Implementing Secure Key Coordination Scheme for Line Topology Wireless Sensor Networks

Walid Elgenaidi, Thomas Newe, Eoin O'Connel,
Muftah Fraifer, Avijit Mathur, Daniel Toal and
Gerard Dooly

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.69484

Abstract

There has been a significant increase in the implementation of wireless sensor networks (WSNs) in different disciplines, including the monitoring of maritime environments, healthcare systems and industrial sectors. WSNs must regulate different sorts of data transmission such as routing protocols and secure key management protocols. An efficient WSNs' architecture must address the capability for remote sensor data management, for example encrypted transmitting data between nodes. This system demonstrates the capability to adapt its sensor members in the network in response to environmental changes or the condition of sensor nodes. The key management technique for any secure application must minimally provide security services such as authenticity, confidentiality, integrity, scalability (S), and flexibility. This chapter studies and analyzes different key management schemes that are implemented in WSN applications and evaluates the performance of secure key coordination algorithm for line topology WSNs. This scheme provides traveling packet for a source to end user via an individually encrypted link between authenticated sensor nodes. It will be shown how security algorithms are applied on a network, such as advanced encryption standard (AES)-based WSNs in real time, e.g., Waspmote sensor platform at the University of Limerick Campus.

Keywords: wireless sensor networks, WSN, security, maritime WSN, dynamic symmetric key update, Waspmote, algorithm, predistribution

1. Introduction

Key management schemes should provide a level of security requirements, low-energy consumption, and low-key storage overhead (KSO). However, some requirements are conflicting and difficult to provide at the same time. For these reasons, an efficient key management scheme should be applied to meet the requirements of the deployed applications. Moreover, distribution techniques that are applicable employ assorted key management schemes such as asymmetrical key cryptography and require numerous communication and computation capabilities. Thus, it is substantial to examine the different requirements, constraints, and evaluate various key management schemes that are applied to wireless sensor network (WSNs). The key management schemes based on WSNs must meet particular criteria for efficiency in light of vulnerability to attackers, including scalability, authenticity, flexibility, resistance value against node capture, key connectivity, and key storage overhead. This chapter investigates the core mechanism in the security of WSNs, which is managing the security keys in the network. Although wireless sensor nodes have limited resources and vulnerable against malicious attacks, different key management schemes in WSNs have been proposed in order to secure the communications between network members.

This chapter is organized as following; after briefly introduced, a detailed review about the existing security key management techniques based on symmetric encryption techniques in WSNs will be presented in Section 2. This chapter will classify schemes into different classes relying on key distribution approaches into three categories; probabilistic predistribution technique, deterministic predistribution technique and combined (probabilistic and deterministic) predistribution technique. The most common security schemes implemented in each technique will be explained and evaluated in detail in Sections 3–5. In Section 6, the performance evaluation of predistributed schemes will be presented in terms of distribution technique (DT), node type (NT), scalability (S), resistance value against node capture (RV), key connectivity (KC), and key storage overhead (KSO). Section 7 will explain the design, outdoor implementation, and evaluation of a secure and efficient key coordination algorithm for line topology-based WSNs. The performance evaluation of the outdoor implementation measurements will be discussed in terms of the received single strength indicator (RSSI) and current consumption. Section 8 highlights how to connect Libelium motes to IoT sensors via Intel Galileo in order to provide a secure solution for an end-to-end IoT system via the cloud. The chapter is concluded in Section 9.

2. Symmetric cryptography key management schemes in WSNs

Most of the key management schemes work in phases including key generation, key transmission, generation of transport key, and revocation of compromised keys. In the symmetric cryptography predistribution scheme, keys are stored in the sensor nodes that are used in a transport key (session key) generation phase. The predistribution key management techniques can be classified based on key distribution mechanism to two types: probabilistic and deterministic scheme. In this section, key management schemes are classified into classes that are relying on key distribution approach as illustrated in **Figure 1**. For a better understanding, we have classified these existing key management schemes into three different categories according to their distribution techniques.

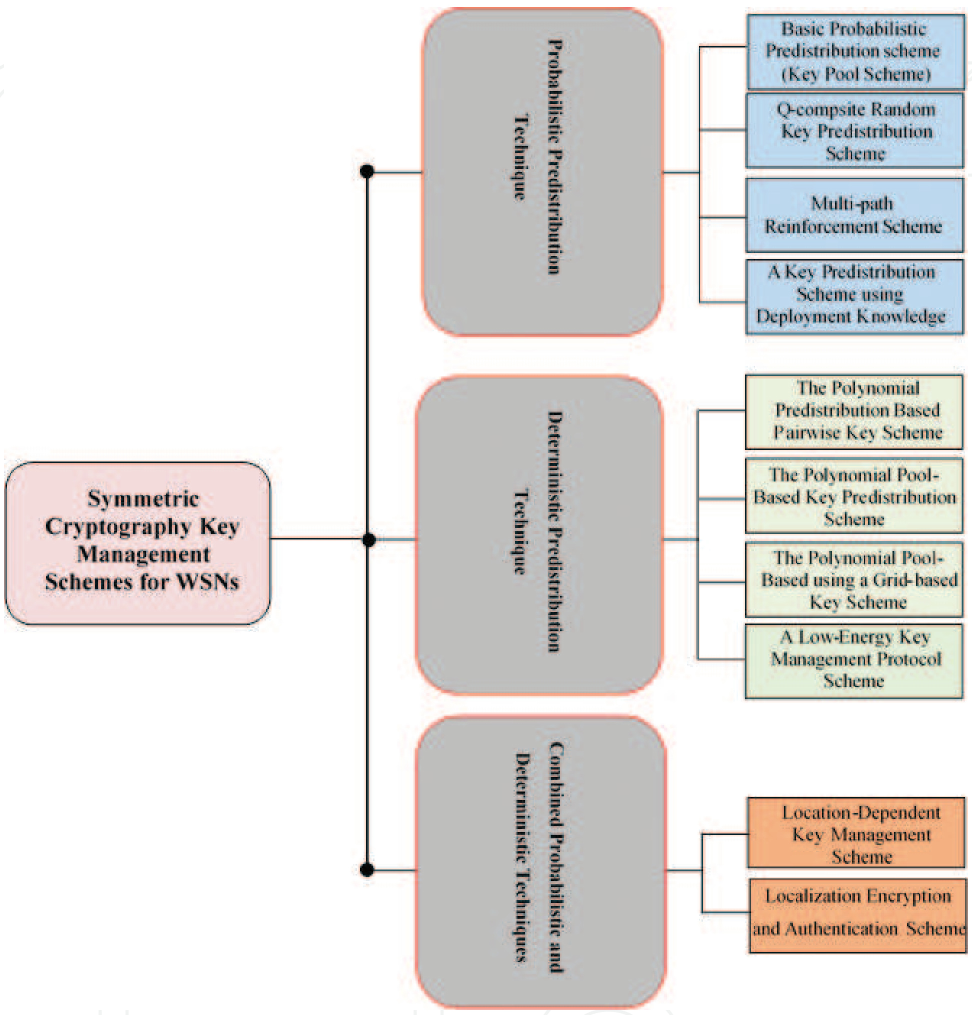
Symmetric Cryptography Key Management Schemes for WSNs
Probabilistic Predistribution Technique
Basic Probabilistic Predistribution scheme (Key Pool Scheme)
Q-compsite Random Key Predistribution Scheme
Multi-path Reinforcement Scheme
A Key Predistribution Scheme using Deployment Knowledge
Deterministic Predistribution Technique
The Polynomial Predistribution Based Pairwise Key Scheme
The Polynomial Pool-Based Key Predistribution Scheme
The Polynomial Pool-Based using a Grid-based Key Scheme
A Low-Energy Key Management Protocol Scheme
Combined Probabilistic and Deterministic Techniques
Location-Dependent Key Management Scheme
Localization Encryption and Authentication Scheme

Figure 1. Classification of key management schemes based on symmetric cryptography for WSNs.

