the corresponding sensor node$_i$ and into the sink nodes.

3.2. Key Distribution Phase

Key Distribution Phase Steps

Sink node:
$K_{compl} \longleftarrow Ran_{gen}\{0,1\}^{128}$, $Tag \longleftarrow H\left(K_{compl}\right)$, and *Timestamp* T
$C \longleftarrow E_{AK_{sink}}\left(K_{compl} \,||\, Tag \longleftarrow H(K_{compl}) \,||\, T\right)$
$\overset{send}{\Longrightarrow} \left[C \longleftarrow E_{AK_{sink}}(K_{compl} \,||\, Tag \longleftarrow H(K_{compl}) \,||\, T)\right]$
Sensor nodes:
$\overset{recv}{\Longleftarrow} \left[C \longleftarrow E_{AK_{sink}}\left(K_{compl} \,||\, Tag \longleftarrow H(K_{compl}) \,||\, T\right)\right]$
$P \longleftarrow D_{AK_{nodes}}\left(C \longleftarrow E_{AK_{sink}}\left(K_{compl} \,||\, Tag \longleftarrow H(K_{compl}) \,||\, T\right)\right)$
$f_{verify}(T) = \begin{cases} accept, & if\ T \leq time\ threhsold \\ reject, & if\ T > time\ threhsold \end{cases}$
$Tag' \longleftarrow H\left(K_{compl}\right)$
$f_{compare}(Tag, Tag') = \begin{cases} accept, & if\ match \\ reject, & if\ dismatch \end{cases}$
$K_{unique_i} = K_{compl} \oplus K_{local_i}$

After deploying the sensor nodes, the sink node generates a random complementary key $K_{compl} \longleftarrow Ran_{gen}\{0,1\}^{128}$, computes its hash value $Tag \longleftarrow H(K_{compl})$, 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: $\overset{send}{\Longrightarrow} \left[C \longleftarrow E_{AK_{sink}}(K_{compl} \,||\, Tag \longleftarrow H(K_{compl}) \,||\, T)\right]$. These neighbors forward the cipher to their neighbors in a multihop fashion until all of the sensor nodes have received the cipher $\overset{recv}{\Longleftarrow} \left[C \longleftarrow\right.$

$E_{AK_{sink}}\left(K_{compl} \,||\, Tag \longleftarrow H(K_{compl}) \,||\, T\right)\Big]$. Because each sensor node is loaded with the asymmetric key AK_{nodes} in the pre-deployment phase, a sensor node$_i$ can decrypt the cipher as follows: $P \longleftarrow D_{AK_{nodes}}\left(C \longleftarrow E_{AK_{sink}}\left(K_{compl} \,||\, Tag \longleftarrow H(K_{compl}) \,||\, T\right)\right)$ and verifies the timestamp $f_{verify}(T)$ based on a predefined threshold. If the timestamp exceeds the threshold, the sensor node$_i$ rejects the cipher. Otherwise, the sensor node$_i$ hashes the complementary key $Tag' \longleftarrow H\left(K_{compl}\right)$ 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$_i$ rejects the cipher; otherwise, the sensor node$_i$ produces its unique key by XORing the complementary key and its local key as follows: $K_{unique_i} = K_{compl} \oplus K_{local_i}$.

3.3. Post-Key Distribution Phase

Post-key Distribution Phase Steps

Sensor nodes:

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

$C \longleftarrow E^{\perp}_{K_{unique_i}}(D \,||\, Id_i \,||\, T)$

$\overset{send}{\Longrightarrow}\left[Id_i' \,||\, C \longleftarrow E^{\perp}_{K_{unique_i}}(D \,||\, Id_i \,||\, T)\right]$

Sink node:

$\overset{recv}{\Longleftarrow}\left[Id_i' \,||\, C \longleftarrow E^{\perp}_{K_{unique_i}}(D \,||\, Id_i \,||\, T)\right]$

$F : Id_i' \mapsto K_{unique_i}$

$P \longleftarrow D^{\perp}_{K_{unique_i}}\left(C \longleftarrow E^{\perp}_{K_{unique_i}}(D \,||\, Id_i \,||\, T)\right)$

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

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

After establishing the key distribution phase, the sensor nodes have already produced their unique keys. Therefore, when sensor node$_i$ 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: $\overset{send}{\Longrightarrow}\left[Id_i' \,||\, C \longleftarrow E^{\perp}_{K_{unique_i}}(D \,||\, Id_i \,||\, T)\right]$, where $\perp$ is a probabilistic encryption algorithm. Because this study is concerned with key distribution, the sensor node$_i$ can use any secure probabilistic encryption algorithm. When the sink node receives the following cipher $\overset{recv}{\Longleftarrow}\left[Id_i' \,||\, C \longleftarrow E^{\perp}_{K_{unique_i}}(D \,||\, Id_i \,||\, T)\right]$, 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.

