Journal of Ambient Intelligence and Humanized Computing (2019) 10:3349–3364
https://doi.org/10.1007/s12652-018-0772-9

ORIGINAL RESEARCH

A self-managing volatile key scheme for wireless sensor networks

Abdelkader Laouid[1,5] · Abdelnasser Dahmani[4] · Hani Ragab Hassen[3] · Ahcène Bounceur[2] · Reinhardt Euler[2] · Farid Lalem[2] · Abdelkamel Tari[1]

Received: 13 June 2016 / Accepted: 16 March 2018 / Published online: 26 March 2018

Abstract
The last decade has seen major advances in Wireless Sensor Networks (WSN). The limited amount of resources in WSNs is the main challenge for securing them. Indeed, sensors are inherently small devices with limited embedded storage, processor and battery capacity. The situation is even worse when attackers, with virtually unlimited amount of resources, have direct physical access to sensors. Cryptographic keys are used for authentication, authorization, confidentiality, data integrity, as well as many other security services. Several proposals have been made for key management in WSNs. In this paper, we review some notable key management schemes and propose a new one. The originality of our work lies in two main facts: first, we do not place a master key on all sensors before deploying them as several other proposals did, but we rather place the master key in a subset of sensors; second, the master keys are volatile, i.e., sensors use the master key to bootstrap the system and they delete them shortly after that. Our extensive simulations have shown the efficiency of our approach.

Keywords Key management · Wireless sensor networks · Secure connectivity coverage · Network scalability · Distributed algorithms

1 Introduction

The theories, the applications and the benefits obtained from the new generation of embedded devices leads the search community to design and integrate the required concepts that characterized them such as dynamicity, scalability, resource consumption and so on (Shi et al. 2011). For instance, pervasive healthcare systems, smart grids, and unmanned aircraft systems are examples of Cyber-Physical Systems (CPSs) and Internet of Things (IoT) networks. In fact, the CPSs and IoT are the combination of computational elements and physical entities that can interact with humans through small physical entities called sensors. Sensors are relatively small devices that can measure environment parameters, such as temperature, pressure and vibration. This has led to a versatile wide spectrum of applications, ranging from nuclear site monitoring to body sensor networks. Sensors are typically interconnected using their wireless network interfaces, thus forming a *Wireless Sensor Network* (*WSN*). One or more base stations are used to collect sensor measurements. These sensor measurements are exposed to several attack vectors while information is transmitted within the network before reaching the base stations (Padmavathi et al. 2009; Karlof and Wagner 2003).

Security issues have lead researchers to propose a multitude of key management schemes, as for instance (Reegan and Baburaj 2013; Xiao et al. 2007). These schemes can be classified according to several criteria as illustrated in Fig. 1, where the taxonomy of key management schemes is based on their encryption key mechanism, as well as on their key establishment strategies (Diop et al. 2013; Zhang and Varadharajan 2010). In self-enforcing schemes, an *asymmetric key management* class, the establishment of keys is made using asymmetric cryptography which is currently rarely used regarding the resource allocation and therefore not suitable

✉ Abdelkader Laouid
abdelkader-laouid@univ-eloued.dz

1 Department of Computer Science, LIMED Laboratory, University of Bejaia, Bejaia, Algeria

2 Lab-STICC UMR CNRS 6285, Université de Bretagne Occidentale, 20 Avenue Victor Le Gorgeu, 29238 Brest, France

3 School of Mathematical and Computer Sciences, Heriot-Watt University, Edinburgh, UK

4 University Center of Tamanrasset, Tamanrasset, Algeria

5 University of Echahid Hamma Lakhdar, PO Box 789, 39000 El-Oued, Algeria

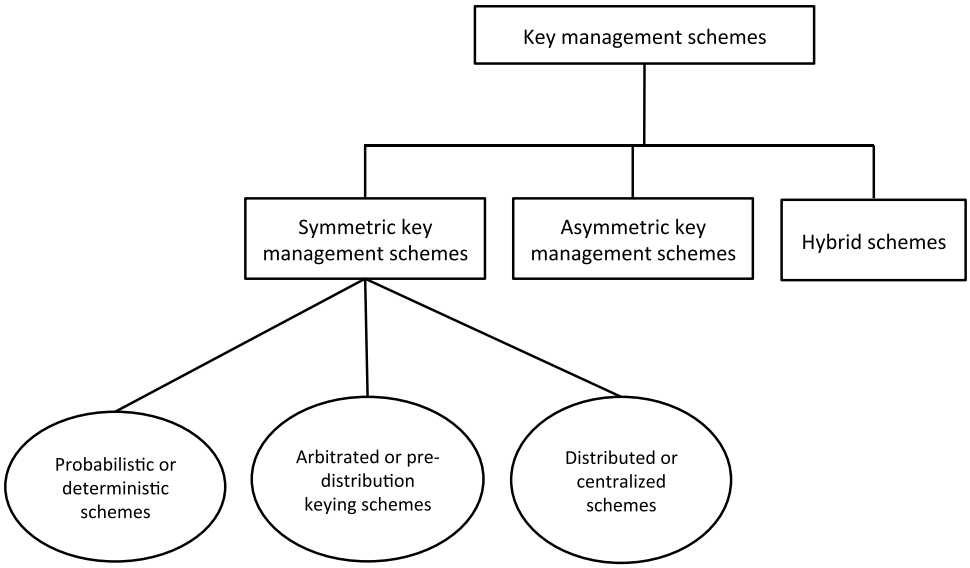

Fig. 1 Taxonomy of existing key management schemes

for sensors (Kavitha and Sridharan 2010). Instead, most of the existing security schemes make use of a *symmetric cryptography system* to design an efficient key management scheme because these schemes require less computation time than other schemes. However, the essential disadvantage of symmetric encryption is the key distribution that is mostly done through a third party (Abomhara et al. 2010). Another possibility is to use hybrid key establishment schemes allowing to reduce the high resource allocation cost on the sensors by placing them on the base station side. However, this solution has some disadvantages such as the incurred overhead from the rising communication between the network nodes and the base station (Chen et al. 2009). Hence, the management of symmetric, or hybrid, keys becomes problematic.

An adequate security mechanism is indispensable to be deployed in order to provide the required security services such as *authentication*, *authorization*, *confidentiality* and *data integrity*. These services are enforced using cryptographic keys. This is why key management is a crucial requirement for *WSNs*. This paper is an extended version of previously published work (Laouid et al. 2016), and it aims, together with some improvements, to propose, implement and analyze a new *Self-managing Volatile Key Scheme* (*Self-VKS*) for *WSNs*. For this purpose, a *symmetric key management scheme* is proposed in order to establish keys between adjacent nodes without using key pools or rings. This contribution is based on a pair of plain and its encrypted correspondent nonce used from each network node in the pre-initialization step, where during this step the master key K_v will be only inserted into some nodes which ensures that an attacker cannot reveal the used K_v from any given node immediately after the deployment, even if he succeeds in capturing a node from the network. The probability to capture a node that has the master key is k / N, where k is the number of nodes that have the master key. Immediately after their deployment, the nodes which have K_v start to share in a secure manner distinguished keys with their adjacent nodes. To ensure the propagation of K_v, every node which holds K_v securely forwards prior to its definite removal to its adjacent nodes using the newly established key and an efficient algorithm. This technique is used to guarantee a short lifetime of K_v over every node and over the whole network as well. In comparison to (Laouid et al. 2016), this work contains some improvements, implementations and a thorough analysis, which can be summarized as follows:

- The ambiguity of previous work of where to start the keying procedure is not efficient in time duration terms. In this extended work, the principle is updated to the following: every node starts the keying procedure with all its neighbors but, in case that two neighbor nodes hold K_v and each of them sends a pair request to each other, the node with a smaller identifier will update the shared key by using the received parameters. Moreover, using the newly proposed principle resolves all those interlock situations that may occur when some nodes are waiting indefinitely for the volatile key to execute the pairwise key process with their neighbors.
- The scheme of the extended work becomes more flexible since the addition of a new node and the node mobility are taken into account.
- Each node has to store a set of prime number groups and a pseudo-random function. This principle is still the same but in order to avoid nested loops finding the intersection between two groups and to minimize the size of used groups, the prime numbers are reordered in a one-row table containing only one instance of each prime number.

– Through implementation and evaluation in a wireless sensor testbed, our extended work clearly demonstrates that the proposed scheme ensures acceptable minimal time duration, resource allocation and security level.

The rest of the paper is organized as follows. In Sect. 2, the notable existing *key management schemes* are discussed. We describe our scheme and give the details of its operation in Sect. 3. A thorough analysis and comparative study with respect to similar key management schemes is presented in Sect. 4. Section 5 concludes our work and identifies future research directions.