3. Symmetrical probabilistic key management based on predistribution technique

This technique is based on an offline distribution of a number of keys where the identities to each sensor node (key ring) are drawn from a large pool of keys. The main idea in this scheme is that any given pair of nodes in the network will share a common key with a certain probability. The initial step in the deployment phase is the exchanging of keys identifiers between sensor nodes in order to discover their common keys. A secure channel will be established between nodes when two nodes notice that they have the same common key [1].

3.1. The basic probabilistic key predistribution scheme

Figure 2 illustrates the basic probabilistic key predistribution scheme. The overlaps between key rings present the shared keys between sensor nodes. Thus the configuration of the sensor network topology is established relying upon the secure channels between sensor nodes. After the deployment phase, and as a result of the exchange of key identifiers messages between node A and node B, they will use K_4 as their shared common key in order to secure the communication channel between them as shown in **Figure 2**. ON the other hand; if one sensor node is captured, the attacker could reveal the key ring which leads to compromised one or more secure channels in the network.

3.2. Q-composite random key predistribution scheme

Chan et al. [2] proposed a solution to improve the basic probabilistic key predistribution scheme against node capture attack called *q*-composite random key predistribution scheme. This solution used *q* number of common keys to establish a secure channel between sensor nodes, which is boosting the network performance against node capture attack. As the number of keys overlap between two nodes is increased, the scheme performs more resistance against communication link break attack. Therefore, the scheme provides security under small-scale attacks.

3.3. The multipath reinforcement scheme

The multipath reinforcement scheme offers good security [2]. This scheme relies on updating links between nodes after the key setup phase. Since security is more of a concern than bandwidth or power drain, the scheme offers good security with additional sensor network communication overhead. The communication links established between sensor nodes after the key discovery phase in the basic scheme are based on the random selection of keys from the key pool. This allowed sensor nodes in the network to share a subset of the same keys and, thereby, possibly threaten multiple nodes when only one is compromised. Firstly, Anderson and Perrig [3]

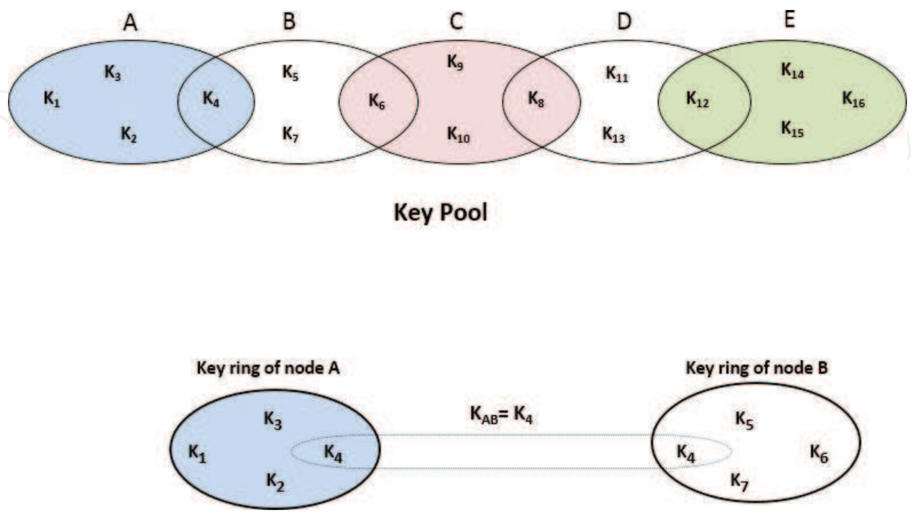

Figure 2. Key pool scheme of key pool size 16 with 4 size of key ring.

applied a multipath to reinforce links in a random key establishment scheme, then Chan et al. [2] used the multipath key reinforcement scheme in order to establish a communication path between every two sensor nodes in WSNs. This technique makes the multipath reinforce scheme stronger with communication links better than that the original basic scheme.

3.4. The basic probabilistic key predistribution scheme using deployment acknowledge

To ensure that the attacker cannot compromise secure channels between noncompromised nodes in Refs. [3, 4] proposed the basic probabilistic key predistribution scheme using deployment acknowledge. This scheme uses non-uniform probability density functions (pdfs) in order to perform deployment acknowledge. In other words, the locations of sensor nodes should be in certain positions. This scheme allowed sensor node to store only the keys that are shared with the other nodes in the networks.

4. Deterministic technique based on predistribution scheme

Generally, in deterministic technique, only the individual keys are predistributed offline. These keys are shared between sensor nodes and the base-station. Thus, after deployment, each node broadcasts only its node identifier using the shared individual keys in order to establish a secure channel with other nodes in the network. However, there is no demand to exchange key identifiers. In predistribution deterministic scheme, the pairwise keys are established after the deployment phase.

4.1. The polynomial-based pairwise key deterministic predistribution scheme

In Ref. [5], polynomial-based pairwise key deterministic predistribution scheme is generated dependent on a bivariate *t*-degree polynomial.

$$f(x,y) = \sum_{i,j=0}^{n} q_{ij} x^i y^j, \quad \text{where } q_{ij} = q_{ji} \tag{1}$$

where q is a prime number above a finite area of Fq, where that should be large enough for a secret cryptographic key, and it has the property of $f(x, y) = f(y, x)$. Initially, every sensor node in the network loaded with a polynomial share of $f(x, y)$, such as $f(i, y)$ in node i. Thus, nodes i and j can generate the common key via evaluating the $f(i,y)$ at point j or $f(j,y)$ at point i.

For example, we assume that sensor nodes 4 and 5 are loaded with a bivariate two-degree polynomial:

$$f(x,y) = x^2 + 3xy + y^2 \tag{2}$$

So the bivariate two-degree polynomial in node 4 is as follows:

$$f(4, y) = 4^2 + 3(4y) + y^2 \tag{3}$$

Also, the bivariate two-degree polynomial in node 5 is as follows:

$$f(5, y) = 5^2 + 3(5y) + y^2 \tag{4}$$

After nodes exchanged their identifiers, sensor node 4 calculates the share pairwise key with node 5:

$$f(5, 4) = 5^2 + 3(5)(4) + 4^2 = 101 \tag{5}$$

And sensor node 5 calculates the shared pairwise key with node 4:

$$f(4, 5) = 4^2 + 3(4)(5) + 5^2 = 101 \tag{6}$$

This scheme is secure and does not expose the link information of other nodes, if n nodes are not compromised. This is making n collusion resistant where the n value relies on the sensor memory storage availability. Since each sensor node in this technique needs to store an n-degree polynomial, which occupies (n + 1) log(q) memory space. Therefore, this scheme is suitable only for nodes with big size of memory storage.

4.2. The polynomial pool-based key predistribution scheme

Modifications based on the basic scheme can address the issues of the scheme. Thus, instead of using a single t-degree polynomial, the polynomial pool-based key predistribution scheme is used. This scheme has three phases:

Step 1. Setup phase: the setup server randomly selects a subset of polynomials Fi from a set of a generated bivariate t-degree polynomials F over finite field Fq which are assigned with a particular ID for the server, and deployed into each node's memory, e.g., node i.

$$F_i \subseteq F \tag{7}$$

Step 2. Direct key establishment: each sensor node in this stage finds the same share of the bivariate polynomial with the other nodes. Next process is that each sensor node establishes the pairwise key.