3.4. Key Refreshment Phase

Key Refreshment Phase Steps

Sink node:

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

$C \longleftarrow E_{AK_{sink}}\left(K_{compl_{new}} \,||\, Tag \longleftarrow H(K_{compl_{new}}) \,||\, T\right)$

$\overset{send}{\Longrightarrow}\left[C \longleftarrow E_{AK_{sink}}(K_{compl_{new}} \,||\, Tag \longleftarrow H(K_{compl_{new}}) \,||\, T)\right]$

Sensor nodes:

$\overset{recv}{\Longleftarrow}\left[C \longleftarrow E_{AK_{sink}}\left(K_{compl_{new}} \,||\, Tag \longleftarrow H(K_{compl_{new}}) \,||\, T\right)\right]$

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

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

$Tag' \longleftarrow H\left(K_{compl_{new}}\right)$

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

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

In the key refreshment phase, the sink node generates a new random complementary key $K_{compl_{new}} \longleftarrow Ran_{gen}\{0,1\}^{128}$, computes its hash value $Tag \longleftarrow H(K_{compl_{new}})$, 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: $\overset{send}{\Longrightarrow}\left[C \longleftarrow E_{AK_{sink}}(K_{compl_{new}} \,||\, Tag \longleftarrow H(K_{compl_{new}}) \,||\, T)\right]$. These neighbors forward the cipher to their neighbors in a multihop fashion until all of the sensor nodes have received the cipher $\overset{recv}{\Longleftarrow}\left[C \longleftarrow E_{AK_{sink}}\left(K_{compl_{new}} \,||\, Tag \longleftarrow H(K_{compl_{new}}) \,||\, T\right)\right]$. Because the sensor nodes are loaded with the asymmetric key AK_{nodes} in the pre-deployment phase, a sensor node$_i$ can decrypt the cipher as follows: $P \longleftarrow D_{AK_{nodes}}\left(C \longleftarrow E_{AK_{sink}}\left(K_{compl_{new}} \,||\, Tag \longleftarrow H(K_{compl_{new}}) \,||\, T\right)\right)$ and verifies the timestamp $f_{verify}(T)$, based on a predefined threshold. If T exceeds the threshold, the sensor node$_i$ rejects the cipher; otherwise, the node hashes the new complementary key $Tag' \longleftarrow H\left(K_{compl_{new}}\right)$ 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$_i$ rejects the cipher; otherwise, it produces its new unique key by XORing the new complementary key and its local key as follows: $K_{unique_{i_{new}}} = K_{compl_{new}} \oplus K_{local_i}$.

4. Findings and Analyses

In this section, we analyze the efficiency and security of the proposed protocol in comparison to the schemes proposed in [26,35–37]. These analyses are based on the metrics presented in Table 1.

4.1. Efficiency Analysis

4.1.1. Energy Consumption

Table 3 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 Appendix A.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.P.E" 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.

As shown in Table 3, the nodes' transceivers consume the lowest amount of energy in the proposed protocol, followed by scheme [37], at 2.75 mJ and 4.15 mJ, respectively; whereas, the nodes' transceivers consume the largest amount of energy in scheme [35], followed by scheme [36] and scheme [26], at 295.77 mJ, 150.37 mJ, and 46.02 mJ, respectively.

In contrast, the nodes' microcontrollers consume the lowest amount of energy in scheme [26], followed by scheme [36] and scheme [35], at 0.15 mJ, 1.29 mJ, and 27.50 mJ, respectively, and the largest amount of energy consumed is in the proposed protocol, followed by scheme [37], at 38.35 mJ and 38.23 mJ, respectively.

However, the total energy consumption in the proposed protocol is 41.10 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.

Table 3. Energy consumption of each key distribution scheme.

Descriptions	Schemes / Parameters	Our Protocol	Scheme [26]	Scheme [35]	Scheme [36]	Scheme [37]
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	0.02 mJ
	E.TRX	2.75 mJ	46.02 mJ	295.77 mJ	150.37 mJ	4.15 mJ
The parameters that contribute to the energy consumption of nodes' microcontroller.	**C.S.K**	$\mathcal{O}(1)$	$\mathcal{O}(2 \cdot \sqrt{n-1})$	$\mathcal{O}(1)$	$\mathcal{O}\left(\sqrt{-P \cdot log(1-P_c)}\right)$ [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.P.E	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	**T.E.C**	41.10 mJ	46.17 + $2t^2$ mJ	323.28 mJ	151.65 mJ	42.39 mJ

[1] The key pool size P is equal to the number of sensor nodes multiplied by 10, and P_c is equal to 0.99%. [2] 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 does not specify the type of encryption and decryption algorithm used.

4.1.2. Key Storage Overhead

Table 4 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 2 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.

Table 4. Key Storage overhead for each scheme.

Scheme	Key Storage Overhead
The proposed protocol	3
Scheme [26]	$2 \cdot \sqrt{n-1}$
Scheme [35]	6
Scheme [36]	$\sqrt{-P \cdot log(1-P_c)}$ [1]
Scheme [37]	n

n represents the number of nodes. [1] The key pool size p is equal to the number of sensor nodes multiplied by 10, and P_c is equal to 0.99%.