2 Related work

A good deal of studies has been proposed on the topic of key management schemes for wireless sensor networks. Following the taxonomy illustrated in Fig. 1, authors classify the proposed key management schemes into three categories based on their encryption techniques: *symmetric*, *asymmetric* and *hybrid*. The symmetric schemes are then divided into three subcategories based on the key establishment mechanism.

Public key technology is mainly used in Internet applications. The existing asymmetric security mechanisms for wire-line and wireless networks cannot apply to wireless sensor networks because of the constrained energy, memory and computation capability (Zhang and Varadharajan 2010). However, some research groups have introduced improvements to successfully implement the asymmetric cryptography in wireless sensor networks. An example: the author in Sahingoz (2013) proposes an asymmetric key management system for wireless sensor networks. A hybrid network structure is used, where the proposed model consists of four main components: sensor nodes, cluster heads, a sink node and an *Unmanned Aerial Vehicle* (UAV). To establish a secret key between two adjacent nodes, each sensor node stores only its private key and the public key of the UAV. After the deployment and prior to the secure exchange of messages between two nodes, there is a need to know the ID of each other. Hereafter, each sensor node should obtain the public key of its neighbor nodes from the UAV.

Some authors have proposed hybrid key establishment schemes for wireless sensor networks. The motivation is to exploit the difference among the base station, the cluster heads and the sensors. In Rahman and Sampalli (2012), the authors propose a hybrid key management scheme for hierarchical WSNs, in which they combine a pairing based encryption with a dynamic pairwise symmetric key management. More precisely, the pairing based encryption is an identity based encryption which is used only in key agreement between key managers (because of its computational overhead). The scheme of Rahman and Sampalli (2012) provides a mechanism for key refresh and revocation. Regarding node addition, the base station follows the new node addition process discussed in Rahman et al. (2010), adding a private secret for the key manager. The sensor nodes are supposed to be homogeneous, but the key agreements inter- and intra-clusters are different. Therefore, the key manager nodes will die before other nodes, which causes malfunction of the whole network. The ECC algorithm is used in Huang et al. (2003) to perform security functions on sensors with limited computing resources. Compared with other public key crypto algorithms, much smaller key lengths are required with ECC to provide a desired level of security, which means faster processing speed, smaller communication complexity, in addition to smaller key storage requirements. Moreover, in order to prevent the impersonation attack, certificates are used which provide a mechanism to check cryptographically to whom the public key belongs and if the device is a legitimate member of a particular network. The certificates are acquired only before each device joins the network using the elliptic curve implicit certificate scheme. However, a performance evaluation shows that the sensors need more battery and computational resources while the security manager is much more powerful, hence we restrict the attention to the efficient symmetric key management side only.

Symmetric key management schemes can be either deterministic or probabilistic (Yang et al. 2015). In the deterministic case, a total secure connectivity coverage is guaranteed, in as much as every two neighboring nodes can establish a direct secure link. This type of schemes enables a flexible deployment and a simple key establishment. However, some key information may not be used which causes a waste of memory space. In probabilistic schemes, the secure connectivity is conditioned on the existence of shared keys between adjacent nodes. The advantage of this kind of schemes is that every two sensor nodes can establish the session key directly. However, the flexibility of deployment decreases and computational overhead of key negotiation becomes large. In Bechkit et al. (2013), the authors propose a probabilistic pre-distribution key management scheme based on unital design theory, to solve the dependence problem between the key ring design and the network size of existing schemes. In fact, unital design theory allows a smart building of blocks with unique features that allow to cope with scalability and connectivity issues. Before deployment, each node is preloaded with t completely disjoint blocks of order m unital designs, where each block corresponds to a key set. After deployment, neighboring nodes will identify their common keys using key identifiers corresponding to their respective distinct key rings. When two nodes have more than one common key, they will use a hash function to obtain the shared pairwise key. Hence, in the absence of the common point between two adjacent nodes, the pairwise key is established

through a secure path. We are once again facing a protocol, where the choice of its parameters (the value of t) is crucial. The latter affects the security, scalability, storage and even the energy consumption of the protocol. The authors of Deng et al. (2005) propose a deterministic transitory master key scheme for pairwise key setup, which is an improvement of the LEAP scheme. To limit the impact of master key compromising, they use it for authentication and do not include it in the computation of the pairwise keys. Furthermore, the authors suggest erasing the master key from the nodes, just after they have established the pairwise key with their neighbors. A new version of this protocol is proposed, in which they consider the addition of a new node. Therefore, before the destruction of the master key, each node will generate a new key using a node identifier and the master key. In addition, every node generates a number of verifiers, which consist of a random number and its combination with the master key. These parameters are used afterwards to authenticate the joining node. This point represents a major drawback of this protocol. Indeed, increasing the number of the verifiers augments the number of nodes that can be added, which causes a noticeable waste of memory space. Otherwise, the memory space reduction limits the number of nodes to be added. Authors of Zhang et al. (2011) propose a distributed deterministic key management scheme for hierarchical WSNs. In this protocol, every sensor node is preloaded with a networkwide shared pseudo-random function and an initial master key. From these parameters and its identifier the node calculates its individual key. After deployment and during the initialization phase, each node sends a network joining message. The pairwise key and the local cluster key will be obtained using information contained in this message. Therefore, the communication overhead is reduced. Data transferred within the network are always authenticated, although not necessarily encrypted. The keys are updated after having been used a certain number of times (a threshold). As for mobile nodes and new nodes joining the network, the keys are established using an elliptic curve digital signature algorithm. The main advantage of this protocol is that the pairwise keys are decentralized, which implies that compromising a node does not affect any other non-compromised pairwise key. However, this protocol is not adequate for dense and large-scale networks.

Another factor partitions the symmetric key management schemes into arbitrated keying schemes or pre-distribution schemes Jr. et al. (2010). Concerning arbitrated keying schemes, the key establishment must pass through a trusted central point. These schemes are attractive under the condition that the central point is considered totally secure, whereas in the pre-distribution schemes the keys are loaded into every node before the deployment. This is suitable for WSNs, because we do not need a central station after the deployment. Authors of Gandino et al. (2009) propose a new version of an arbitrated key distribution scheme for distributed wireless sensor networks. The additional aspect is the use of a transitory master key to perform some transformation in order to increase the number of different keys used in the network. The initialization phase is the same as in classical random key distribution schemes, namely, that each node performs with his neighbor a shared key discovery from the ID of a starting key pool. The next step is the arrangement of the two nodes on a randomly chosen number of iterations. According to this value, they operate a transformation of the shared key using the master key. In addition to the shared key, the remaining keys also undergo a transformation according to a random number of iterations. Finally, any secret material apart from the final pairwise key is erased. Thus, this protocol slightly increases the security of the random key distribution schemes at the expense of storage space. Indeed, the stored parameters are numerous. The authors of Du et al. (2003) propose a key pre-distribution scheme built on Blom's method to benefit from their λ -secure property, and to combine it with random key pre-distribution. Blom's method uses one key space, which ensures the existence of a unique key shared by every two nodes. This property can be represented by a complete graph. The reasoning of the authors is to use a connected graph instead of a complete graph. Hence, they use multiple key spaces instead of one, in the aim to reduce key information in each node. It follows that two nodes with a common space can calculate a pairwise key. On the other hand, they can conduct a key agreement via other nodes which share pairwise keys with them. Furthermore, the authors suggest to treat two-hop neighbors as direct neighbors. Thus, one-hop neighbors must send the received message in turn. We note, however, as an important aspect, that the protocol proposed does not provide a key refresh process. Also, it is complex to choose the parameter values that decide upon security, performance and network connectivity. The authors of Al and Yoshigoe (2008) propose a key management scheme for hierarchical wireless sensor networks, where the key is updated at every sending using the last message received. Each node possesses its own pre-distributed key and a key generator which it has received from a supernode. These two keys will be used to generate the first encrypted key. Regarding the following encrypted keys, they will be calculated using the pre-distributed key together with the last received message. Occasionally, supernodes will distribute a new key generator and the previous steps are executed again. The one-time-keys are also protected by a masking policy, that is used as part of the hidden operation. We also note, that the master key is the same and inserted into all sensor nodes. This implies that the capture of a given node at the moment of the deployment may compromise the entire network.

