Wireless Netw
DOI 10.1007/s11276-012-0450-3

Low-cost group rekeying for unattended wireless sensor networks

Juan Hernández-Serrano · Juan Vera-del-Campo · Josep Pegueroles · Carlos Gañán

Abstract Wireless sensor networks (WSNs) are made up of large groups of nodes that perform distributed monitoring services. Since sensor measurements are often sensitive data acquired in hostile environments, securing WSN becomes mandatory. However, WSNs consists of low-end devices and frequently preclude the presence of a centralized security manager. Therefore, achieving security is even more challenging. State-of-the-art proposals rely on: (1) attended and centralized security systems; or (2) establishing initial keys without taking into account how to efficiently manage rekeying. In this paper we present a scalable group key management proposal for unattended WSNs that is designed to reduce the rekeying cost when the group membership changes.

Keywords WSN · GKM · Rekeying · Security · Sensors

1 Introduction

One of the main characteristics of *wireless sensor networks* (WSNs) is the cooperation of the participants in order to collect and provide useful sensed information of a certain scenario. However, the wireless nature of WSNs makes the information flow easily accessible by anyone in the vicinity. Therefore, it is necessary to establish a secure communication channel between sensor nodes where no attacker can eavesdrop, modify, replay, or inject messages. This is what is called group security, which is targeted to provide group privacy, since data should be protected just from outsiders, and group authentication, since the only sources of communication should be the members of the group. In order to achieve this goal, every member knows a set of keys that are usually classified as: keys shared by two nodes or pairwise keys, and keys shared between group nodes or groupwise keys. Pairwise keys allow secure routing by hop-by-hop encryption and authentication, and provide easy isolation of a kidnapped member since compromised keys are just not used anymore. On the other hand, groupwise keys allow secure routing without the need of costly hop-by-hop encryption but providing hop-by-hop authentication and integrity (checking message authentication codes at every link). However groupwise keys are not resilient against node kidnapping and thus must be updated whenever a member is compromised. Nevertheless both types of keys are part of a vicious circle if asymmetric cryptography is not used: in order to securely agree on pairwise keys, nodes need a preshared secret that it is often the groupwise key; and in order to securely update a groupwise key, secure communications must be provided often by means of uncompromised pairwise keys.

Group key management (GKM) studies the generation and updating of the aforementioned keying material during the entire group life [34]. GKM thus guarantees that only the current group members can authenticate and understand or decrypt messages within the group. The main challenge of GKM implies the securely update and distribution of the

J. Hernández-Serrano (✉) · J. Vera-del-Campo · J. Pegueroles · C. Gañán
Department of Telematics, Universitat Politècnica de Catalunya (UPC), Jordi Girona 1-3, 08034 Barcelona, Spain
e-mail: jserrano@entel.upc.edu

J. Vera-del-Campo
e-mail: juanvi@entel.upc.edu

J. Pegueroles
e-mail: josep.pegueroles@entel.upc.edu

C. Gañán
e-mail: carlos.ganan@entel.upc.edu

Published online: 25 May 2012

keying material whenever a new member joins the group or a member leaves or is expelled from the group. Obviously, in order to expel a member, a previous detection of misbehavior, malfunction or impersonation is needed. This is studied in another branch of research, complementary to the study of GKM, mostly focused on *intrusion detection systems* (IDS) [16].

Since WSNs are devoted to provide sensed data, detection of misbehaving or compromised nodes in WSNs is mainly the case of outlier detection [38], often also known as anomaly detection or deviation detection. Outliers can be defined as "those measurements that significantly deviate from the normal pattern of sensed data" [7]. The identification of outlier sources in WSNs have been addressed in several research topics, being the main ones fault detection [8, 36], event detection [20, 25] and intrusion detection [9, 37]. To the best of our knowledge, there is still room for improvement and additional work. However, its study is out of the scope of this contribution.

We are certain that the application of known GKM techniques to WSN can secure the exchanged critical data while incurring a low impact on network performance. However, many WSNs, such as the *unattended WSNs* (UWSNs), preclude the fixed presence of a centralized data-collection point, which usually manages group security [2, 14, 26]. Within this unattended nature, a secure distributed cooperation framework for sharing data, resources and/or services between the UWSN members becomes mandatory. Distributively guarantying GKM for UWSNs thus arises as a very challenging task.