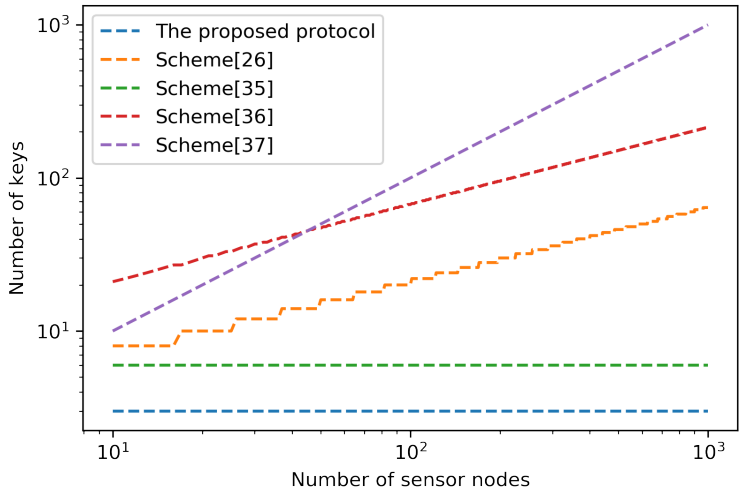

Figure 2. The magnitude of key storage overhead.

4.1.3. 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 3a shows a random deployment of 200 nodes over an area of 1000 ft · 1000 ft. In Figure 3b, 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 Appendix A.1).

Figure 4 shows the implementation of the proposed protocol and the corresponding schemes on the WSN shown in Figure 3b. 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:

$$The\ key\ connectivity\ (\%) = \frac{Secured\ links}{Total\ number\ of\ links} \times 100, \quad (5)$$

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 4a,c,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 Figure 4b,d reach only 87% and 99%, respectively.

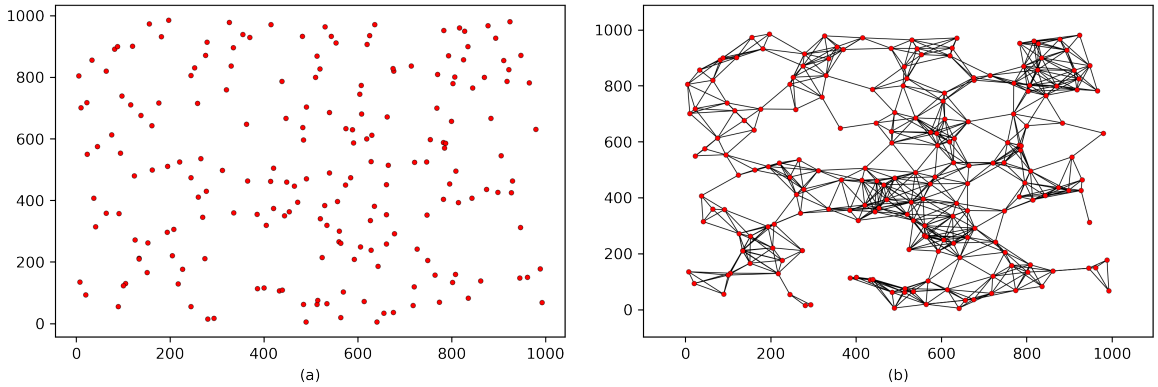

Figure 3. Modeling a WSN: (**a**) random deployment of sensor nodes; (**b**) wireless signal range of nodes' transceivers.