Furthermore, symmetric key management schemes can be categorised into distributed and centralized based on whether a

central key controller is involved for key distribution (He et al. 2013). In the distributed case, the neighboring nodes collaborate to reach key negotiation, whereas in centralized schemes, the key establishment involves a central key controller. Therefore, in the latter, the security of the network closely depends on the security of the central key controller. The authors of Mehmood et al. (2017) propose an Inter-Cluster Multiple Key Distribution Scheme for WSN (ICMDS) for securing the exchanged information between the sensors and their respective cluster heads. In fact, ICMDS uses a pre-distribution key technique which works efficiently in multi-hop clustering environments. However, in ICMDS the base station is exploited as a third part for computing the shared key, where a centralized solution in WSN environments is not always preferred. A key predistribution scheme is proposed in Gao et al. (2017) based on mixed-level orthogonal arrays, which enables sensor nodes to communicate securely. The secure connectivity and resilience are ensured but several choices for the parameters can be used which leads to manually configure the network. In Chen et al. (2011), the authors propose a dynamic distribution key management scheme. The principle is to use the data key only once to send a single packet. This key is calculated from the source identifier, a counter, the sharing session key and a dynamic distribution key. The distribution key, in turn, is calculated from the source and destination identifiers, a counter, the sharing session key and a random number. To ensure the confidentiality of the distribution key, the sharing session key is used to encrypt it. Consequently, a transmitted packet consists of the cipher text of data and the cipher text of a distributed key. At the reception, the node will decrypt the data and will use the distribution key to calculate the data key of the next package. The weakness of this protocol is that if the sharing session key is compromised, the whole network may be compromised. The dynamic pairwise symmetric key management used for key agreement between nodes of the same cluster is described in Rahman et al. (2010), and based on Du et al. (2003) as we discussed previously.

As a conclusion, the security services can be ensured by cryptographic keys and key distribution mechanisms. Selecting the most appropriate mechanisms is one of the vital parts of the system. The selected mechanism should meet the constraints of sensor nodes like power consumption, code size, data size, and processing time. In our proposed system, we aim to develop a secure WSN system minimizing these resource consumptions and maximizing security performance.

3 Proposed self-managing volatile key scheme

This paper extends the work published in Laouid et al. (2016) and beyond, contains some improvements to ensure the efficiency of the proposed scheme by minimizing the lifetime of the volatile key over the nodes. Moreover, the proposed *Self-VKS* will be thoroughly analysed in the next section to show its feasibility in reality. In fact, the motivation to extend the published work is to exploit simple operations allowing to respond to issues and security metrics encountered in WSNs (He et al. 2013; Simplício et al. 2010). Table 1 shows a set of used primitives in the proposed deterministic scheme. To achieve the aim that every two adjacent nodes share the same key between them, all nodes should be pre-loaded by initial information prior to the deployment. As a next step, each pair of adjacent nodes securely establishes its own key using the initial configuration where a Message Authentication Code (*MAC*) is used before starting the secure communication in view of an authenticated channel. The initial information is given by a pseudo-random function, finite groups which contain prime numbers and a volatile master key (only inserted in some particular nodes). From here on, we use the term of Volatile Key (K_v) instead of the name Master Key since in the proposed scheme the master key may appear in some nodes but not in others. A solid key management scheme is a scheme which considers that an attacker may have a very high power computation resource to extract the master key prior to the erasing time running out. In (Zhang et al., 2011) and (Deng et al., 2005), it is supposed that an attacker can capture any node from the network in order to extract the master key before accomplishing the pairwise key step. Authors in (Zhang et al., 2011) assume that an attacker cannot extract the key prior to 10s, and to ensure the confidentiality of the master key,

Table 1 Primitives used in our deterministic scheme

Notation	Description
N	Number of deployed nodes
n_i	The ith node in the network
ID_i	Identifier of node i
$n_i \rightarrow type, *: M$	Node n_i broadcasts the message M to all nodes in its radio range
PReq and *PResp*	Request and Response, respectively, of a pairwise key between two nodes
$nonce_i$	Random value generated by n_i
X_i	Arbitrary group selected by n_i
K_v	Volatile master key used to share securely the $nonce_i$ between n_i and n_j
$f(int, int)$	Pseudo-random generator that takes 2 integers as parameters
K_{ij}	Pairwise secret key between n_i and n_j
counter	Used to prevent a data replay attack; can also be used as *nonce* (Rogaway 2004)
$E_k(M)$ and $D_k(M)$	Encryption and decryption of message M using the key k

Table 2 Prime number group example.

Group	Prime numbers			
A	P_1	P_2	P_3	P_4
B	P_1	P_5	P_6	P_7
C	P_2	P_5	P_8	P_9
D	P_3	P_6	P_8	P_{10}
E	P_4	P_7	P_9	P_{10}

Table 3 Plain and encrypted data (P_i, C_i) inserted into node n_i

Plain data	Encrypted data
$nonce_i \| \| X_i$	$E_{K_v}(nonce_i \| \| X_i) \| \| MAC$

a countdown is defined to erase all initial data in time less than 10s. Indeed, it is possible to extract the master key in this case if an attacker ceases or destroys the countdown before reaching 10s. Moreover, the authors assume that every deployed node will run correctly after the deployment until the erasing key step, which is strongly impossible in case that thousands of nodes are deployed.

3.1 Pre-configuration step

To reach the objective that any two adjacent nodes can securely share the same key, they are pre-configured by confidential and public information prior to the deployment step. The plain and its encrypted correspondent are used in this proposition to securely share the same value of a variable between every pair of adjacent nodes n_i and n_j. Moreover, a distributed decision during the computation of this value from n_i and n_j is considered. The scenario, is that each pair of adjacent nodes n_i and n_j securely generates the private key K_{ij} using the generated variable values as parameters for a pseudo-random function. In the pre-configuration step, it shall insert into every network node a plain with its encrypted correspondent of random nonce (P_i, C_i) and a selected group as shown in Table 3. Also, a finite number of prime numbers, selected randomly and divided into groups in such a way that all intersections between two distinct groups contain a single prime number, as illustrated by an example in Table 2, is inserted into every network node. The goal of these groups is to distribute the task of key computation between every two adjacent nodes, where each node n_i and n_j randomly selects one group, and the intersection value of both selected groups is considered as a common prime number selected by n_i and n_j.

A volatile key K_v is used only to securely establish the pairwise key before removing it. Therefore, each legitimate node is pre-loaded by the following information prior to the deployment step:

- The pre-defined prime number groups are considered public data accessible by anyone.
- A pseudo-random function for the generation of nonces, which in turn are considered confidential data and which should only be sent securely.
- A hash function to compute the message authentication code *MAC* which is used to compute the shared key K_{ij}.
- An encryption function to encrypt and decrypt data.

3.1.1 Contribution

The major contribution of this work is to respond to the issues raised in (Zhang et al. 2011) and (Deng et al., 2005), that the master key is inserted into each legitimate node, and its life duration is very short to avoid being recovered by an attacker. However, an attacker can capture any network node immediately after the deployment to recover the used master key. Therefore, in this work, K_v is only inserted into some particular nodes. In order to minimize the probability p for a volatile key to be recoverable by an attacker without losing its short lifetime duration, we insert K_v into k nodes prior to the deployment step. The number of nodes which do not hold K_v equals l so that $k \cup l = N$. In fact, the proposed scheme works efficiently if K_v is inserted into nodes that are well secured and which cannot be captured by an attacker. For instance, in the battle scenario K_v can be inserted into the sensors that are in the border. However, even if an attacker succeeds in capturing a node from the network, the probability to capture a node that has the master key is k / N, where k is the number of nodes that have the master key.

3.2 Deployment and keying step

Every network node executes the following steps to share a secret key with each of its neighbors:

- *Discovering step* immediately after the deployment, every node broadcasts a discovering neighbor message. The discovering neighbor message aims to discover the neighboring nodes and is also used as a *PReq* message to start the keying step. The discover message contains the identifier of the source, the encrypted nonce and group. From here on, *encrypted data* means the encrypted nonce and group as shown in Table 3. To avoid the collision of broadcasting at a same time, every node implements a binary exponential backoff algorithm to avoid collisions before broadcasting.
- *Storing step* every node stores for each received *PReq* the identifier and the encrypted data of the received *PReq*.
- *Distributed computation step* hereafter, a node n_i starts the computation of the shared key with its neighbors if it has the volatile key. This technique is used to guarantee a

short lifetime of the volatile key over the whole network. An authenticated broadcast message is used to prevent an attacker from starting a false pairwise request. Otherwise, the node waits until receiving a *PResp* from one of its neighbors. In case that a node n_i holds K_v, it starts the keying procedure with one of its neighbors, say n_j, using the stored identifiers and the *encrypted data*, and it resends to n_j, in plain format, *PResp* which contains its identifier and the selected group X_i.