In this paper we present a distributed, unattended, self-organized GKM protocol specifically suited to low-end devices. One of the main targets of our proposal is to reduce the added energy consumption due to securing the group. This is why the presented protocol is designed to minimize the computational costs and, especially, the necessary transmitted messages, which in fact incur the highest energy costs [24].

The rest of the paper is organized as follows. Prior to detailing our protocol, in Sect. 2 we discuss the ongoing work on securing UWSN and the motivation of our proposal. Then, in Sect. 3 we describe the process for distributively creating a secure group. How to secure and efficiently manage the group dynamics due to losses, leavings and joinings is detailed in Sects. 4 and 5. Next, in Sect. 6, we evaluate both analytically and by simulation our protocol at the application layer. Section 7 is an exercise that shows how to apply the previous results in a realistic physical-layer scenario. In Sect. 8, we present a comparison with similar state-of-the-art proposals. Finally, in Sect. 9, we summarize the conclusions and future work related to these topics.

2 Related work and motivation

Most of current distributed GKM proposals in the literature for UWSNs focus on generating the necessary group keys from a set of predistributed keying material [5, 6, 11, 12, 21, 32, 39]. In all these proposals the process of securely agreeing on a common group key is either a decentralized heavy process, or relies on a powerful central base station, or it is not provided at all. With regard to UWSN, the first is the only option. Consequently, an efficient rekeying scheme is mandatory to avoid a costly repetition of the initial group creation every time the membership changes. This fact is especially unsuitable for common sensor nodes which have limited computational resources and battery; not to mention how worse the situation becomes when the group is very large. Therefore, distributed GKM techniques focusing on rekeying efficiency with an autonomous ability to regenerate the group (unattended creation/updating of the group keying material) must be provided.

Considering that most of the energy consumption in WSNs owes to the transmission of a message over a wireless channel (reaching up to 3 orders of magnitude [24] more than processing), any GKM proposal for WSNs must be designed to minimize the number of messages that have to be transmitted for rekeying. The most successful proposals for reducing the rekeying problem order are, even nowadays, based on the old well-studied logical tree hierarchies of *key-encryption keys* (KEKs) [14, 30, 34]. The simplest method for providing group security is merely based on the use of a groupwise key shared by all the group members, so-called the networkwise key or group key or session encryption key (SEK). This key allows every group member to: (1) send encrypted data; (2) decrypt received data, and (3) authenticate itself as a group member since the knowledge of the session key guarantees that it belongs to the group. However, in order to securely update the SEK when the group membership changes, some other keys are necessary, so-called KEKs. These KEKs are organized into logical trees that actually improve the rekeying efficiency in terms of bandwidth and latency. In this kind of methods, a key server builds a KEK tree only known to it and assigns a subset of the tree KEKs to each of the members of the group. This subset is made of the keys that correspond to the tree nodes in the path from the leaves—where the members are—to the tree root. When a member leaves or joins a group, only the KEKs belonging to that member need to be changed. Then, new keys are delivered to the remaining members and the tree is reconstructed using the unchanged keys. In short, the cost of rekeying is reduced from $O(N)$ to $O(L)$, with N the number of members and L the depth of the tree.

As the number of necessary messages for rekeying depends on L, tree-based algorithms are more efficient when L gets its minimum value. And this is the case of a balanced tree (all leaves at the same level or at most between two adjacent levels) where the depth of the tree is exactly $\log_x N$ with x the maximum number of nodes hanging from a given node (commonly $x = 2$ which leads to binary trees). The need for balanced trees has been well studied and many protocols [22, 29] have dealt with it. However, none of the previous proposals fulfills the requirement of unattended operation since they rely on fixed always available centralized security servers. Therefore, providing unattended tree-based GKM proposals is challenging.

In [15] a GKM protocol that used logical key trees constructed from the contributions of all network members is presented. This proposal fulfills the two main design objectives of GKM used in wireless ad hoc networks: (1) it minimizes the cost of updating keying material, and (2) it resolves the first objective by implementing a distributed GKM system based on peer relations (peer-to-peer, P2P) in which every group member contributes. However, [15] does not take care of very low-end devices, such as sensor nodes, in which every step forward in resource savings, no matter how small, is of paramount importance; in which even the necessary rekeying when the membership changes could lead to exhausting network nodes. The new proposal described here is intended for wireless ad hoc networks with low-end devices, such as WSNs. Therefore, besides minimizing the required messages for rekeying, we use symmetric cryptography instead of asymmetric cryptography to reduce the cost of the new protocol and provide new "low-cost" methods for authentication and pairwise agreement.