Step 3. Path key establishment: This phase is based on the key predistribution scheme aforementioned in the previous section.

Initially, randomly selected polynomials are uploaded into each node. However, this scheme will functionally resemble the polynomial pool-based key distribution, if only one polynomial remaining in the pool, or all of the polynomials are 0-degree. While polynomial deployment is the main issue of the setup phase, the complex issue is in the direct key establishment phase, where each sensor node needs to find the other sensor nodes that share the same polynomial. In this stage, sensor nodes use two techniques: deterministic pairwise key based on predistribution scheme [6] and real-time discovery scheme [7]. In the deterministic pairwise key based on predistribution scheme, the knowledge of the nodes with which each node will share a polynomial is preloaded. In this approach, each node carries the node IDs of sensor nodes, which share the same polynomial. The drawbacks of this approach are that sensor node does not offer the flexibility of joining new nodes into the network. In fact, this approach is predistributed information of sensor nodes, which helps an adversary to gain access to the stored data in the case of a compromised node.

In the real-time discovery approach, source node aims to find a communication path with other sensor nodes that did not share polynomial in order to establish a pairwise key. The source node sends a message with the destination node ID to the intermediate nodes that already share a common key. Since the communication paths between sensor nodes that share common keys is secure, we assume that a node that is located between the source and destination shares the same secret key with the destination node. Thus, a communication link between the source and destination nodes has been discovered through which they may discover a common key. However, the issue of real-time discovery approach is the overhead of communication.

4.3. The polynomial pool-based using a grid-based key scheme

Ref. [8] provides all the properties of the polynomial pool-based key predistribution. In this scheme, each two sensors can establish a pairwise key in case of uncompromised nodes and the nodes can communicate with each other. Also if some sensor nodes are compromised, uncompromised nodes still have a great opportunity for the establishment of a pairwise key. By using grid-based key predistribution, a sensor node can determine the other sensor nodes, which can establish a pairwise key with, and it can choose which proper polynomial should be used for key establishment.

This scheme consists of constructing a $m \times m$ grid, where the number of sensor nodes in network can be at most m^2 and a set of $2m$ polynomials is constructed as follows:

$$\{f_i^c(x,y), f_i^r(x,y)\}, \quad i = 0, \ldots m-1 \tag{8}$$

where c and r represent the column and row. Thus, each row i in the grid is associated with a polynomial $f_i^r(x,y)$ where i = 0 to $m - 1$, and each column j in the grid is associated with polynomial $f_j^c(x,y)$ where j = 0 to $m - 1$. The setup server assigns to each node a unique intersection point (i, j) in the grid and allocated polynomial shares of $f_i^r(x,y)$ and $f_j^c(x,y)$. Using this information, each node can perform key discovery and path key establishment. Each sensor node has two bivariate polynomials, while each polynomial (assigned to either the row or the column) shared by m different sensor nodes in the network. Therefore, each node can directly establish two ($m - 1$) pairwise keys with other sensor nodes.

In this scheme, the attacker can prevent the two nodes from establishing a shared key, or by attacking the communication link between two nodes by either compromising the pairwise key. Moreover, the adversary can attack the entire sensor network by attacking a pair of nodes and finding their common key without actually compromising the nodes. This can be done through compromising the sharing polynomial of the two nodes. Thus, the attacker must compromise at least *t*+1 nodes in order to discover the polynomial share. In case the attacker successfully discovers the polynomial that is shared between two nodes, sensor nodes can avoid such an attack by establishing a common key through path key establishment.

4.4. A low-energy key management protocol for wireless sensor networks scheme

The main idea in Ref. [9] is that sensor nodes do not generate keys, and there is no sensor-to-sensor secure communication. Thus, each sensor node needs to store particularly two secret keys to secure the communication links with the base-station and the cluster gateways. Since the sensor nodes are not trusted in this scheme, storing a small number of keys is advantages for the memory constrained. These keys are stored in the flash RAM of sensor nodes before deployment. The gateways must have big size of memory resources in order to store a huge number of keys. Each gateway in the cluster stores all of the keys that are shared with the sensor nodes, keys shared with the command node, and key shared with one other gateway in the network (**Figure 3**). In addition, the command node stores all of the secret keys in the network.

The number of keys stored by the command node is equal to the total number of the gateways and the sensor nodes. Before the deployment phase, each sensor is preloaded with the identifier and the shared key of its cluster gateway. Then after deployment, each sensor node broadcasts "hello" message included in the identifier of its gateway. Therefore, the gateway shapes the cluster and sends assigned message to all sensor nodes in its own gateway cluster. Subsequently, each gateway broadcasts the identifiers of its group to the other gateways in the network. This broadcast stage helps minimize the volume of inter-gateway traffic.

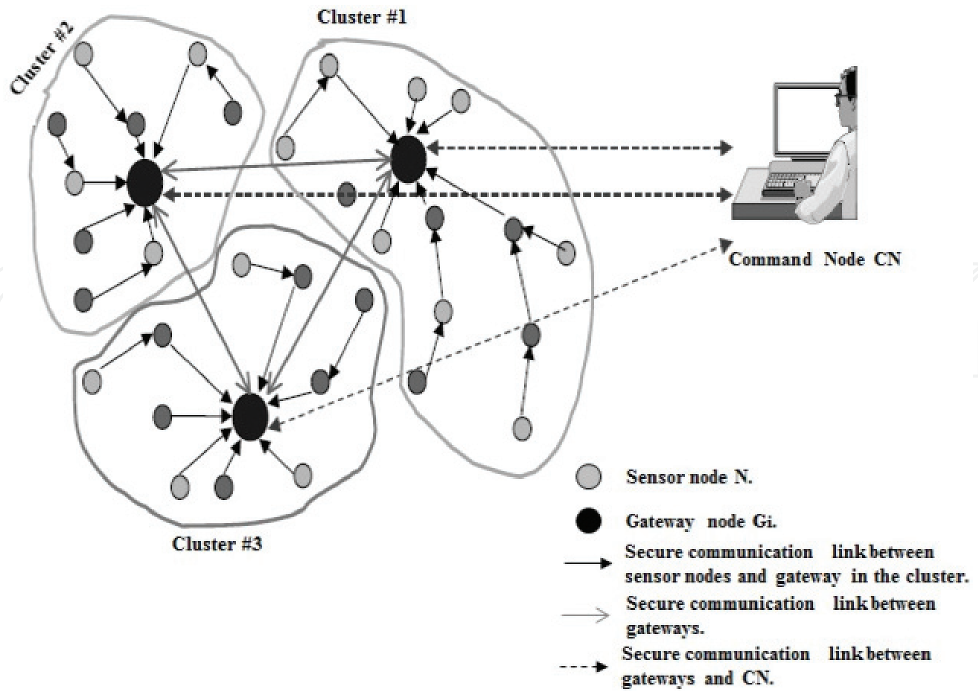
Cluster #1
Cluster #2
Cluster #3
Command Node CN
Sensor node N.
Gateway node Gi.
Secure communication link between sensor nodes and gateway in the cluster.
Secure communication link between gateways.
Secure communication link between gateways and CN.

Figure 3. A low-energy key management protocol for WSNs.

The scheme provides a cost-effective key management infrastructure for security wireless sensor networks and the sensor nodes are not required to store large numbers of keys. Moreover, the scheme supports key revocation and renewal mechanisms, an energy conserving technique to provide key management for sensor networks presented as well.