– *Computing the shared key step* after the decryption of the $D_{K_v}(E_{K_v}(nonce_j||X_j))$ from n_i, each node can compute the shared value $nonce_{ij}$ using Eq. (1). Therefore, using the scenario described above, every pair of neighboring nodes is able to compute a common key, as illustrated in Fig. 2, where ***Gen*** means the generation of a random *nonce*, and ***Sel*** means the selection of an arbitrary group. Finally, the key shred by n_i and n_j is computed using the pseudo-random function as shown in Equation 2. The scenario of these steps is shown in Algorithm 1 which should be executed by every legitimate node immediately after the deployment.

$$nonce_{ij} = nonce_j \ mod \ (X_i \cap X_j) \tag{1}$$

$$K_{ij} = f(nonce_j||ID_j, nonce_{ij}||ID_i) \tag{2}$$

P_1	P_2	P_5	P_3	P_6	P_8	P_4	P_7	P_9	P_{10}

3.2.1 Efficiency settings

In order to maximize the keying time duration, the concerned node securely sends the volatile key using the shared key K_{ij} in the *PResp* message, where the node n_j computes K_{ij} before sending the *PResp* as shown in Fig. 2. Also, n_i computes K_{ij} before revealing K_v. Equation 1 gives $nonce_{ij} = nonce_j$ if $nonce_j < P_{ij}$, meaning that the shared key between n_i and n_j is not a distributed task and may decrease confidentiality. Therefore, we enforce that every generated nonce should be greater than the biggest prime number of all groups. Moreover, to avoid to flood the network, every node runs the scheme of Fig. 2 only once, where it computes the shared key with the *PResp* sender before starting the keying procedure with other neighbor nodes. Finally, we have to resolve the case that two adjacent nodes hold the volatile key at time t, a situation in which both will send PResp to each other, and which would lead to sharing a different key between these two adjacent nodes. In this case, the node with the smaller identifier will update K_{ij} by recomputing the new shared key using the received selected group and its own

P_1	P_2	P_5	P_3	P_6	P_8	P_4	P_7	P_9	P_{10}

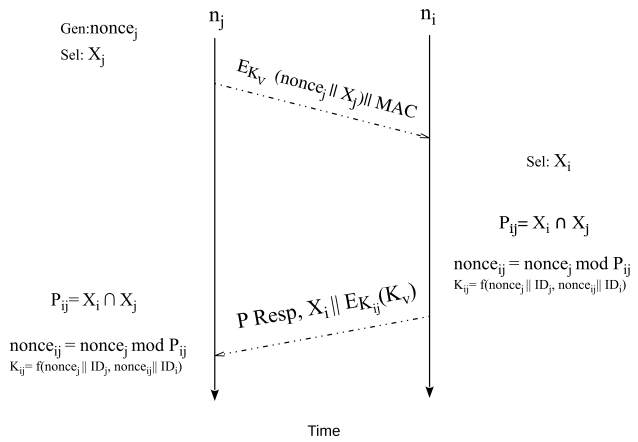

Fig. 2 Key establishment scenario between nodes n_i and n_j

3.2.2 Resource allocation

In comparison with some proposed schemes which use the pool and ring key, we gain an important storage space because, as shown in Sect. 3.1, a nonce (with its encrypted correspondent) and a table of prime numbers are inserted during the initialization step. The size of the prime numbers is not related to the size of the network which allows to provide a scalable network without the need of further storage resource.

In order to avoid the nested loops to find the intersection between two groups and to minimize the size of used groups, we reorder them in a one-row table by inserting each number from Table 2 as soon as it appears a second time using the following equation:

$$index = j + \sum_{k=2}^{k=i-1} (k-1)! \tag{3}$$

where $i = max\{X_i, X_j\}$ and $j = min\{X_i, X_j\}$. Therefore, $P_{ij} = tab[index]$.

Example: Suppose that n_i selects group D and n_j selects group B from Table 2. Then $P_{ij} = P_6$. To use Eq. 3 we reorder Table 2 as follows:

Algorithm 1 Keying scenario for n_i.

Require: *Prime number group*, (P_i, C_i);
Ensure: K_{ij};
1: $isExist \leftarrow true$; //if n_i holds K_v, otherwise $isExist \leftarrow flase$
2: $isK_{ij} \leftarrow true$; //if K_{ij} already calculated
3: $n_i \rightarrow PReq, * : ID_i || C_i$;
4: **if** *Receive* $(PReq_j)$ **then**
5: insert ID_j and C_j into neighborList table;
6: **end if**
7: **if** isExist **then**
8: **for** $j \in$ neighborList **do**
9: $D_{K_v}(C_j)$;
10: *Execute* n_i operations shown in Figure 2;
11: **end for**
12: erase K_v;
13: $isExist \leftarrow false$;
14: **end if**
15: **if** *Receive* $(PResp_j$ & $!isK_{ij})$ **then**
16: *Execute* n_j operations shown in Figure 2;
17: $K_v = D_{K_{ij}}(K_v)$;
18: **for** $k \in$ neighborList **do**
19: $D_{K_v}(C_k)$;
20: *Execute* n_i operations shown in Figure 2;
21: $n_i \rightarrow PResp, ID_k : X_i || E_{K_{ik}}(K_v)$;
22: **end for**
23: erase K_v;
24: **else**
25: **if** *Receive* $(PResp_j$ & $isK_{ij})$ **then**
26: **if** $(ID_i < ID_j)$ **then**
27: *Execute* n_j operations shown in Figure 2;
28: **end if**
29: **end if**
30: **end if**

to obtain:

$$index = 2 + [(2-1)! + (3-1)!] = 5$$

On the other hand, the encryption of the used nonces before the deployment and the sending of *PReq* during the discovering neighbor phase allows to save a lot of energy. Therefore, in the proposed scheme every node sends just to each of its neighbors a response right after the calculation of the shared key between them.

3.3 Key refresh and data freshness

Beyond their limited resource of computation other kinds of attacks such as capture of a sensor node, theft of sensed data, injection of false data, etc. may occur. Hence, the proposed key management scheme should still focus on the two metrics *Efficiency* and *Highlight*, which are pertinent for WSN security concepts. Moreover, main functionalities such as key refresh should be ensured. To refresh the key between two adjacent nodes, the same scheme is used when only the concerned nodes n_i and n_j will participate in the generation of the new pairwise key K'_{ij}. To reach this aim, a counter is used where any node fires key-refresh as soon as this counter reaches its bound or by receiving an authenticated alert of misbehavior within the network. In both cases, the network node n_i starts the same deployment scenario for updating the key with its neighbors by generating a new $nonce'_{ij}$, selecting a new group X'_i and sending a *PReq* to all its neighbors using K_{ij} to encrypt the new generated nonce and the selected group $E_{K_{ij}}(nonce'_i || X'_i)$. To prevent a replay data attack, we assign to each neighbor node n_j of a given node n_i a distinct $counter_{ij}$ which increases its value by one for each message sending and which allows to reach two goals. First, to ensure data freshness and to prevent the attacker to reuse sent data where we use the counter as a parameter in the encryption phase before sending the encrypted data. Second, to use the counter to update the shared key as shown above. An unauthorized attacker eavesdrops on the communication between two neighbor nodes in order to steal information stored in a system by wiretapping (Uma and Padmavathi 2013). Therefore, in passive attack scenarios an attacker can eavesdrop some confidential information when a given node sends the same ciphered data every time, and it is easy for an attacker to eavesdrop on some captured messages, prior to key-refresh, the confidential information without decryption. Therefore, the counter can also be used as a *nonce* because it will be used once for every single different key. This *nonce* is useful for the situation where the sensors send a limited number of data types, for instance the values "True" or "False".

3.4 Addition of a new node

Most of the proposed algorithms for key pre-distribution suffer from the lack of a distributed mechanism for adding new nodes. The *Elliptic Curve Digital Signature Algorithm* (ECDSA) is used in Zhang et al. (2011) to authenticate new nodes without a remarkable impact on energy consumption. However, the proposed solution in Zhang et al. (2011) is not always ready to be implemented in sensor networks because this kind of networks suffer from the lack of a predefined network structure. In fact, asymmetric cryptography solutions are used for WSNs in many approaches by including a noticeable improvement, but they are insufficient, too, due to the time of computation and resource consumption. Our approach focusses on symmetric key solutions which are preferred in terms of energy consumption and computation speed.

A distributed technique for adding new nodes is proposed using symmetric key cryptography. Indeed, when we propose a high security scheme, it will lead us to high resource consumption. Therefore, weighting between security level and resource consumption is preferred. The welcoming key K_w is used to ensure the adding of new legitimate nodes to the deployed network. The scenario of adding new nodes to the deployed network is similar to the scenario presented in Sect. 3.2, where before addition to the network those nodes will be pre-configured in the same way as the deployed nodes. In this case, the nonce and selected group

are encrypted using K_w, which itself will not be inserted into them. After the deployment of the new nodes and unlike to the keying step of Sect. 3.2, the *PReq* and *PResp* run in one way, where the new node broadcasts a *PReq* and every neighbor node responds to it. During the deployment of new nodes the base station securely sends K_w to the sink node to securely share it with every deployed node. Hereafter, the execution of Algorithm 1 is used to integrate the new nodes. The welcoming key will not be stored in the deployed nodes, they only use it if there are new nodes in their radio range. On the other hand, any new node cannot join the network if it is not in the covered range of at least one of the deployed nodes. Thus, a new node not within the covered network range should wait until one or more of its new neighbor nodes are connected to the network.