In short, below we present a distributed, unattended, self-organized protocol that, to the best of our knowledge, is the only one allowing to secure a large group of nodes in a distributed, unattended, self-organizing manner, and which is specifically suitable for devices with resource constraints, such as sensor nodes, since it minimizes both the necessary transmitted messages, as it is based on logical trees of keys, and the computational costs, as only makes use of symmetric cryptography. Since the use of a GKM protocol allows to maintain group security, it is actually a basic tenet for guaranteeing confidentiality and group authentication. Hence, the proposed protocol can securely distribute the necessary keying material for protecting the WSN against the following attacks by an external attacker: eavesdropping (messages are encrypted), modification (message authentication codes allow to check integrity of received messages), replay (as long as the message authentication codes use some kind of timestamps), and/or injection of messages (no valid packets can be generated without knowing the current keys).

To the extent of our research, there are three proposals in the state of the art [10, 15, 19] that apply logical trees of KEKs to implement a GKM protocol in a self-organizing scenario. The main differences between them and our approach are:

1. In [15, 19], asymmetric cryptography is used for authentication and key agreement. On the contrary, our proposal only uses symmetric cryptography, which requires less computational power.
2. In [10], the rekeying can be an unattended operation, but the creation of the secure group needs to be managed by a centralized powerful node. On the other hand, our proposal extends the unattended operation even to the creation of the secure group.

3 Initial deployment of the secure group

Before the group is deployed, the base station (or gateway) offline generates an initial group key K_g and decides the value of parameter k that, as we will detail in Sect. 5, represents the maximum number of new members that can be registered in the group after the deployment. Then, the base station preloads every node/member with a groupwise symmetrical key or SEK that is the result of applying k one-way hash functions to the initial group key K_g as in (1) with H the selected hash function.

$$K_g^k = \begin{cases} H(K_g^{k-1}) & k > 1 \\ H(K_g) & k = 0 \end{cases} \tag{1}$$

As aforementioned, a secure UWSN can run with just the use of a SEK for group-secure routing. Nevertheless, when a node is kidnapped, the SEK is compromised, and, in order to isolate such kidnapped node and guarantying secure rekeying, pairwise keys are also needed for hop-by-hop encryption. As a result, besides providing a group key management scheme (see Sect. 3.2), every node must agree a pairwise key with all the nodes with whom it has direct visibility at the link layer (see Sect. 3.1). From now on we will call these nodes as neighbors.

3.1 Agreement of pairwise keys

Pairwise key agreement schemes either use a preloaded shared secret or require the collaboration of a trusted third party (e.g., SNEP [31]). Our proposal is designed to unattended networks that consequently do not have access to external trusted third parties; therefore, it relies on the preloaded shared secret approach. This shared secret is the current SEK, which, at a first stage, is preloaded into every group member (K_g^k), and then updated to the new agreed SEKs as detailed in Sect. 3.2.

The process of securely agreeing a pairwise key is as follows. Each member u computes a key $K^u = f_{\mathrm{SEK}}(ID_u, \tau_u)$ before the secure group is established; where f is a one-way cryptographic function based on key the initial SEK, ID_u is the node identifier, and τ_u is a timestamp that guarantees key freshness. Then, the pairwise key agreement between member A an member B is: (1) A sends ID_A, τ_A, K^A to B and vice-versa B sends ID_B, τ_B, K^B to A; (2) both A and B authenticate the received data by using the shared SEK in order to check K^A and K^B respectively; (3) A and B agree a shared key as $K_{A-B} = f_{K^B}(ID_A)$.

Generation of pairwise keys is needed for hop-by-hop encryption whenever a sensor node is kidnapped, but it will not be necessary at all for the rest of the initial creation phase. Once the group has been established (see Sect. 3.2), the SEK is updated to the new updated group key and thus the pairwise keys cannot be compromised.