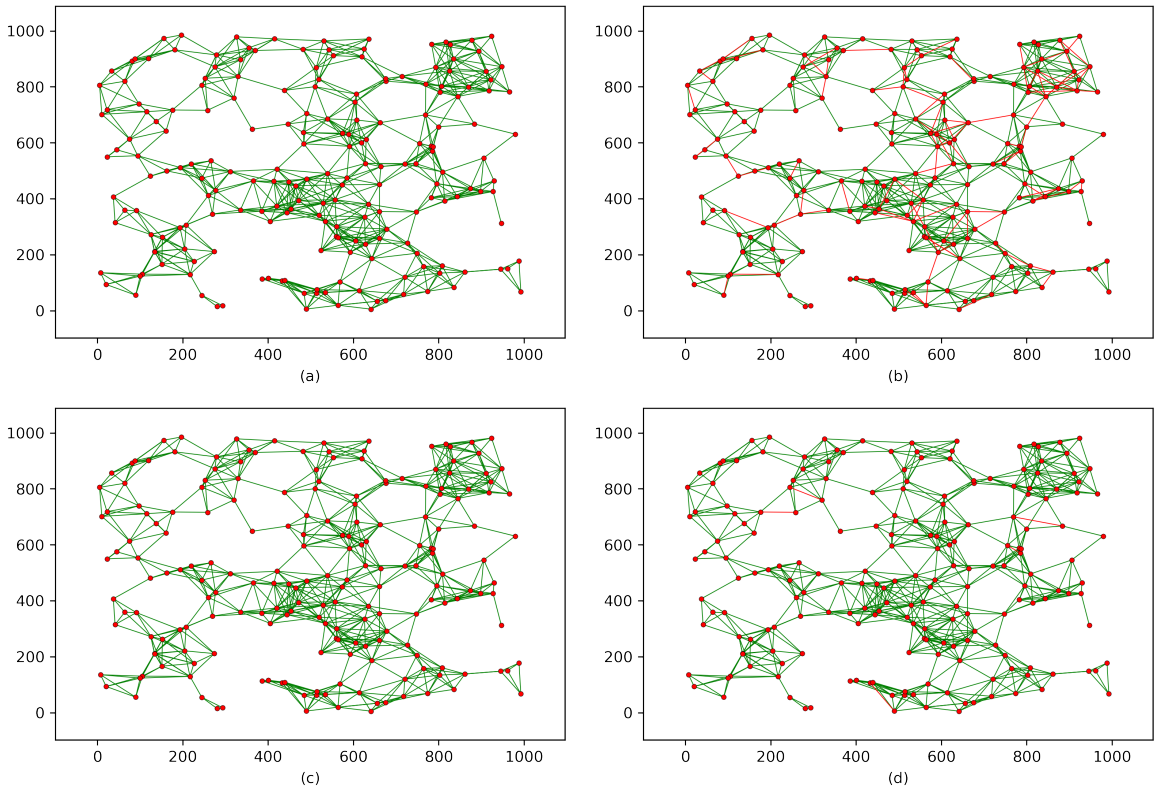

Figure 4. *Cont.*

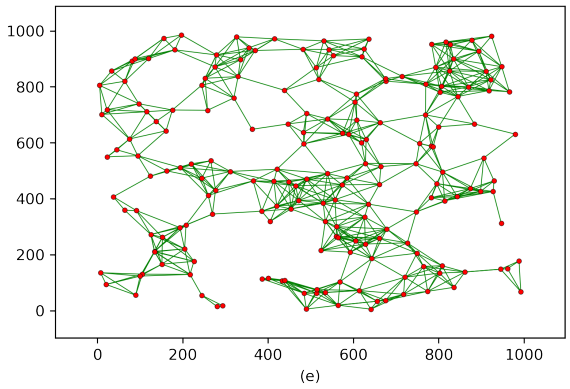

Figure 4. Key connectivity after implementing key distribution/establishment process for each scheme. (**a**) the proposed protocol; (**b**) Scheme [26]; (**c**) Scheme [35]; (**d**) Scheme [36] ; (**e**) scheme [37].

4.2. Security Analysis

4.2.1. 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:

$$\overset{send}{\Longrightarrow} \Big[C \longleftarrow E_{AK_{sink}}(K_{compl} \,||\, Tag \longleftarrow H(K_{compl}) \,||\, T)\Big]$$
$$\overset{send}{\Longrightarrow} \Big[C \longleftarrow E_{AK_{sink}}(K_{compl_{new}} \,||\, Tag \longleftarrow H(K_{compl_{new}}) \,||\, T)\Big].$$

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

$$\overset{send}{\Longrightarrow} \Big[Id_i' \,||\, C \longleftarrow E^{\perp}_{K_{unique_i}}(D \,||\, Id_i \,||\, T)\Big].$$

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.

4.2.2. 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.

$$\{K_P, K_R\} \longleftarrow RSA_{gen}$$
$$K_P \overset{def}{=} AK_{sink} \text{ and } K_R \overset{def}{=} AK_{nodes}.$$

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

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

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.

$$C \longleftarrow E^{\perp}_{K_{unique_i}}(D \,||\, Id_i \,||\, T).$$

Therefore, an adversary cannot impersonate either the sink node or any sensor node in the proposed protocol. Scheme [37] is vulnerable to man-in-the-middle attacks; however, the other compared schemes are not subjected to this type of attack.

4.2.3. 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, we mount node capture attacks on the key connectivity of each scheme presented in Figure 4. Thus, Figure 5 shows each scheme's resilience against node capture attacks. The impact of a node capture attack can be defined as follows:

$$\textit{The impact of a node capture attack on a WSN}\,(\%) = \frac{\textit{Compromised links}}{\textit{Total number of secured links}} \times 100, \quad (6)$$

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:

1. An adversary is able to physically capture 5% of the sensor nodes randomly (i.e., 10 sensor nodes in the example network).
2. Because capturing the sink node will compromise any given WSN, in Assumption 1, node capture does not include the sink node.
3. 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 5a, 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 Figure 5b,c are decreased by 51% and 75%, respectively (assuming the network in Figure 5c has 4 clusters and that some of the compromised

nodes are located in 3 different clusters). The key connectivity in Figure 5d is only slightly affected, decreasing by only 6%. In contrast, the network in Figure 5e is heavily impacted, and the key connectivity is decreased by 100%.