3.5 Dynamicity and node mobility

Many WSN applications impose the mobility of the connected nodes (Hammoudeh 2016), and ensuring a secure such mobility has been a large research field in recent years (Nack 2010). This paper focuses on unidirectional trust way using the scheme presented in Sect. 3. The third part of this scheme is used to trust the new neighbors by means of the node which has been moving. After the keying step, every node generates its own mobility key and by using this key, sends an encrypted and plain nonce to its neighbors. Therefore, when a node moves to another position it shall ask the new neighbors to share a new key using its own mobility encrypted nonce. Hereafter, the new neighbors look for the corresponding encrypted data of the newly arriving node from the network in order to start the keying scenario shown in Fig. 2. Only the moved node can trust its new neighbors, and the opposite is suitable only if we suppose that the attacker cannot extract the internal data after capturing nodes.

4 Evaluation and analysis

This section analyzes the proposed scheme by focussing on two criteria. First, a real experimentation with 12 sensors is used to verify that the algorithm converges in an acceptable time, i.e., all nodes are able to establish keys with their neighbors in a realistic time. Second, a simulation of a network containing numerous sensors is needed to measure the efficiency and scalability of the proposed technique.

In Nadeem and Javed (2005), Nadeem and Javed observed that the Blowfish algorithm is the fastest encryption algorithm while ensuring robustness with a large key size. Therefore, the Blowfish algorithm is run with a key size of 128 bits to encrypt and decrypt the exchanged data. Furthermore, in Estébanez et al. (2014) authors show that the SuperFastHash algorithm is elegant, extremely fast, providing the convenient function for *MAC* calculation as well as the generation of keys from nonces. Table 4 defines the materials and tools used in this analysis.

4.1 Node to node experimentation

A real experimentation of 12 connected TelosB sensors is used to compute the execution and communication time of the proposed scheme. We placed the volatile key on one sensor which was connected to a laptop using a serial port in order to obtain the time needed for keying. Furthermore, the neighborhood discovery time was fixed to 4 s. We ran our experimentation a 100 times and observed that the average lifetime of a volatile key on a sensor before its deletion was 4.023 ms. The average time it took to propagate the volatile key throughout all sensors was 4.045 ms. In the next section, we will use the real values of the obtained time to simulate a network with hundreds of sensors.

4.2 Stochastic automata network (SAN) modeling and simulation

To evaluate and study the behavior of the proposed *Self-VKS* for a large number of sensors, a SAN for an analytical model is used (Plateau and Atif 1991). Furthermore, we ran the simulation experiment using the CupCarbon simulator tool.

4.2.1 Network modelization

We consider networks in which there are two types of traffic [pair request traffic ($PReq_t$) and pair response request ($PResp_t$)]. Moreover, there is one buffer for each traffic type with finite capacity B_{req} for pair request traffic.

Additionally, we assume that the arrival of pair response traffic follows a Poisson process. The Poisson distribution characterizes discrete events occurring independently of one another in time, as is the case for the occurrence of the volatile key K_v when a given sensor fires $PResp_t$. The duration of a pair request *PReq* is fixed to a predefined time $t_1 = x$. We serve a *PResp* packet if there is no white token in the buffer

Table 4 Software and devices used in our analysis

Tool	Description
TelosB	Implementing some functions to estimate their real execution time
CupCarbon Simulator	Checking the behavior of the proposed scheme by an animated interface

and a black token is not available. The white token means that the keying process is done, whereas the black token means that the sensor holds K_v.

The proposed system can now be described using the defined automata. Before giving the automata, we may describe the different transactions as follows:

- T_{wait} is a transaction which indicates that there is no $PResp_t$ and no white token in the buffer.
- T_{idle} is a transaction which fires if there is $PResp_t$ traffic but no black token in the buffer.
- $T_{process}$ is a transaction which fires if there is $PResp_t$ traffic and an available black token in the buffer.
- T_{done} is a transaction which fires if there is no $PResp_t$ traffic, but a black and white token in the buffer.

As for all transactions, the summation of the routing probability is equal to one as there is only one state at a time t. The system behavior is described by the automata given in Fig. 3, where $t_2 = 0$.

Once the automata built, it is possible to deduce the different parameters. Basically, the proposed scheme is influenced by three parameters: the first is the number of nodes which obtain K_v at the pre-initialization phase, the second indicates the localization of these nodes after the deployment and the third represents the density of the deployed network.

4.2.2 Simulation experiment

We have used a CupCarbon based sensor network simulator (Bounceur 2016; Sharaf et al. 2017) to measure energy consumption overhead. Our key management approach uses pre-defined encrypted nonces to generate keys, or to perform any extensive computation associated with key management. A few encryption/decryption operations are required during keying process, nevertheless their complexity depends on the number and the location of sensors that hold the volatile key at the beginning, and assuming that there exist very low energy consuming algorithms that provide sufficient level of security. In the simulation, the first objective is to observe the behavior of the network, which is difficult to be predicted analytically. In this context, our metrics are the energy consumption per node, within the whole network, and the overhead of security in terms of energy consumption.

The implementation of Algorithm 1 simply says that the proposed scheme does a flooding by starts from sensors that hold the volatile master key K_v, where the communication starts in the form of a tree [distributed BFS (Ueno and Suzumura 2013)]. Basically, when each node has on average k neighbors, the total consumption of the network is equal to $E_m = (k+1) \times N$ (E_m is the value of the consumed energy in terms of the number of messages). Hence, the efficiency of the energy consumption is immediately related to the density, the size of the packet and the cost of the used encryption and decryption techniques. At the level of one node, each sensor will broadcasts the discover message and it will receive at most k messages. This analysis means that the consumed energy average by each sensor is $E_m = k + 1$ Therefore, for the aim to study the energy of consumption it must especially play on the density and not on the number of nodes. In fact, the augmentation of the number of sensors by keeping the same node density leads us to have the same average of the energy consumption. As shown in Fig. 4, the simulations confirm that the average energy consumed according to the size of the network (same density) still always the same. In contrast, the scenario of using a topology more dense by augmenting the number of sensors

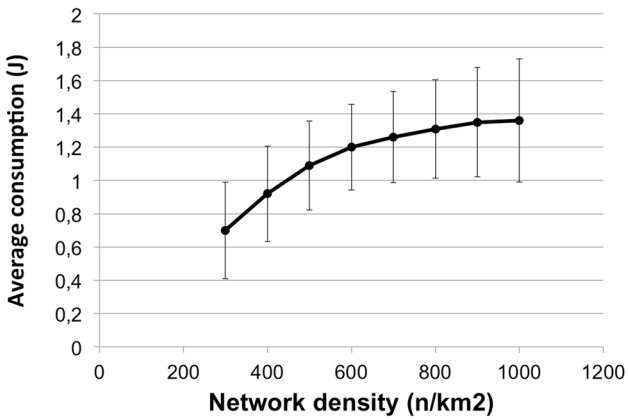

Fig. 4 The impact of the network density on the consumed energy

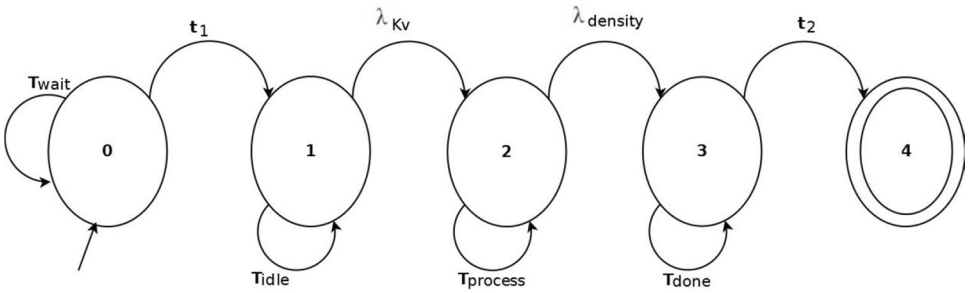

Fig. 3 Automata system modelization

in the same testing area, Fig. 4 shows an acceptable increase of energy consumed by the deployed network. The average energy consumption overhead is very low, in terms of micro Joules (mJ). This overhead consists of communications cost at the sensor due to transmission of confidential information during bootstrapping of the network. The obtained simulation results found that the total energy used in the network for the proposed scheme was 2.73 J, and for the Energy-efficient Distributed Deterministic Key management scheme (*EDDK*) a total energy 4.2*J* was used. It is obvious from the described principle of *EDDK* management scheme in the Related work Section, that the proposed scheme uses significantly less energy. This is because every node in the network is not responsible for generating nonces and encrypting them before starting the keying process.