3.2 Creating the logical tree of KEKs

The logical KEK tree used in this proposal is generated from cooperation of the sensor nodes and thus it is created in a bottom-up manner: every KEK is generated as a function of the KEKs associated to the underlying nodes. Consequently, we define the key associated to a node (i, j) as in expression (2). With i, the depth (row) in the tree; j, the column position within row i (see the logical tree of KEKs in Fig. 1); L, the depth of the tree; $g()$, an unidirectional function that it is used to blind the keys; $f()$, a combinatorial function; and *random*(), a random function used to create the keys associated to the leaf nodes (the group members).

$$K_{i,j} = f(g(K_{i+1,2j}), g(K_{i+1,2j+1})) \quad \begin{cases} 0 \le i < L \\ 0 \le j < 2^i \end{cases} \tag{2}$$

$$K_{i,j} = random(i,j) \quad \begin{cases} i = L \\ 0 \le j < 2^i \end{cases} \tag{3}$$

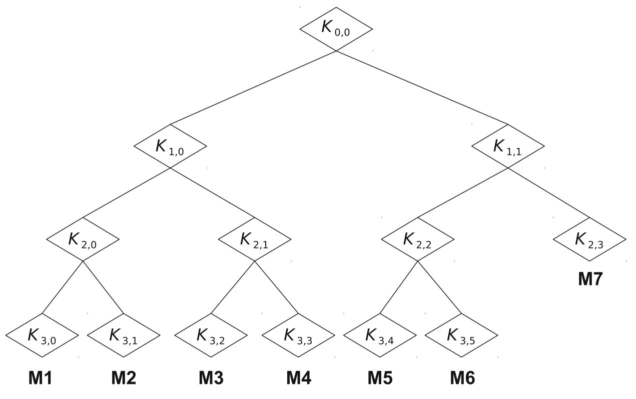


Fig. 1 Logical tree of KEKs

As in every tree based GKM scheme, each member must be able to calculate all of the tree keys from the leaf node where it is placed to the root of the tree. Consequently, every member must know both its key and the blinded keys of every sibling node in its path to the root. For example, consider the tree in Fig. 1, M1 must know its own KEK $K_{3,0}$ and the blinded KEKs $g(K_{3,1})$, $g(K_{2,1})$ and $g(K_{1,1})$ in order to calculate its necessary tree KEKs as: $K_{2,0} = f(g(K_{3,0}), g(K_{3,1}))$, $K_{1,0} = f(g(K_{2,0}), g(K_{2,1}))$ and $K_{0,0} = f(g(K_{1,0}), g(K_{1,1}))$.

In general terms, the creation of the secure group works as follows:

1. Every sensor node broadcasts a unique identifier to its neighbors that from now on we will call it weight[1]. Once all the weights have been sent, each node knows its one-hop neighbors.
2. The nodes start the association by pairs. After association, a key is created and shared by both incumbents, and the node with highest weight becomes the leader of the pair. At this moment every pair of nodes forms a logical tree of KEKs with depth 1. The weight of the formed trees is the sum of its members' weights.
3. Pair leaders start to associate forming groups of four nodes. Consequently, a new key is created and shared by all of the four members of the group and the member with highest weight is the new leader. At this moment every set of four nodes forms a logical tree of KEKs with depth 2.
4. Next, (leaders of) groups of four nodes start the association into sets of eight nodes with a new shared key (a logical tree of KEKs with depth 3). The association process is repeated time and again until an only logical binary tree of keys is created.
5. Once the group has been established, the SEK is updated to $K_{0,0}$ and thus K_g^k is no longer needed but to authenticate new member joinings as detailed in Sect. 5.

The leaders of every subset or group formed during the tree creation, *group leaders* (GLs) from now on, manage the association process, which is just the generation of a shared key by means of interchanging the blinded keys of both incumbents as in expression (2).

In order to guarantee that the process leads to an almost balanced tree of KEKs, the protocol tends to avoid association between trees of very different depths. Let us denote a_i as the association priority of the group leaded by member i. Then, a_i is a 3-tuple made up of (depth of tree leaded by i, weight of tree leaded by i, weight of i).

[1] The reason of calling it weight is to leave an open door for future research on how different unique weight assignments affect the protocol performance. Anyhow, in the simplest implementation, the weight is just a random unique identifier.

Elements a_i can be ordered in a strict ascendant fashion following the next criterion: $a_i > a_j \; \forall \; i \neq j$ if and only if the tree depth of group leaded by i is less than the tree depth of group leaded by j; and, in case of tie, the tree weight of group leaded by i is greater than the tree weight of group leaded by j; and, in case of tie, the weight of i is greater than weight of j.

Consequently, the GL of group with highest association priority within its neighboring groups will try to associate with the GL of its neighboring group with the next highest association priority; the rest of neighboring GLs just wait to be the ones with highest priority. Once associated, the new created group has less association priority since the depth of its tree has been increased in one level. As a result, other neighboring trees, which would be waiting, start to associate.