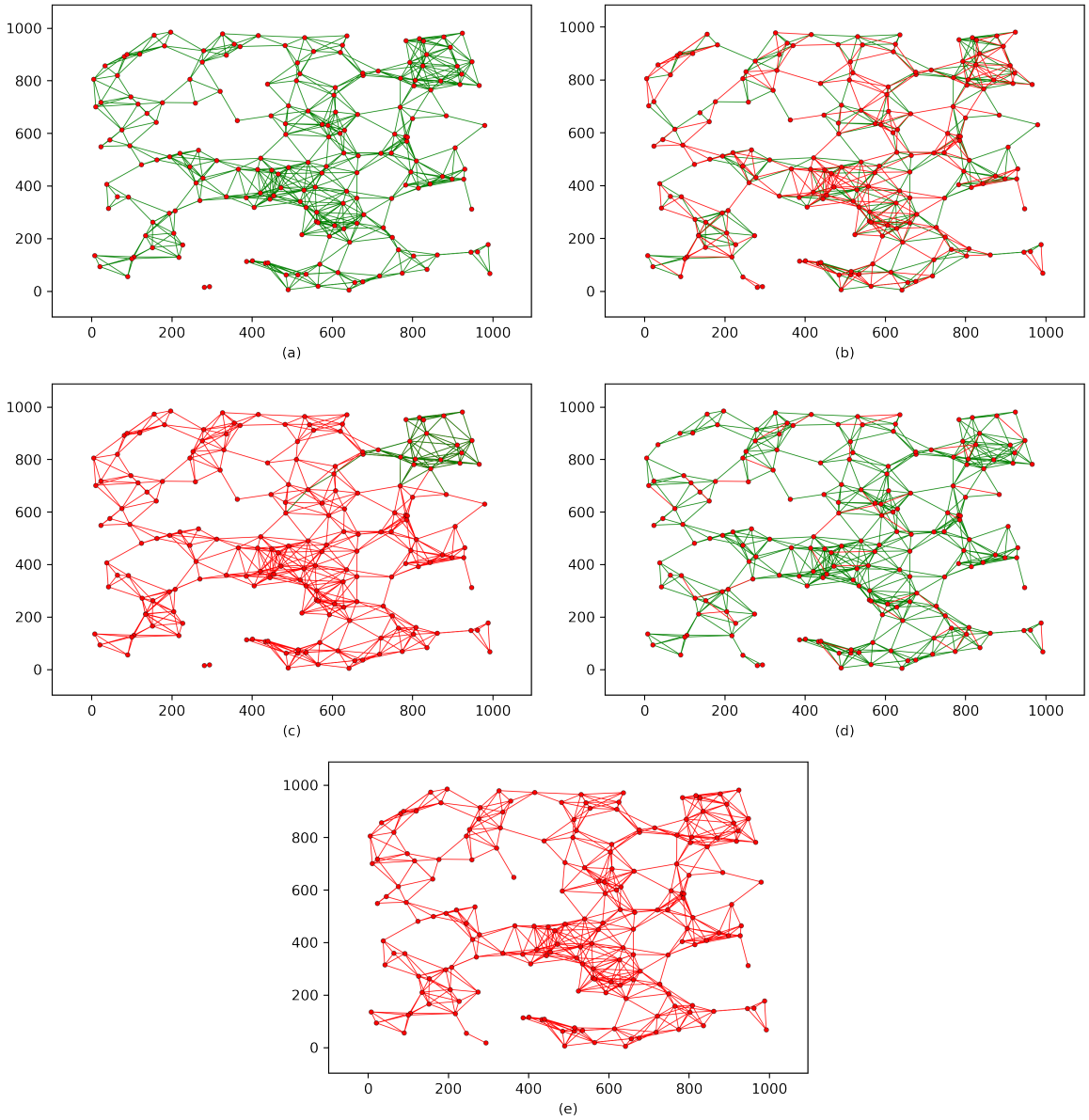

Figure 5. Schemes' resilience against node capture attacks. (**a**) The proposed protocol; (**b**) Scheme [26]; (**c**) Scheme [35]; (**d**) Scheme [36] ; (**e**) Scheme [37].

5. 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.

5.1. 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 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. Verification result of reachability and secrecy.

5.2. 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 7 shows the complete verification result. The verification confirmes 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 7. Verification result of authentication.

5.3. Observational Equivalence

In applied π-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 8 shows the complete verification result. The analysis showes 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,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)
)

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

Figure 8. Verification result of observational equivalence.

6. Conclusions

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.

Author Contributions: The work has been primarily conducted by M.R.A. under the supervision of K.M.E.

Funding: This research was funded by the University of Bridgeport for financial funding to publish this research.

Acknowledgments: The authors acknowledge the reviewers for their valuable comments that have improved this paper to appear in its current form.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Methodology

Appendix A.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 A1. 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 Appendix A.2 and Appendix A.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 Appendix A.2.

Moreover, in this experiment, the transceiver parameters of our sensor node model are based on XBee transceiver S1 [41], and the microcontroller used in the implementation is an Atmega328p [42] (clocked at 16 MHz). The following table shows these parameters.

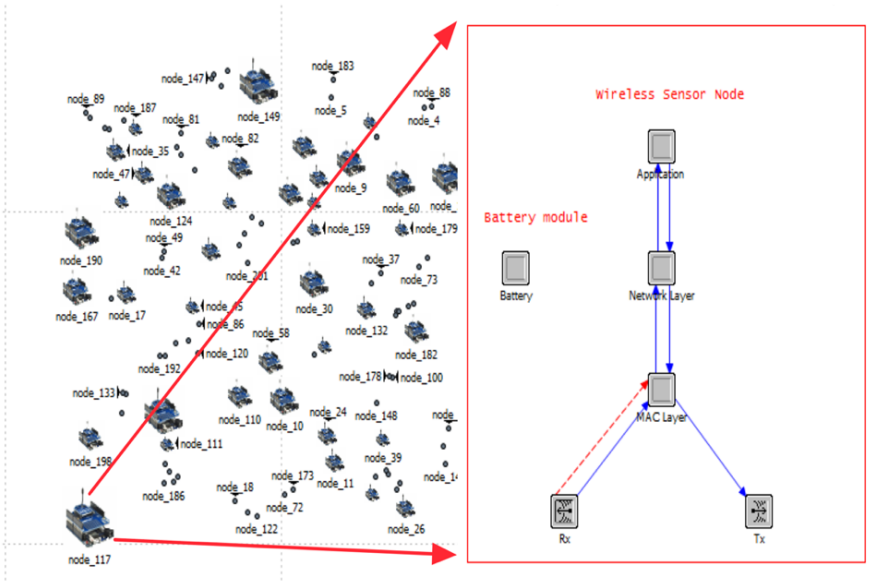

Figure A1. Our wireless sensor node model.