On the other hand, the system described in the previous section has been implemented on CupCarbon tools to check its feasibility. Figure 5 shows the used topology of 196 nodes deployed in an area of 320×220 m^2, where the obtained execution and communication time values in the previous real experimentation are used in this section. This topology aims: (1) to represent the behavior of the proposed scheme at time t, where at the beginning K_v is inserted into the central node. The yellow circle represents the sensors which hold K_v at time t and the red arrow represents the communication direction between the nodes. (2) to analyze the influence of the fixed parameters on the required time of the proposed keying technique, where in the first scenario, K_v is inserted into the center of the topology (i.e., into the triangular sensor lying in the center), and in the second scenario, K_v is inserted at the corner (i.e., into the triangular sensors lying on the border). The third scenario uses the same number of sensors in another dense topology to deduce the influence of the density on the execution time. As a result, Fig. 6 shows that starting from the central node has an advantage to run the keying process in all directions which minimizes

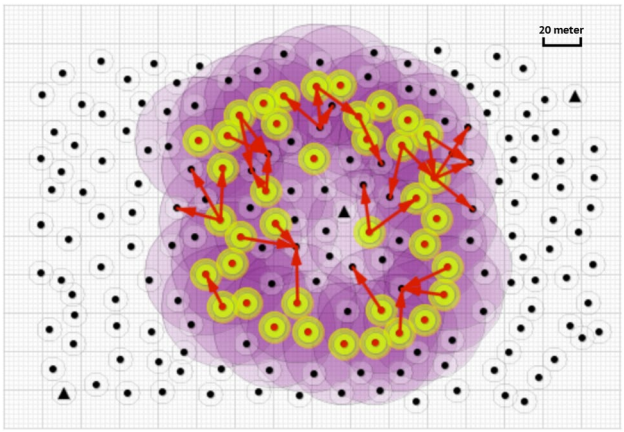

Fig. 5 The propagation of the volatile key

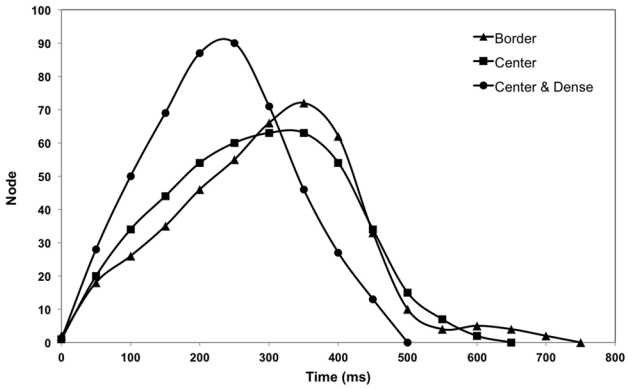

Fig. 6 Number of sensors that hold the volatile key

considerably the time to finish the keying over the whole network. Moreover, the density is relatively noticeable when a dense architecture is used.

In fact, the proposed scheme can be more efficient in case that the number of sensors that hold the volatile key k does not exceed a known threshold at any given time t with the objective of keeping the probability P_t below P, where $P = k/N$ is a predefined threshold. Furthermore, the short execution time is related to the position and the number of sensors that hold the volatile key at the beginning. Figure 6 shows that a dense topology has the shortest execution time, but at $t = 220$ *ms* the number of sensors that hold K_v is big. Also observe that inserting K_v at the corner is the worst way to prevent an attacker. Therefore, the best strategy is to insert the volatile key in the center of the deployed sensors with an acceptable density. In fact, the density parameter draws attention to treat the relation between the density and the living lifetime of K_v. Figure 7 indicates the time required by a sensor with n neighbors. The living time is suitable for $n < 10$ but it is not practicable if $n > 25$ which means that the delay increases with an increasing number of neighbors. Other factors can also cause a noticeable delay for a dense topology such as collusion managing, processing time and so on.

4.3 Security analysis

This section critically reviews security aspects of the proposal and discusses the impact of known outsider attacks, including physical attacks where the attacker can get hold of one or more sensors.

4.3.1 Security validation

It is crucial to start by proving that, under optimal conditions, our scheme is secure (the next few subsections will discuss attack scenarios). This amounts to emphasize the following three points:

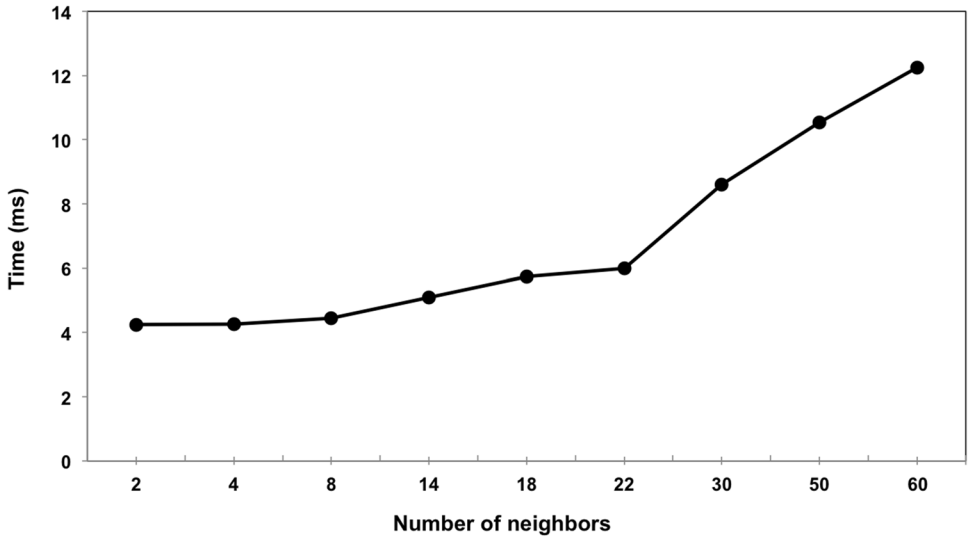

Fig. 7 The lifetime of K_v as a function of the number of neighbors

1. The pairwise key establishment between two nodes is a secure process.
2. The volatile master key, while in transit between two nodes, is secured properly.
3. All nodes will receive K_v.

We start with the first two points: the key establishment is done between two nodes; one node (n_i) holds K_v, the other (n_j) knows a token $\{nonce_j||X_j\}_{K_v}$. The exchange starts with n_i broadcasting a message saying that it holds K_v, to which n_j replies by sending its encrypted token. Once the token is received, n_i extracts the nonce as well as X_j. It finds the common prime number, then uses it to create a new nonce that is unique to i and j. Both nonces are used by n_i to create a new key K_{ij}, and to send K_v (the volatile master key) encrypted using K_{ij}. The security of these messages depends on Blowfish, which is the algorithm used to encrypt all exchanges. Blowfish does not have any known weaknesses. Furthermore, a message authentication code is used to detect modifications of the exchanged message. At the end of this process, (1) a shared key K_{ij} has been securely established using the Blowfish algorithm, with inputs from both n_j and n_i, and (2) K_v has been securely shared with n_j using K_{ij}, thus proving the first two points.

As to the third point, K_v is propagated to a node that did not hold it (n_i) and from a node that held it (n_j) after the secure exchange described above. Let us prove by contradiction, that every node in the network will get hold of K_v after some time. Assume that n_k, which belongs to the same connected network as a holder of the key, n_j, did not receive K_v. If n_j does not have any neighbors, then it is not part of the network, which is in contradiction to the assumption that n_j and n_k are in the same network. So n_j has neighbors, and we may consider one of it, say n_{k_1}. If n_{k_1} received K_v at any point, then it would have transferred it to n_k after key establishment. We conclude that n_{k_1} has never received the key either. We iteratively apply the same logic as we did with n_k to build a chain of nodes n_{k_1}, ..., n_{k_M} that are interconnected, never received K_v, and have at least one path to n_j. The network has a finite number of nodes, say N, and by following our logic, k_M will reach the value of N, which would mean that n_j itself has never received the key, resulting in a contradiction to our assumption that n_j holds K_v. This proves that any node n_k in the same network as at least one node n_j holding K_v will receive it at the end. The rest of this section discusses how effective a variety of WSN attacks are against the proposed scheme.

4.3.2 Physical attack

An attacker might be able to get physical access to a sensor. This situation could be worse if this happens while the volatile master key is in that sensor. It has been proven that extracting keys from sensors is not an instantaneous operation (Casola et al. 2011). Unlike most proposed deterministic schemes, the *Self-VKS* scheme is more efficient against a physical attack, where an attacker has a probability $P = k/N$ to pick a sensor holding K_v.

As a countermeasure to this attack, the volatile master key lifetime can be set to be smaller than the extraction time i.e., if the extraction time is XT and the lifetime of K_v is KL, it is enough to extract K_v such that $XT < KL$. That is, if a sensor falls in the hand of an attacker, then they can extract K_v after XT ms, but that would have already expired before they get hold of it. The same applies to pairwise keys, as these would have been expired with the expiry of K_v.

4.3.3 Hello flood attack

Nodes discover their neighbors by broadcasting an encrypted "hello" message. Sending the "hello" message can be done only within a predefined small window of time, after which all "hello" messages are simply ignored, thus removing the threat of a potential "hello" flood message.