5. Combined techniques based on predistribution scheme

5.1. Location-dependent key management scheme

Relying on their environment positions of nodes, the location-dependent key management scheme in Ref. [10] decides which keys to load on each node. All sensor nodes in this approach are considered to be static and communicate through encrypted links, also new nodes can be added at any time. Additionally, this scheme is assumed that nodes are transmitting at different power levels and different ranges. In this scheme, the nodes that transmit at different power levels and are tamper proof are called anchors. Each sensor node in the location-dependent key management scheme is preloaded with a subset of keys along with a single common key that is shared between all nodes. These keys are computed by a key server and placed into a key pool. However, the anchor nodes do not get keys from the key pool [11].

5.2. Localization encryption and authentication protocol

Zhu et al. [12] innovated a key management scheme called localization encryption and authentication protocol (LEAP). This scheme offers the main WSNs security requirements such as confidentiality and authentication. Also, LEAP supports diverse communication patterns, including unicast which is addressing a single node, a local broadcast which is addressing a group of nodes in a neighborhood, and global broadcast by addressing all the nodes in the network. One of the crucial points in LEAP is survivability, where compromising of some sensor nodes does not affect the entire network. Furthermore, LEAP is based on the theory that different types of messages exchanged between sensor nodes, where all the packets transferred in the network always need to be authenticated and encryption of packets carrying routing information is not always needed. In LEAP, each sensor node stores four types of keys as illustrated in **Figure 4** including individual, pairwise, cluster, and group.

- *Individual key:* This is a unique key that is shared between each sensor node and the base station in the network. Sensor nodes use this key to secure the communication link with the base station. The individual key is generated and preloaded into each sensor node before the deployment process. The individual key K_{IN} for node *N* (each node has a unique id) is generated as follows:

$$K_{IN} = fK_m(N) \tag{9}$$

  where *f* is a pseudo-random function and K_m is the master key known only to the controller. Thus, the controller might only keep its master key in order to generate any individual key

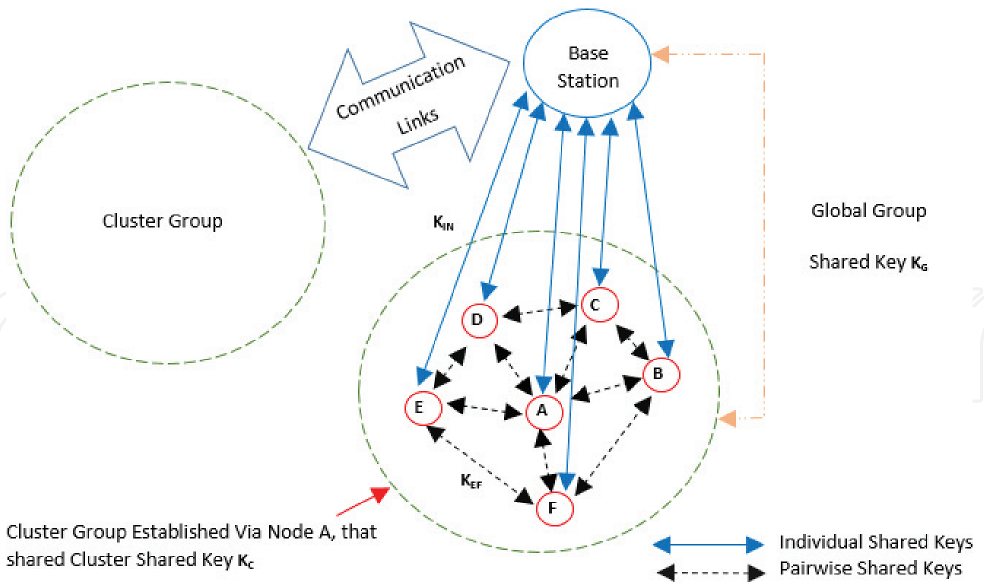

Figure 4. Communication links in LEAP.

rather than store all the individual keys. In fact, the computational efficiency of pseudorandom functions is negligible, thus the controller generates individual keys on the fly.

- *Pairwise key:* This key is shared between each sensor node and its immediate neighbors (one-hop neighbors) in the network. Pairwise Key is used to secure communication links between sensor nodes which are used to transfer individual messages like sharing a cluster key or sending data to an aggregator node. Therefore, this type of pairwise key is most commonly used in the network. This scheme assumes that neighbors of a node are relatively static and a sensor node that is being added to the network will discover most of its neighbors at the time of its initial deployment (T_{est}). Moreover, [12] showed in practice that T_{est} is smaller than the time interval (T_{min}) for an attacker to compromise a sensor node. There are four stages that describe the demonstration of the pairwise key establishment of new deployed node (e.g., *N*) in the network. These stages are key predistribution, neighbor discovery, pairwise key establishment, and key erasure.

- *Key predistribution:* During the initial stage of key predistribution, the K_{IN} is generated by the controller and loaded into each node. Then each node (e.g., *N*) computes a master key.

$$K_N = fK_{IN}(N) \tag{10}$$

- *Neighbor discovery:* The first stage in deployment phase is discovering node's neighbors. By broadcasting a "hello" message that contains its ID, Node *N* starts to discover its neighbors, and in the same time starts a timer of T_{est}. ACK response of the "hello" message contains neighbor ID (e.g., *V*). Node *N* authenticates the ACK using master key K_V as follows:

$$K_V = fK_{IN}(V) \tag{11}$$

Since node *N* knows K_{IN}, it can also derive K_V and then verify node *V*'s identity.

- Pairwise key establishment: Both Nodes N and V compute their pairwise key as the following:

$$K_{NV} = fK_V \tag{12}$$

- *Key erasure:* In the case a timer expires, node N erases K_{IN} and all the master keys (e.g., K_v) of the other neighboring nodes, which it generated in the neighbor discovery stage, and keeps its own master key K_N. Subsequently, the communication links between node N and another node cannot be decrypted without the key K_{IN}.

    - Cluster key: This is a shared key (K_c) between multiple neighboring nodes, which is very important since it is mainly used for securing locally broadcast messages. In case, two neighboring nodes intend to send the same message to the base station, the node with a better message signal may be elected to send the message to the base station. A discovery of this process is only possible to implement if a node shares a common key with its neighboring nodes. In order to reduce the network communication overhead, LEAP uses cluster key to select which messages to transfer. The establishment of the cluster key between neighbors in the network relies on the shared pairwise key. Where node N generates K_c and before it transmits this key to its neighbors, it needs to encrypt that key using the pairwise key that is shared with each neighbor (V_1, V_2, V_3, …, V_n). Next step, node V_1 decrypts and stores the K_c, and then it sends back its cluster key to node N. In node revocation, the new cluster key will be generated and swapped between all the neighboring nodes in the cluster via the same aforementioned method.

    - A global group key: All nodes in the network shared this key including the base station. This key is used by the base station to encrypt messages which are broadcasted to all nodes in the network. A basic approach to establish a global key K_G for a sensor network is to preload each node with the global key in key predistribution stage. However, this key needs to securely update in case a compromised node is revoked. Ref. [12] proposed an efficient mechanism that relied on cluster keys (Kc), where the transmission will only be one key. The mechanism was employed in Ref. [13] for broadcast authentication. The sensor nodes in Ref. [13] are organized into a breadth-first spanning tree, where each node remembers all the neighbors including the parent and children of a spanning tree [14].