Table A1. Experiment Parameters.

Description	Parameters	Values
Channel	Data Rate	250 kbps
	Frame Size	1024 bits
	Transmission power	0 dBm
	Modulation	bpsk
	Receiver Sensitivity	−92 dBm
Transceiver	Tx Current draw	45 mA @ 3.3 VDC
	Rx Current draw	50 mA @ 3.3 VDC
Microcontroller	Microcontroller	3.2 mA @ 4.5 V
Power Requirements	Tx power consumption	148.5 mW
	Rx power consumption	165 mW
	Microcontroller power consumption	14.4 mW

Appendix A.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 (A1). 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 paper, the energy consumption of a wireless sensor node is calculated by Equation (A2):

$$E_{sensor\ node} = E_{transceiver} + E_{microcontroller} + E_{sensor} \quad \text{(A1)}$$

$$E_{sensor\ node} = E_{transceiver} + E_{microcontroller}. \quad \text{(A2)}$$

The energy consumption is generally calculated by Equation (A3):

$$Energy(J) = Power(W) \cdot Time(S), \quad \text{(A3)}$$

where $Power(W)$ is calculated by Equation (A4):

$$Power(W) = Voltage(V) \cdot Current(A). \tag{A4}$$

Based on Equation (A3), we calculate the energy consumption of nodes' transceivers and microcontrollers. The following subsections describe these calculations.

Appendix A.2.1. Energy Consumption of the Transceiver

The energy consumption of a node's transceiver $E_{transceiver}$, given in Equation (A2), 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 $E_{Tx_{TPO}}$ and the transmitter electronics $E_{Tx_{elec}}$, which can be found by the following equation:

$$E_{Tx} = E_{Tx_{TPO}} + E_{Tx_{elec}}, \tag{A5}$$

where the energy consumed by $E_{Tx_{TPO}}$ is found by Equation (A6):

$$E_{Tx_{TPO}} = Tx_{power_{output}} \cdot Tx_{time}, \tag{A6}$$

where $Tx_{power_{output}}$ is the TPO, and it depends on the transceiver module (described in Appendix A.1). Tx_{time} is the time that a node's transmitter takes to send one frame and is found by Equation (A7):

$$Tx_{time} = \left(\frac{Fr_{Size}}{Tx_{DR}} \right), \tag{A7}$$

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 Appendix A.1).

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

$$E_{Tx_{elec}} = Tx_{time} \cdot \left((Tx_{elec_{current}} \cdot Tx_{elec_{voltage}}) + (MA_{Tx_{current}} \cdot MA_{Tx_{voltage}}) \right), \tag{A8}$$

where Tx_{time} is found as shown in Equation (A7). $Tx_{elec_{current}}$ 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:

$$E_{Rx} = Rx_{time} \cdot \left((Rx_{current} \cdot Rx_{voltage}) + (MA_{Rx_{current}} \cdot MA_{Rx_{voltage}}) \right), \tag{A9}$$

where Rx_{time} is the time that a node's receiver requires to receive one frame, which can be found by Equation (A10). $Rx_{current}$ 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.

$$Rx_{time} = \left(\frac{Fr_{Size}}{Rx_{DR}} \right), \tag{A10}$$

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 Appendix A.1).

Appendix A.2.2. Energy Consumption of the Microcontroller

The energy consumption of a node's microcontroller, $E_{microcontroller}$, presented in Equation (A2), represents the energy consumed by a node's microcontroller, E_{MA}, and can be found using the following equation:

$$E_{MA} = MA_{Time_{operate}} \cdot MA_{current} \cdot MA_{voltage}, \tag{A11}$$

$MA_{Time_{operate}}$ 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.

Appendix A.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:

$$P_{Rx} = P_{Tx} \cdot G_{Tx} \cdot G_{Rx} \cdot P_{Loss}, \tag{A12}$$

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:

$$P_{Loss} = \left(\frac{\lambda}{4\pi D}\right)^2, \tag{A13}$$

where λ 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 (A14):

$$SNR = 10 \log_{10}\left(\frac{P_{Rx}}{N_{Intern}}\right). \tag{A14}$$

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:

$$F_{Errors} = \left(\frac{BER}{F_{size}}\right), \tag{A15}$$

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.

Appendix A.4. Fast Modular Exponentiation Algorithm

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, Algorithm A1, which has a polynomial running time in the length of e (logarithmic complexity).