4.3.4 Sybil and replay attack

An attacker can disturb the network by reusing sent data to generate a redundancy of information in order to spend the nodes' energy or by declaring himself as a node using the identifier of other nodes. We prevent this type of attacks by including a sequential number in the nonce used for each K_{ij} as nonce in order to ensure the data freshness. Furthermore, the sequential number is used to update K_{ij} in order to avoid message reuse.

4.3.5 Selective attack

The proposed scheme is susceptible against a selective attack because an attacker can follow the transceiver activity of deployed nodes during the keying. Then, as shown in Fig. 6, the short lifetime of K_v over one node is ensured when an adequate density architecture is used. Hence, we define a density threshold (which is directly related to the number of nodes holding K_v) in order to prevent a selective attack against the proposed solution.

4.4 Comparative study

Historically, several schemes have been proposed for use in WSNs. The continuous appearance of new key management proposals has motivated us to finalize this work by a comparative study. To highlight the features for evaluating wireless sensor network key management schemes such a study should include aspects of computational complexity (computing overhead or processing complexity), communication complexity (overhead), storage complexity (overhead), security strength (resilience), connectivity (connection probability), and scalability (revocation/addition). Hence, the proposed scheme is compared here with the main existing schemes for key management. Table 5 shows the characteristics of each scheme depending on the most indispensable security needs in WSNs. With regard to all these schemes and WSN needs, our scheme responds to almost all encountered key management challenges.

The principal difference between the proposed scheme and the others is that a prepared nonce in the pre-initialization step is used by each node n_i to compute the shared keys K_{ij} where n_j is a neighbor of n_i. The divergence of K_{ij} between two, or more, neighbors is ensured by using the identifier of n_i and n_j. Also, a pseudo-random function and prime numbers are used by n_i and n_j. In fact, this principle offers many advantages compared with most of the existing deterministic schemes. Preparing encrypted nonces in the pre-intialization step instead of encrypting them in the keying step affords a noticeable gain in keying execution time. Additionally, by having the master key being held by n_i or n_j only is enough to start the keying step. Furthermore, the proposed *Self-VKS* is more efficient against physical attacks than (Deng et al. 2005) and (Zhang et al. 2011) because the master key is specially injected into some nodes prior to the deployment. Such strategies oblige attackers not only to capture nodes and recover keys from their memories, but also to constantly eavesdrop the network for gathering some confidential information.

For the aim to evaluate the feasibility and show the efficiency of the proposed scheme, we simulate and compare it with *EDDK* (Zhang et al. 2011), where the same conditions are considered. The principle of Zhang et al. (2011) is a distributed deterministic key management scheme for hierarchical WSNs, in which every sensor node is preloaded with a networkwide shared pseudo-random function and an initial master key. From these parameters and its identifier the node calculates its individual key. Hereafter, during the initialization phase, each node sends a network joining message. The pairwise key and the local cluster key will be obtained using information contained in this message. The information transferred within the network is always authenticated, although not necessarily encrypted. Moreover, the keys are updated after having been used a certain number of times (using a counter). We note that almost all of the previous steps are similar to the proposed work but, unlike to *Self-VKS*, a clustered architecture is used in *EDDK*. Furthermore, for mobile nodes and new nodes joining the network, the keys are established using an elliptic curve digital signature algorithm in *EDDK*. To reach this end, we have implemented the *EDDK* scheme on the CabCarbon simulator, where the topology shown in Fig. 5 has been used. The main parameters that will be compared between these two schemes are the lifetime of the master key and the efficiency in terms of the scalability and density. During the evaluation of the lifetime of the master key, we inject the master key into each deployed sensor in *EDDK* for each scenario prior to starting the simulation. Whereas, in *Self-VKS* the first scenario does not consider the principle of the volatile key by injecting the volatile key into each sensor. By repeating the same scenario we inject for each distinct scenario the volatile key on *Self-VKS* into half, third, quarter and so on of the deployed sensors as shown in Fig. 8. The value t represents the time consumed from the deployment until the latest sensor removes the injected master key which is considered as the lifetime of the master key over the whole network. We observe that our proposition is better than *EDDK* in case that

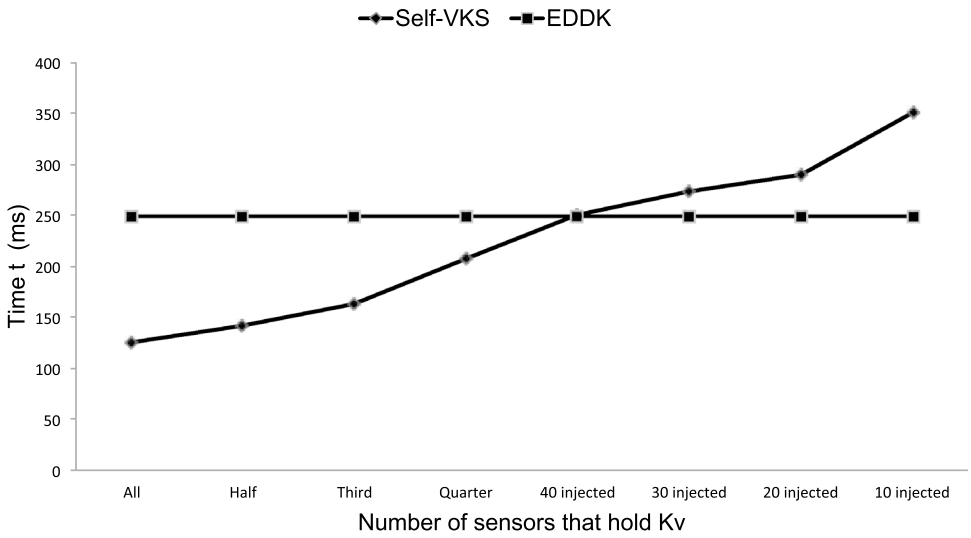

Fig. 8 The lifetime of the master key as a function of the sensors that hold it in the pre-configuration step

Table 5 Comparison of our scheme with the main existing schemes

Protocol	Deter/prob	Centr/decentr	Hiera/flat	Key	Features	Scalability	Energy
Du et al. (2003)	Probabilistic	Decentralized	Flat	Static	λ-secure property	Scalable	Relatively expensive
Gandino et al. (2009)	Probabilistic	Decentralized	Flat	Static	Transitory master key	No	Relatively expensive
Bechkit et al. (2013)	Probabilistic	Decentralized	Flat	Static	Unital design	Moderately	Relatively expensive
Deng et al. (2005)	Deterministic	Decentralized	Flat	Static	Transitory master key	Yes	Moderately expensive
Zhang et al. (2011)	Deterministic	Decentralized	Hierarchical	Dynamic	Threshold Key update	No	Moderately expensive
Rahman and Sampalli (2012)	Deterministic	Decentralized	Hierarchical	Dynamic	Hybrid cryptography	Yes	Relatively expensive
Chen et al. (2011)	Deterministic	Decentralized	Flat	Dynamic	Using distribution key	Moderately	Inexpensive
Al and Yoshigoe (2008)	Deterministic	Centralized	Hierarchical	Dynamic	One time key	Yes	Inexpensive
Our proposition	Deterministic	Decentralized	Flat	Dynamic	Threshold Key update	Yes	Moderately expensive

the master key is injected into all deployed sensors. In fact, this gained time is obtained by encrypting the confidential nonces during the configuration step which speeds the computation during the keying step. Furthermore, the proposed scheme prevents the attacker from capturing any sensor to extract the used master key during the keying step. By injecting the master key only into such sensors the lifetime is still accepted until a certain threshold.

In fact, the proposed *Self-VKS* ensures the main requested functionalities in key management issues such as node addition, key update, network scalability and node mobility, while the density plays an essential role for weighting between the execution time over the whole network and the lifetime of K_v over the sensor nodes. On the other hand, any proposed scheme should be able to scale without compromising the security requirements (Wang et al. 2006). A dense topology should also be supported due to the nature of the sensor nodes which are often densely deployed in a sensor field and which have the capability to collect data and route data back to the admin. Figure 6 shows that the proposed scheme supports dense networks and that it represents a noticeable advantage for minimizing the propagation time t of the volatile key.

5 Conclusion and future research

In this work, we have proposed a scalable and dynamic key management scheme that improves data confidentiality. For the first time and for a deterministic key management, we use a plain together with its encrypted correspondent nonce in order to be able to start from the state where some particular nodes hold the master key at the beginning. The mechanism of the proposed scheme reduces energy consumption incurred by data transmission, because we have minimized the number of messages used to establish the key and introduced many improvements to minimize computation and to save storage space. Moreover, we guarantee major sensor node activities such as key refresh, addition of new nodes and node mobility to be taken into consideration for resource

consumption. A security analysis of the proposed scheme shows its feasibility in terms of time duration to finish the keying as well reactivity against the major known WSN attacks.