6. Evaluation of the performance of key predistribution scheme

This section evaluates the performances of aforementioned predistribution schemes in terms of distribution technique (DT), node type (NT), scalability (S), resistance value against node capture (RV), key connectivity (KC) and key storage overhead (KSO). **Table 1** illustrates the comparison between the schemes.

Scheme	KDT	NT	S	RV	KC	KSO	Ref
The basic probabilistic key predistribution scheme	Probabilistic	All	Good	$1 - \frac{k_{\text{ring}}}{k_p}$	Medium	K_{ring}	[1]
Q-random key predistribution Scheme	Probabilistic	All	Medium	$1 - \frac{k_{\text{ring}}!(k_{\text{ring}}-q)!}{(k_{\text{ring}}-q)!k_p!}$	Poor	K_{ring}	[2]
The basic multipath reinforcement scheme	Probabilistic	All	Good	$1 - \frac{P(2P-P^2)^{N_N}}{0.5865nd^2N_N}$			[2, 3]
The basic probabilistic key predistribution scheme using deployment acknowledge	Probabilistic	All	Good	$1 - \frac{\frac{m}{\lvert S\rvert}}{\left(1-\frac{m}{\lvert S\rvert}\right)^x}$	Good	$\leq K_{ring}$	[3, 4]
The polynomial based pairwise key deterministic predistribution scheme	Deterministic	All	Excellent	$1 - \frac{\left(\frac{k}{\lvert S\rvert}\right)}{\left(1-\frac{k}{\lvert S\rvert}\right)^x}$	Good	ck where ($K_{\text{ring}} \leq s$)	[5]
The polynomial pool-based using a grid-based key scheme [++]	Deterministic	All	Good	$1 - \frac{12m-6}{N}$	Good	$2(t+1)$ $\log(q)+$ $2(t+1)l$	[6–8]
A low-energy key management protocol for wireless sensor networks scheme	Deterministic	1. Cluster gateways	Excellent	$1 - \frac{G-1+CN_1+C_1}{C_{i+1}+CN_{i-1}+\sum_1^t G_i-1}$	Poor	$G+C_i+1$	[9]
		2. Command nodes		$1 - \frac{G}{n+G}$		G	
		3. Ordinary nodes		$1 - \frac{1}{n-1}$		$2\,K$	
Localization encryption and authentication protocol	Combined	All	Excellent	$1 - \frac{1+2N_n+n}{\frac{n(n-1)}{2}+2N_n}$	Excellent	$2N_n+3$	[11–13]

k_{ring} : Key ring.
k_p :Key Pool.
q: composite random key.
p: Basic propabilty of sompromise link.
d: Probability of sharing sufficient communication keys.
n: Number of deplied sensor nodes.
$\lvert S\rvert$: Size of the global key pool.
m: Number of keys carried by each sensor node.
N_n: Number of node neighbors.
x: Number of compromised nodes.
G_k: Keys generation.
C: Total number of independently generated.
Keys via a unique generation key G_k.
S: Polynomials in the sensor node.
G: Number of gateways
C_i: Number of nodes in the cluster.
CN_i: command node.

Table 1. Comparison between predistribution schemes.

6.1. Key distribution technique (KDT)

The aims of the key management protocol are to establish secure communication links that are used to exchange data between sensor nodes and update the encryption keys in case of some node compromised. KDT refers to the approach of distributing secret keys between authorized sensor nodes in WSN in order to establish secure communication links.

6.2. Node type (NT)

As we defined the KDT above, sensor nodes are classified into different categories based on their function. In some cases in WSNs, nodes could work as supervised nodes or supervisor nodes. Thus, in this section, the NT indicates to node classification in the networks.

6.3. Scalability (S)

The term of scalability in WSNs means each key management scheme must allow for the variation in the size of the network. In other words, distributed scheme must be able to add more nodes in case of new nodes request to join the network.

6.4. Resistance value against node capture (RV)

As adversary might attack the network by compromising one or more nodes in the network and reveal the secure communication links between sensor nodes, the attacker can compromise the entire network. Therefore, key management scheme must resist against such attacks. In this section, we evaluate the RV as the ratio between number of compromised communication links (N_{CCL}) and the total communication links (N_{TCL}) in the network when one sensor node is compromised:

$$RV = 1 - \frac{N_{\mathrm{CCL}}}{N_{\mathrm{TCL}}} \tag{13}$$

where RV = 1 implies that capturing one sensor node in a key management scheme will not affect the other secure communication links in the networks. However, RV = 0 implies that capturing one sensor node in a key management scheme will lead to compromise the rest of the security communication links in the network.

6.5. Key connectivity (KC)

KC means the probability of neighboring nodes sharing some common keys or nonneighboring nodes securely communicating via secure intermediate nodes sharing common keys. Therefore, the communication between sensor nodes in the network must be high.

6.6. Key storage overhead (KSO)

Due to the limitations on sensor nodes in term of memory space, KSO is considered as the most important constraint in designing suitable key management scheme in WSNs. In this section, we evaluate KSO as the total number of keys stored in each sensor node in the network.

7. Secure and Efficient Key Coordination Algorithm for Line Topology Network maintenance for use in maritime wireless sensor networks

The Secure and Efficient Key Coordination Algorithm for Line Topology Network (SEKCA) is a symmetric security scheme for a maritime coastal environment monitoring-based WSN [15, 16].

The scheme provides security for travelling packets via individually encrypted links between authenticated neighbors, thus addressing the main issues in security mechanisms based on the symmetric encryption algorithm, including memory, storage space, key generation, and a reiteration of a global rekeying process. Furthermore, SEKCA proposes a dynamic update key based on a trusted node configuration, called a Leader node (L_n), which works as a trusted third party. The technique has been implemented in real time on a Waspmote test bed sensor platform [16]. The deployment of a Waspmote sensor node integrated with the XBee 802.15.4 pro module in an outdoor environment using IEEE 802.15.4 /2.4GHz standard will be used to demonstrate the implementation of the algorithm. This mechanism relies upon the concepts of location-based routing [17]. Packets travel between nodes based on the information of the next repeater node. Subsequently, the static position of the sensor node encourages proposing a strategy for managing and controlling the transmission of secure packets between wireless nodes. In addition, maintaining network connectivity when removing or joining nodes makes it necessary to know the identification of the neighboring node/nodes in order to exchange the cryptographic keys and create a secure communication link. The main idea behind this work is to allow an ordinary node to determine its authenticated neighbor without the use of complex computations. Nodes will make their decision depending on the recommendation message from the L_n [16]. The L_n is located at a calculated distance from the line topology of ordinary nodes. The packet travels from the source to the destination based on the location of the nodes in the neighboring node's list of members, which is coordinated from/by the L_n. Each ordinary node must forward the packet to its next authenticated neighbor through link encryption with its adjacent key, "k_{nj}". The fundamental motivation behind this strategy is to configure a network line topology with simple and scalable security algorithms [16]. **Figure 5** illustrates the deployment of the nodes in our scheme and **Table 2** provides details on the notation used [16].

7.1. Implementation outline of SEKCA

The Waspmote sensor node as shown in **Figure 6** is provided with different frequency radio and protocols. In SEKCA, the XBee-Pro protocol is used for communication between nodes. This provides for a maximum communication distance of 7000 m between nodes which is ideal for the line topology used in a maritime coastal environment monitoring.

SEKCA is based on Advanced Encryption Standard (AES) with a key length of 128 bits, where AES encrypts a block of elements using the electronic codebook (ECB) encryption mode as shown in **Table 3**.