Algorithm A1 Square-and-multiply

Input : base a; mod n; binary representation of exponent e $[e_t, e_{t-1}, ..., e_0]_2$
Output: $a^e \ mod \ n$

```
b ← 1
 for (i = t; i > 0; i − −) do
    b = b · b mod n
     if (e_i == 1) then
        b = b · a mod n
     end
end
return b
```

References

1. Wan, L.; Han, G.; Shu, L.; Feng, N.; Zhu, C.; Lloret, J. Distributed parameter estimation for mobile wireless sensor network based on cloud computing in battlefield surveillance system. *IEEE Access* **2015**, *3*, 1729–1739. [CrossRef]
2. Trasviña-Moreno, C.A.; Blasco, R.; Marco, Á.; Casas, R.; Trasviña-Castro, A. Unmanned aerial vehicle based wireless sensor network for marine-coastal environment monitoring. *Sensors* **2017**, *17*, 460. [CrossRef] [PubMed]
3. Noel, A.B.; Abdaoui, A.; Elfouly, T.; Ahmed, M.H.; Badawy, A.; Shehata, M.S. Structural health monitoring using wireless sensor networks: A comprehensive survey. *IEEE Commun. Surv. Tutor.* **2017**, *19*, 1403–1423. [CrossRef]
4. Lin, J.R.; Talty, T.; Tonguz, O.K. A blind zone alert system based on intra-vehicular wireless sensor networks. *IEEE Trans. Ind. Inf.* **2015**, *11*, 476–484. [CrossRef]
5. Liang, T.; Yuan, Y.J. Wearable medical monitoring systems based on wireless networks: A review. *IEEE Sens. J.* **2016**, *16*, 8186–8199. [CrossRef]
6. Iqbal, Z.; Kim, K.; Lee, H.N. A Cooperative Wireless Sensor Network for Indoor Industrial Monitoring. *IEEE Trans. Ind. Inf.* **2017**, *13*, 482–491. [CrossRef]
7. Bapat, V.; Kale, P.; Shinde, V.; Deshpande, N.; Shaligram, A. WSN application for crop protection to divert animal intrusions in the agricultural land. *Comput. Electron. Agric.* **2017**, *133*, 88–96. [CrossRef]
8. Aponte-Luis, J.; Gómez-Galán, J.A.; Gómez-Bravo, F.; Sánchez-Raya, M.; Alcina-Espigado, J.; Teixido-Rovira, P.M. An Efficient Wireless Sensor Network for Industrial Monitoring and Control. *Sensors* **2018**, *18*, 182. [CrossRef] [PubMed]
9. Antolín, D.; Medrano, N.; Calvo, B.; Pérez, F. A wearable wireless sensor network for indoor smart environment monitoring in safety applications. *Sensors* **2017**, *17*, 365. [CrossRef] [PubMed]
10. Adu-Manu, K.S.; Tapparello, C.; Heinzelman, W.; Katsriku, F.A.; Abdulai, J.D. Water Quality Monitoring Using Wireless Sensor Networks: Current Trends and Future Research Directions. *ACM Trans. Sensor Netw.* **2017**, *13*, 4. [CrossRef]
11. Aguirre, E.; Lopez-Iturri, P.; Azpilicueta, L.; Redondo, A.; Astrain, J.J.; Villadangos, J.; Bahillo, A.; Perallos, A.; Falcone, F. Design and Implementation of Context Aware Applications with Wireless Sensor Network Support in Urban Train Transportation Environments. *IEEE Sens. J.* **2017**, *17*, 169–178. [CrossRef]
12. Mainetti, L.; Patrono, L.; Vilei, A. Evolution of wireless sensor networks towards the internet of things: A survey. In Proceedings of the 19th International Conference on Software, Telecommunications and Computer Networks, SoftCOM 2011, Split, Croatia, 15–17 September 2011; pp. 1–6.
13. Lazarescu, M.T. Design of a WSN platform for long-term environmental monitoring for IoT applications. *IEEE J. Emerg. Sel. Top. Circuits Syst.* **2013**, *3*, 45–54. [CrossRef]
14. Kocakulak, M.; Butun, I. An overview of Wireless Sensor Networks towards internet of things. In Proceedings of the 2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 9–11 January 2017.
15. Flammini, A.; Sisinni, E. Wireless sensor networking in the internet of things and cloud computing era. *Procedia Eng.* **2014**, *87*, 672–679. [CrossRef]
16. IEEE 802 Working Group. *IEEE Standard for Local and Metropolitan Area Networks—Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)*; IEEE Std: Piscataway, NJ, USA, 2011; Volume 802, p. 4-2011.
17. Alshammari, M.R.; Elleithy, K.M. Efficient key distribution protocol for wireless sensor networks. In Proceedings of the 2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 8–10 January 2018; pp. 980–985.
18. Akyildiz, I.F.; Su, W.; Sankarasubramaniam, Y.; Cayirci, E. Wireless sensor networks: A survey. *Comput. Netw.* **2002**, *38*, 393–422. [CrossRef]
19. Zhang, J.; Varadharajan, V. Wireless sensor network key management survey and taxonomy. *J. Netw. Comput. Appl.* **2010**, *33*, 63–75. [CrossRef]
20. Shim, K.A. A survey of public-key cryptographic primitives in wireless sensor networks. *IEEE Commun. Surv. Tutor.* **2016**, *18*, 577–601. [CrossRef]
21. Mahajan, P.; Sardana, A. Key distribution schemes in wireless sensor networks: Novel classification and analysis. In *Advances in Computing and Information Technology*; Springer: Berlin, Germany, 2012; pp. 43–53.