In future research, we intend to investigate the feasibility to adapt this principle to the use of asymmetric cryptography systems. Furthermore, with the rapid progression of Internet of Things (IoT) and the emerging Big Data paradigm in the context of Cloud-enabled large-scale sensor networks (Cuzzocrea et al. 2013), a variety of IoT applications are available and quickly gaining attention in our real world. IoT not only has the same security issues as sensor networks, but also its specialties such as privacy issues (Tai et al. 2017). Hence, we may extend our current approach to manage the key technologies of IoT networks because data processing awareness for future IoT is highly desired.

Acknowledgements We thank the anonymous referees for their constructive comments and suggestions, which helped to improve the quality of this paper. The first author acknowledges the support of the PNE program established by the government of Algeria.Thanks to CupCarbon, new proposals can be evaluated and compared with existing solutions in an accurate robust environment. CupCarbon (CupCarbonCupCarbon, A Smart City and IoT WSN Simulator http://www.cupcarbon.com) is an open source Java project and freely available for download.

References

Abomhara M, Zakaria O, Khalifa OO (2010) An overview of video encryption techniques. Int J Comput Theory Eng 2(1):103

Al M, Yoshigoe K (2008) A Secure and Energy-Efficient Key Generation Mechanism for Wireless Sensor Networks. In: PDPTA, pp 587–593

Bechkit W, Challal Y, Bouabdallah A, Tarokh V (2013) A highly scalable key pre-distribution scheme for wireless sensor networks. IEEE Trans Wirel Commun 12(2):948–959

Bounceur A (2016) Cupcarbon: a new platform for designing and simulating smart-city and IOT wireless sensor networks (SCI-WSN). In: Proceedings of the international conference on internet of things and cloud computing. ACM. Cambridge, United Kingdom, p 1

Casola V, De Benedictis A, Drago A, Mazzocca N (2011) Analysis and comparison of security protocols in wireless sensor networks. In: Reliable distributed systems workshops (SRDSW), 2011 30th IEEE symposium. IEEE, Madrid, Spain, pp 52–56

Chen S, Liao X, Shu R, Shen X, Xu X, Zheng X (2011) High performance networking, computing, and communication systems: second international conference, ICHCC 2011, Singapore, 5–6 May 2011. Selected papers. Chapter dynamic key management scheme in wireless sensor networks, Springer, Berlin, pp 381–385

Chen X, Makki K, Yen K, Pissinou N (2009) Sensor network security: a survey. IEEE Commun Surv Tutorials 11(2):52–73

Cuzzocrea A, Fortino G, Rana O (2013) Managing data and processes in cloud-enabled large-scale sensor networks: state-of-the-art and future research directions. In: Cluster, cloud and grid computing (CCGrid), 2013 13th IEEE/ACM international symposium. IEEE, Delft, Netherlands, pp 583–588

Deng J, Hartung C, Han R, Mishra S (2005) A practical study of transitory master key establishment for wireless sensor networks. In: First international conference on security and privacy for emerging areas in communications networks (SECURECOMM'05). Athens, Greece, pp 289–302

Diop A, Qi Y, Wang Q, Hussain S (2013) An advanced survey on secure energy-efficient hierarchical routing protocols in wireless sensor networks. arXiv preprint arXiv:1306.4595

Du W, Deng J, Han Y S, Varshney PK (2003) A pairwise key pre-distribution scheme for wireless sensor networks. In: Proceedings of the 10th ACM conference on computer and communications security, CCS '03, ACM, New York, pp 42–51

Estébanez C, Saez Y, Recio G, Isasi P (2014) Performance of the most common non-cryptographic hash functions. Softw Pract Exp 44(6):681–698

Gandino F, Montrucchio B, Rebaudengo M (2009) Random key pre-distribution with transitory master key for wireless sensor networks. In: Proceedings of the 5th international student workshop on emerging networking experiments and technologies. ACM, Rome, Italy, pp 27–28

Gao Q, Ma W, Li X (2017) A key predistribution scheme based on mixed-level orthogonal arrays. Adhoc & Sens Wireless Netw 37(1–4):53–69

Hammoudeh M (2016) Putting the lab on the map: a wireless sensor network system for border security and surveillance. In: Proceedings of the international conference on internet of things and cloud computing, ACM, Cambridge, United Kingdom, p 4

He X, Niedermeier M, De Meer H (2013) Dynamic key management in wireless sensor networks: a survey. J Netw Comput Appl 36(2):611–622

Huang Q, Cukier J, Kobayashi H, Liu B, Zhang J (2003) Fast authenticated key establishment protocols for self-organizing sensor networks. In: Proceedings of the 2nd ACM international conference on wireless sensor networks and applications. ACM, San Diego, CA, USA, pp 141–150

Jr MAS, Barreto PS, Carvalho CBMTC (2010) A survey on key management mechanisms for distributed wireless sensor networks. Comput Netw 54:2591–2612

Karlof C, Wagner D (2003) Secure routing in wireless sensor networks: attacks and countermeasures. Adhoc Netw 1(2):293–315

Kavitha T, Sridharan D (2010) Security vulnerabilities in wireless sensor networks: a survey. J Inf Assur Secur 5(1):31–44

Laouid A, Messai ML, Bounceur A., Euler R, Dahmani A, Tari A (2016) A dynamic and distributed key management scheme for wireless sensor networks. In: Proceedings of the international conference on internet of things and cloud computing. ACM. Cambridge, United Kingdom, p 70

Mehmood A, Umar MM, Song H (2017) ICMDS: secure inter-cluster multiple-key distribution scheme for wireless sensor networks. Ad Hoc Netw 55:97–106

Nack F (2010) An overview on wireless sensor networks. Institute of Computer Science (ICS), Freie Universität Berlin

Nadeem A, Javed MY (2005) A performance comparison of data encryption algorithms. In: Information and communication technologies, ICICT 2005. First international conference. IEEE, Karachi, Pakistan, pp 84–89

Padmavathi DG, Shanmugapriya M et al (2009) A survey of attacks, security mechanisms and challenges in wireless sensor networks. arXiv:0909.0576

Plateau B, Atif K (1991) Stochastic automata network of modeling parallel systems. IEEE Trans Softw Eng 17(10):1093–1108

Rahman M, Sampalli S (2012) A hybrid key management protocol for wireless sensor networks. In: 2012 IEEE 11th international conference on trust, security and privacy in computing and communications. Liverpool, UK, pp 769–776

Rahman M, Sampalli S, Hussain S (2010) A robust pair-wise and group key management protocol for wireless sensor network. In: 2010 IEEE Globecom workshops

Reegan AS, Baburaj E (2013) Key management schemes in wireless sensor networks: a survey. In: Circuits, power and computing technologies (ICCPCT), 2013 international conference. IEEE, Nagercoil, India, pp 813–820

Rogaway P (2004) Nonce-based symmetric encryption. In: Fast software encryption, Springer, Berlin, pp 348–358

Sahingoz OK (2013) Large scale wireless sensor networks with multi-level dynamic key management scheme. J Syst Archit 59(9):801–807

Sharaf M, Abughazala M, Muccini H, Abusair M (2017) An architecture framework for modelling and simulation of situational-aware cyber-physical systems. In: European conference on software architecture, Springer, Berlin, pp 95–111

Shi J, Wan J, Yan H, Suo H (2011) A survey of cyber-physical systems. In: Wireless communications and signal processing (WCSP), 2011 international conference. IEEE, pp 1–6

Simplício MA, Barreto PS, Margi CB, Carvalho TC (2010) A survey on key management mechanisms for distributed wireless sensor networks. Comput Netw 54(15):2591–2612

Tai WL, Chang YF, Li WH (2017) An IOT notion–based authentication and key agreement scheme ensuring user anonymity for heterogeneous ad hoc wireless sensor networks. J Inf Secur Appl 34:133–141

Ueno K, Suzumura T (2013) Parallel distributed breadth first search on GPU. In: High performance computing (HiPC), 2013 20th international conference. IEEE, Bangalore, India, pp 314–323

Uma M, Padmavathi G (2013) A survey on various cyber attacks and their classification. IJ Netw Secur 15(5):390–396

Wang Y, Attebury G, Ramamurthy B (2006) A survey of security issues in wireless sensor networks. Comput Sci Eng

Xiao Y, Rayi VK, Sun B, Du X, Hu F, Galloway M (2007) A survey of key management schemes in wireless sensor networks. Comput Commun 30(11):2314–2341

Yang Q, Zhu X, Fu H, Che X (2015) Survey of security technologies on wireless sensor networks. J Sens 2015:9

Zhang J, Varadharajan V (2010) Wireless sensor network key management survey and taxonomy. J Netw Comput Appl 33(2):63–75

Zhang X, He J, Wei Q (2011) Eddk: energy-efficient distributed deterministic key management for wireless sensor networks. EURASIP J Wirel Commun Netw 12:1–11