Sensor data "M" is encrypted in the application layer via software with AES 128 using the source key "k_s", which is shared exclusively between the source and the destination nodes. Then, the encrypted frame is encrypted again with the shared adjacent key "k_{nji}" (AES-128), which is shared exclusively between every set of two neighbors as in Eq. (14). The repeater node that forwards the sensor data to the destination in the network will decrypt the information once using the shared adjacent key, "k_{ni}" [16]. Then, to ensure complete confidentiality and privacy, before forwarding the data to the next repeater, the node will encrypt it via its adjacent key, "k_{nj}". Thus, the repeater will not be able to see the original sensor data transmitted due to the encryption with the source key, "k_s" [16]. Equation below shows this process where the red

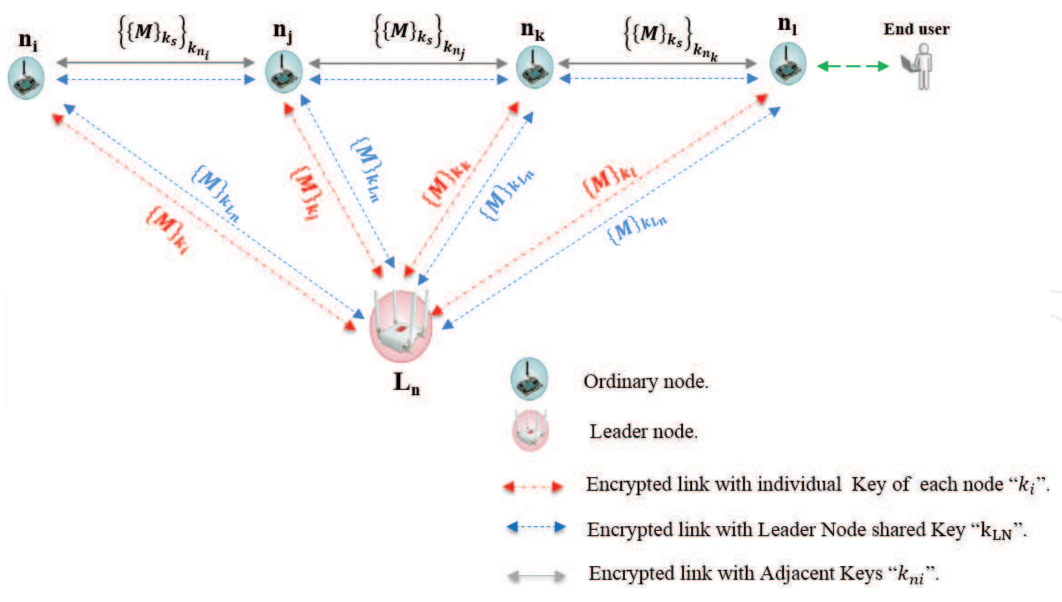
n_i
n_j
n_k
n_l
End user
$\{\{M\}_{k_s}\}_{k_{n_i}}$
$\{\{M\}_{k_s}\}_{k_{n_j}}$
$\{\{M\}_{k_s}\}_{k_{n_k}}$
$\{M\}_{k_{Ln}}$
$\{M\}_{k_i}$
L_n
Ordinary node.
Leader node.
Encrypted link with individual Key of each node "k_i".
Encrypted link with Leader Node shared Key "k_{LN}".
Encrypted link with Adjacent Keys "k_{ni}".

Figure 5. Nodes in line topology "originally published in Ref. [16], under a CC BY license".

text indicates decryption with a shared adjacent key which is shared with the neighbor of the node performing the encryption [16].

$$\rightarrow nj{:}\left\{\{M\}_{k_s}\right\}_k \text{then at } nj{:}\left\{\left\{\{M\}_{k_s}\right\}_k\right\}k \text{ then } nj \rightarrow nk{:}\left\{\{M\}_{k_s}\right\}_{k_{nj}} \text{ etc.} \tag{14}$$

All ordinary nodes and the L_n must share a master key called the Leader node key, "k_{Ln}". This key is used for all the confidential communication between network members in processes such

Symbol	Description
n_i, n_j, n_k, n_l	Ordinary nodes in the line topology.
Ln	Leader node.
$\{\ \}_k$	Symmetric encryption/decryption with key K.
M	Transmitted sensor data.
k_s	Secret encryption source key.
k_{ni}	Secret encryption adjacent key of node i.
k_i	Secret encryption individual key of node i.
k_{Ln}	Secret encryption leader node key.
$\{\{M\}_{ks}\}_{kni}$	Sensor data encrypted with source and adjacent keys.
$\{M\}_{ki}$	Message encrypted with individual key of node i.
$\{M\}_{kLn}$	Message encrypted with leader nod key.

Table 2. Explanation of notation used in **Figure 1**.

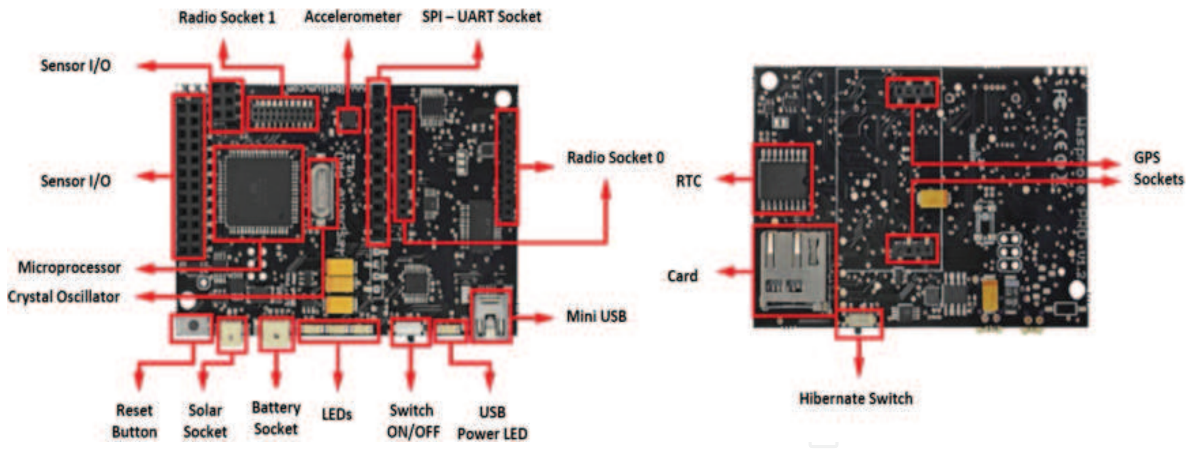

Figure 6. Waspmote components top and bottom sides "originally published in Ref. [16], under a CC BY license".

Algorithm	Key size	Data block size	Mode cipher	Padding
AES-128	128 bits	16 bytes	ECB	ZEROS

Table 3. AES-128 with ECB cipher mode and zeros padding "originally published in Ref. [16], under a CC BY license".

as when a new member joins the network, and for monitoring the behavior of the ordinary nodes [16]. The individual key "k_j" is a unique predistributed key between every ordinary node and the leader node. This key is used in the re-keying phase during the revocation process [16].

In the case of a network member being revoked, only the leader node key 'k_{ln}', and one adjacent key needs to be renewed [16]. Subsequently, every node in the network has a key re-generation mechanism to create a new key [16]. This mechanism relies upon the Message Digest 5 algorithm (MD5) outlined in **Table 4** and a hash of the Real Time Clock (RTC) value as shown below.

$$k = H(\mathrm{RTC}) \tag{15}$$

Algorithm	Output size (bits)	Internal state size (bits)	Block size (bits)	Max message size (bits)	Word size (bit)
MD5	128	128	512	264-1	32

Table 4. MD5 hash algorithm "originally published in Ref. [16], under a CC BY license".