22. Bala, S.; Sharma, G.; Verma, A.K. A survey and taxonomy of symmetric key management schemes for wireless sensor networks. In Proceedings of the CUBE International Information Technology Conference, Pune, India, 3–5 September 2012; pp. 585–592.
23. Liao, Y.H.; Lei, C.L.; Wang, A.N. A Robust Grid-Based Key Predistribution Scheme for Sensor Networks. In Proceedings of the 2009 Fourth International Conference on Innovative Computing, Information and Control (ICICIC), Kaohsiung, Taiwan, 7–9 December 2009; pp. 760–763.
24. Quy, N.X.; Kumar, V. A high connectivity pre-distribution key management scheme in grid-based wireless sensor networks. In Proceedings of the 2008 International Conference on Convergence and Hybrid Information Technology, Daejeon, Korea, 28–30 August 2008.
25. Wang, N.C.; Chen, Y.L.; Chen, H.L. An Efficient Grid-Based Pairwise Key Predistribution Scheme for Wireless Sensor Networks. *Wirel. Pers. Commun.* **2014**, *78*, 801–816. [CrossRef]
26. Chan, H.; Perrig, A. PIKE: Peer intermediaries for key establishment in sensor networks. In Proceedings of the INFOCOM 2005—24th Annual Joint Conference of the IEEE Computer and Communications Societies, Miami, FL, USA, 13–17 March 2005; Volume 1, pp. 524–535.
27. Schneier, B. *Applied Cryptography: Protocols, Algorithms, and Source Code In C*; John Wiley & Sons: Hoboken, NJ, USA, 2007.
28. Shamir, A. How to share a secret. *Commun. ACM* **1979**, *22*, 612–613. [CrossRef]
29. Hu, T.; Chen D.; Tian, X. An enhanced polynomial-based key establishment scheme for wireless sensor networks. In Proceedings of the 2008 International Workshop on Education Technology and Training & 2008 International Workshop on Geoscience and Remote Sensing, Shanghai, China, 21–22 December 2008; Volume 2, pp. 809–812.
30. Li, X.; Shen, J. A novel key pre-distribution scheme using one-way hash chain and bivariate polynomial for wireless sensor networks. In Proceedings of the 2009 3rd International Conference on Anti-counterfeiting, Security, and Identification in Communication, Hong Kong, China, 20–22 August 2009; pp. 575–580.
31. Ito, H.; Miyaji, A.; Omote, K. RPoK: A strongly resilient polynomial-based random key pre-distribution scheme for multiphase wireless sensor networks. In Proceedings of the 8th Grobal Communications Conference Exhibition & Industry Forum, IEEE GLOBECOM 2010, Institute of Electrical and Electronics Engineers (IEEE), Miami, FL, USA, 6–10 December 2010.
32. Dai, H.; Xu, H. An improved polynomial-based key predistribution scheme for wireless sensor networks. In Proceedings of the 2010 IEEE Global Telecommunications Conference GLOBECOM 2010, Miami, FL, USA, 6–10 December 2010; pp. 424–428.
33. Delgosha, F.; Ayday, E.; Fekri, F. 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, Honolulu, HI, USA, 12–16 August 2007; pp. 236–241.
34. Das, A.K.; Sengupta, I. An effective group-based key establishment scheme for large-scale wireless sensor networks using bivariate polynomials. In Proceedings of the 2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE '08), Bangalore, India, 6–10 January 2008; pp. 9–16.
35. Baburaj, E. Polynomial and multivariate mapping-based triple-key approach for secure key distribution in wireless sensor networks. *Comput. Electr. Eng.* **2017**, *59*, 274–290.
36. Eschenauer, L.; Gligor, V.D. A key-management scheme for distributed sensor networks. In Proceedings of the 9th ACM Conference on Computer and Communications Security, Washington, DC, USA, 18–22 November 2002; pp. 41–47.
37. Yu, Z. The scheme of public key infrastructure for improving wireless sensor networks security. In Proceedings of the 2012 IEEE 3rd International Conference on Software Engineering and Service Science (ICSESS), Beijing, China, 22–24 June 2012; pp. 527–530.
38. Chung, A.; Roedig, U. 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, 23–25 October 2007.
39. Nagy, N.; Nagy, M.; Akl, S.G. *Quantum Wireless Sensor Networks*; UC. Springer: Berlin, Germany, 2008; pp. 177–188.

40. Li, J.S.; Yang, C.F. Quantum communication in distributed wireless sensor networks. In Proceedings of the 2009 IEEE 6th International Conference on Mobile Adhoc and Sensor Systems, Macau, China, 12–15 October 2009; pp. 1024–1029. [CrossRef]
41. Sheet, X.P.D. XBee-Datasheet.pdf. Available online: www.sparkfun.com/datasheets/Wireless/Zigbee (accessed on 3 May 2018).
42. Atmel. 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 (accessed on 5 June 2018).