Due to the straight line network topology, only the node located before the revoked node must update its own adjacent key "k_{nj}". This node then needs to share its new key with the new neighbor that replaces the revoked node in the authenticated neighbors list [16]. This stage is coordinated by the L_n via a method of unicasting a revoked message that is encrypted with a predistributed individual key "k_j" [16].

7.2. Practical outdoor implementation of SEKCA

The outdoor implementation involves four Waspmote nodes, a Waspmote Gateway, four XBee 802.15.4 Pro modules with antennae, an MC1322x USB ZigBee dongle, an Agilent 66321D mobile communication DC Source, a Waspmote Pro IDE version 04 with Waspmote Pro API version 013 software based on Arduino, X-CTU provided by Digi, and the Wireshark network analyser [16]. Each Waspmote was placed on a fixed pole at a height of 80 cm from the ground. The effects of temperature and humidity on RSSI in WSNs as in Ref. [18] were considered. In this scenario the temperature was between 20 and 21°C and the humidity was 70%. SEKCA was implemented into three scenarios [16]:

- Four nodes and the gateway at a distance of 80, 120, or 160 m between end points in line topology.
- Three nodes and the gateway at a distance of 80, 120, or 160 m between end points in the line topology.
- One repeater node between sender and the gateway at a distance of 80, 120, or 160 m between ends in the line topology [16].

In this experiment, the received single strength indicator (RSSI) was measured at the gateway. **Figure 7** shows the average value of RSSI which was measured after receiving 300 encrypted packets of 79 bytes in size at a baud rate of 115,200 bps [16]. These values were measured in the three aforementioned scenarios of distances and repeaters. The signal strength in the 80 m scenario was the strongest in the all case of one, two, and three repeaters at exactly—48, −41 and −36 dB, respectively. However, in the case of one repeater the 160 m scenario had the minimum signal strength value at −64 dB. While, in the case of all possible scenarios the maximum achievable signal strength is −36 dB, which is the 80 m distance with three repeaters

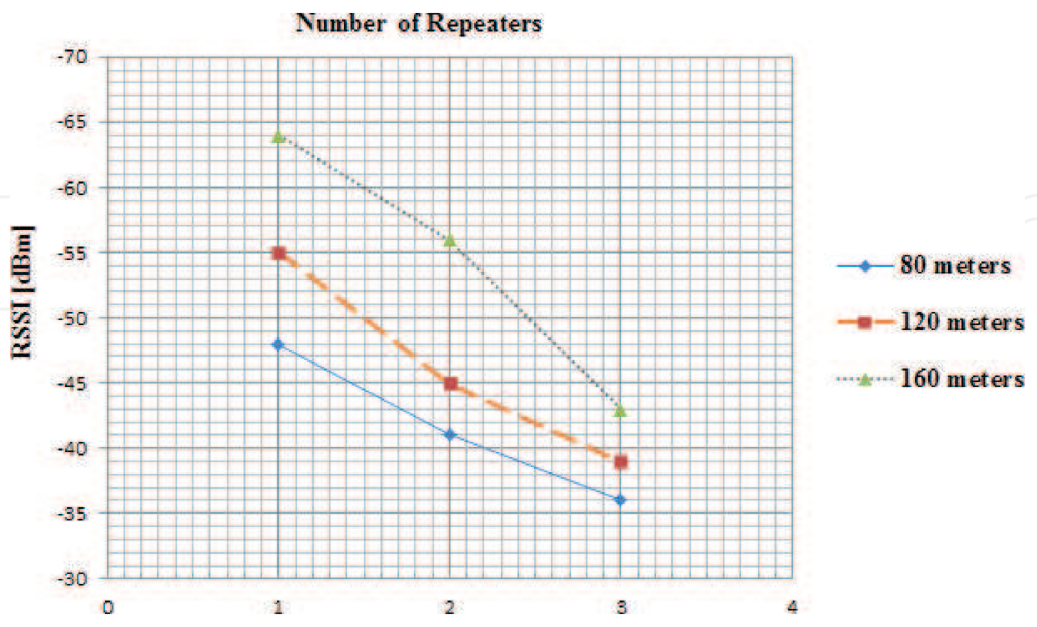

Figure 7. Average RSSI in three different scenarios of outdoor implementation “originally published in Ref. [16], under a CC BY license”.

scenario. These measurements are then used by the leader node to determine the positions of future new joining nodes into the line topology [16].

Data processing relies on the size of the data and the approach used in processing this data. Furthermore, SEKCA has been adapted to use minimum current consumption for data processing. The current consumption of the transmission data has been improved by using sleeping schedules for the receiving/transmitting modules (the XBee current consumption is from 37 to 64 mA with the mode ON fully operational). The measurements were taken using the 66321D-Agilent with input 3.689 V and 0.19999 Ω resistance. The average current consumption of the XBee module is 64.9123 mA when the leader node is operational. The measured current consumption has been improved from 64.9123 to 7.4355 mA, when the system returned to sleeping mode after transmitting/receiving data, which is a reduction of over 88%. **Table 5** provides the current consumption of the scheme modes, this includes when the XBee module is fully functional and sleeping [16].

Scheme mode	Max measured current consumption (mA)	Min measured current consumption (mA)	Average measured current consumption (mA)
Fully functional system	66.732	63.5297	64.9123
XBee module (sleeping)	9.4567	5.6248	7.4355
XBee module current consumption (awake)	57.2753	57.9049	57.4768

Table 5. Measured current consumption of scheme modes "originally published in Ref. [16], under a CC BY license".

SEKCA has improved some common security issues in WSNs [16], including:

- *Efficiency:* 88.55% of current consumption was reduced using the XBee sleeping mode. This will improve the lifetime of all network sensor nodes.
- *Data confidentiality:* By doubling the encryption of the messages, we ensure that only the gateway in the network can decrypt the original data (using AES 128) and after that, we establish peer-to-peer encryption between repeaters [16].
- *Authentication:* Each node has an individual key shared with the leader node, which is used to "sign" the messages in order to ensure the authenticity of the new neighbor and the rekeying. This signing is performed when the leader node reconfigures the network topology in the case of joining or revoking network members [16].
- *Memory capacity storage:* By using the re-generation function, there is no need for each node to store as many keys as are required in some key predistribution schemes where a large pool of keys is located in each node before deployment. In our scheme, each node has only four keys. Therefore, every node requires only a small amount of memory for key storage, which is four times 128 bits [16].
- *Non-repudiation:* by signing the XBee acknowledgment messages, there is now verifiable proof that the information sent has really been received by a specific sensor node. This

signing using a shared source/destination key, *Ks*, that provides weak nonrepudiation as a public key scheme is not yet available in our system.

- *Forward security:* In the reconfiguration of the scheme, new neighbors could not communicate directly before authentication process. The authentication process coordinates via a leader node [16].
- *Backward security:* When a node is revoked from the network, the system ensures that the node cannot receive any new data from the network [16].

8. Toward connecting Libelium Motes to IoT sensors via Intel Galileo

Currently, the system being proposed uses Libelium motes, which connect sensors to the Intel Galileo board. Different Libelium motes show Arduino wireless modules that may be connected to Intel Galileo. These modules are RFID, NFC, Bluetooth, BLE, Wi-Fi, GPRS, 3G, and GPS, which may be connected as Libelium's open hardware division. Now, the Intel Galileo is based on the single core 32bit, 400MHz Quark SoC X1000 processor. In addition, it supports 3.3 or 5 volt shields and has an Ethernet port and two USB ports. Moreover, Galileo software is compatible with Windows, Mac OS and Linux, and the Intel Galileo development board is a prodigious tool for quickly prototyping simple, interactive designs for different types of IoT projects. According to the related works such as Refs. [18–20], it is noticeable that Galileo board is a promising and great tool for any potential IoT projects. When integrating it with sensors it can make big gains thereby bringing several IoT projects to life. Various IoT objects can access and control components such as temperature sensors, humidity sensors, gas sensors and light sensors, and so on. Essentially, sensors are considered the eyes and ears of potential IoT objects by many engineers and researches. This paradigm is proposed to be an end-to-end two-way communication that can be set-up with MQTT protocol as it is a secure solution for an end-to-end IoT system via the cloud. This includes encrypted data in a cloud to keep the data safe in the cloud.

9. Conclusion

This chapter presents an overview of existing key management distribution schemes of a WSN running a symmetric encryption security scheme. A review of probabilistic and deterministic basic key predistribution techniques has shown that the key management-based symmetric encryption algorithm requires less computation and energy consumption; however, these techniques have weaknesses in scalability, resistance value against node capture, and key storage overhead. These weaknesses make them nonapplicable to large-scale wireless sensor networks. Furthermore, different probabilistic and deterministic schemes' basic key predistribution technique was compared in terms of key distribution technique (probabilistic/deterministic), scalability, resistance value against node capture, key connectivity, and key storage overhead in order to show an efficient symmetric-key based scheme for WSNs. In addition to the key management schemes for WSNs, Secure and Efficient Key Coordination Algorithm for Line Topology Network (SEKCA) was also described. The algorithm tackled number of security services such as that made the re-keying process a local operation and minimized the use of key memory storage. The technique was implemented on the Waspmote platform in real time and analysed in terms of a received single strength indicator (RSSI) and current consumption. As cloud computing addresses the issues of data storage, it is also providing a secure solution for an end-to-end IoT system; whereas, security keys could be generated and stored in the cloud.

Acknowledgements

This work was part supported by the SFI Centre for Marine Renewable Energy Ireland (MaREI). Grant References: 12/RC/2302 and 14/SP/2740. In addition, part of the chapter is taken from our previous work as referenced in [16].

Author details

Walid Elgenaidi[1], Thomas Newe[1,3], Eoin O'Connel[1,3]*, Muftah Fraifer[2], Avijit Mathur[1], Daniel Toal[3] and Gerard Dooly[1,3]

*Address all correspondence to: eoin.oconnell@ul.ie

1 Department of Electronic and Computer Engineering, Optical Fibre Sensors Research Centre, Mobile & Marine Robotics Research Centre, University of Limerick, Limerick, Ireland

2 Department of Computer Science and Information System, Interaction Design Centre, the University of Limerick, Limerick, Ireland

3 Department of Electronic and Computer Engineering, Mobile and Marine Robotics Research Centre, University of Limerick, Limerick, Ireland

References

[1] Eschenauer L, Gligor VD. A key-management scheme for distributed sensor networks. In: Proceedings of the 9th ACM conference on Computer and communications security; November 18-22, 2002; Washington, DC, USA. New York, NY, USA: ACM; 2002. pp. 41–47

[2] Chan H, Perrig A, Song D. Random key predistribution schemes for sensor networks. In: IEEE Symposium on Security and Privacy; 11-14 May 2003; Berkeley, CA, USA: IEEE; 2003. pp. 197–213

[3] Anderson R, Perrig A. Key infection: Smart trust for smart dust [Unpublished Manuscript]. 2001

[4] Du W, Deng J, Han YS, Chen S, Varshney PK. A key management scheme for wireless sensor networks using deployment knowledge. Infocom. 2004;**1**:586–597

[5] Rasheed A, Mahapatra R. Key predistribution schemes for establishing pairwise keys with a mobile sink in sensor networks. IEEE Transactions on Parallel and Distributed Systems. 2011;**23:**176–184

[6] Du W, Deng J, Han YS, Varshney PK, Katz J, Khalili A. A pairwise key predistribution scheme for wireless sensor networks. ACM Transactions on Information and System Security. 2005;**8**(2):228–258

[7] Liu D, Ning P. Establishing pairwise keys in distributed sensor networks. In: Proceedings of the 10th ACM Conference on Computer and Communications Security; 27-30 October 2003; Washington DC, USA. New York, NY, USA: ACM; 2003. pp. 52–61

[8] Kalidindi R, Kannan R, Iyengar SS, Durresi A. Sub-grid based key vector assignment: A key pre-distribution scheme for distributed sensor networks. Journal of Pervasive Computing and Communications. 2006;**2**(1):35–43

[9] Jolly G, Ku MC, Kokate P, Younis M. A Low-Energy key management protocol for wireless sensor networks. In: Eighth IEEE International Symposium on Computers and Communication;30 June-3 July 2003; Kemer-Antalya, Turkey, Turkey:IEEE; 2003. pp. 1–6

[10] Anjum F. Location dependent key management using random key-predistribution in sensor networks. In: WiSe '06 Proceedings of the 5th ACM Workshop on Wireless Security. 2006. pp. 21–30

[11] Kifayat K, Merabti M, Shi Q, Llewellyn-jones D. Security in Wireless Sensor Networks. Handbook of Information and Communication Security. Springer Heidelberg Dordrecht London NewYork. 2010. pp. 513–552 DOI 10.1007/978-1-84882-684-7

[12] Zhu S, Setia S, Jajodia S. LEAP+: Efficient security mechanisms for large-scale distributed sensor networks. ACM Transactions on Sensor Networks. 2006;**2**(4):500–528

[13] Liu D, Ning P. Efficient distribution of key chain commitments for broadcast authentication in distributed sensor networks. In: Proceedings of NDSS; 2003. pp. 263–276

[14] Liebeherr J, Dong G. An overlay approach to data security in ad-hoc networks. Ad Hoc Networks. Elsevier. 2006; 5(7):1055–1072. http://dx.doi.org/10.1016/j.adhoc.2006.05.017

[15] Elgenaidi W, Newe T, Connell EO, Toal D, Dooly G, Coleman J. Memory storage administration of security encryption keys for line topology in maritime wireless sensor networks. Sensor. 2016;**16**(2):291–294

[16] Elgenaidi W, Newe T, O'Connell E, Toal D, Dooly G. Secure and efficient key coordination algorithm for line topology network Maintenance for use in maritime wireless sensor networks. Sensors. 2016;**16**(12):2204. DOI: 10.3390/s16122204

[17] Luo H, Ye F, Cheng J, Lu S, Zhang L. TTDD: A Two-tier data dissemination model for Large-scale wireless sensor networks. Journal Wireless Sensor Networks. 2005;**11**(1-2):161–175

[18] Ghayvat H, Liu J, Mukhopadhyay SC, Gui X. Wireless sensor networks: A proposal and implementation for smart home for assisted living. IEEE Sensors Journal. 2005;**15** (12):7341–7348. DOI: 10.1109/jsen.2015.2475626

[19] Belli L, Cirani S, Davoli L, Gorrieri A, Mancin M, Picone M, Ferrari G. Design and deployment of an IoT application-oriented Testbed Computer. 2015;**48**(9):32–40. DOI: 10.1109/mc.2015.253

[20] Kartakis S, Abraham E, McCann J. WaterBox: A testbed for monitoring and controlling smart water networks. CySWater'15. 2015. pp. 1–6