One of the most important aspects to assess is the protocol convergence. That is to say, the protocol has no deadlocks. As already explained, every GL is either associating or awaiting the action of another GL; if all GLs are in a waiting state, deadlock occurs.

Proposition 1 *If the weight of a node is a unique identifier, then the protocol has no deadlocks.*

Proof The uniqueness property of the assigned weights (unique identifiers) implies also uniqueness of a_i values since $a_i = a_j \leftrightarrow i = j$. Consequently, we can create a totally strict ordered set $\mathbb{A}$ of the different values of a_i with $i \in [1, M]$, $M \leq N$ the number of neighboring GLs, and N the total number of sensor nodes deployed. The order criterion is the relation '<' or 'less than'; then $a_i < a_{i+1} \; \forall \; i < M$. □

Deadlock occurs if all GLs are in a waiting state, that is to say that all GLs have a neighboring GL with highest association priority, or that for every $a_i \in \mathbb{A}$ at least another $a_j \in \mathbb{A}$ with $i \neq j$ exists such that $a_i < a_j$. However, we cannot find any $a_i > a_M$ since, because of the transitive property of the '>' operator, $a_M > a_i \; \forall \; i < M$. As a result, deadlock cannot occur.

For a more in depth understanding of the protocol, Fig. 2 depicts the flowchart of every node's operation during the creation of the secure group. The three types of messages used by the protocol are: *report_msg*, used to send member/tree information to the neighboring leaders; *req_msg*, used to request association to a neighboring leader; *ack_msg*, used to acknowledge a requested association; and *neighbor_msg*, that the nodes use to share their neighboring groups with their GL.

3.2.1 *Example of operation*

For the sake of clarity, we next summarize the overall behavior of the protocol with an example. Figures 3, 4 and 5 show an example of creation of a secure group in three iterations. The numbers represent each member identifier/weight; and the lines between members represent direct visibility at the link layer. [A, B, C] represents the group formed by members A, B and C. $K^u_{i,j}$ is the key at depth i and position j in the tree of the group leaded by u.

3.2.1.1 First iteration First of all, each node publishes its weight to its neighbors by sending a local (one-hop) broadcast message. After that, every node knows its one-hop neighbors. Now, the first iteration of the protocol proceeds as follows (see Fig. 3). Initially, every node form a group of an only member with association priority as the 3-tuple (tree depth = 0,tree weight = member weight, GL weight = member weight). Therefore, at the beginning, the groups with highest association priority within their neighborhoods are [3], [6] and [7]. The remaining groups wait for them: [1] waits for [7]; [2] for [4]; [4] for [6]; and [5] for [7].

[3] requests association with [1], [6] with [5], and [7] with [5]. [5] confirms the request received from [7] to associate in the group [7,5] (tree depth = 1, tree weight = 12, GL weight = 7). Both report the action to their current neighboring groups: [5] reports to [1], [6] and [4]; and [7] to [1].

[6] is now the group with highest association priority in its neighborhood and requests association with the next one, [4]. [4] acknowledges and both are associated in the group [6, 4] (tree depth = 1, tree weight = 10, GL weight = 6). Both report it to their neighboring groups: [6] reports to [7, 5]; and [4] to [7, 5], [2] and [1].

Finally, the neighboring group of [1] with highest association priority is [3], so [1] acknowledges the previous request, both associate in the group [3, 1] (tree depth = 1, tree weight = 4, GL weight = 3) and report the action to their current neighboring groups: [1] reports to [6, 4] and [7, 5].

3.2.1.2 Second iteration At this moment, the groups with highest association priority in their neighborhoods are [2] and [7, 5] (see Fig. 4). The rest waits: [6, 4] waits for [2]; and [3, 1] for [7, 5].

[2] and [7, 5] request association to [6, 4]. [6, 4] acknowledges to [2], both form a new group [6, 4, 2] (tree depth = 2, tree weight = 12, GL weight = 6) and report to their neighboring groups [7, 5] and [3, 1].

Now [7, 5] requests association with [3, 1], which is its current neighboring group with highest association priority. [3, 1] acknowledges, both form a new group [7, 5, 3, 1] (tree depth = 2, tree weight = 16, GL weight = 7) and report it to their neighboring group [6, 4, 2].

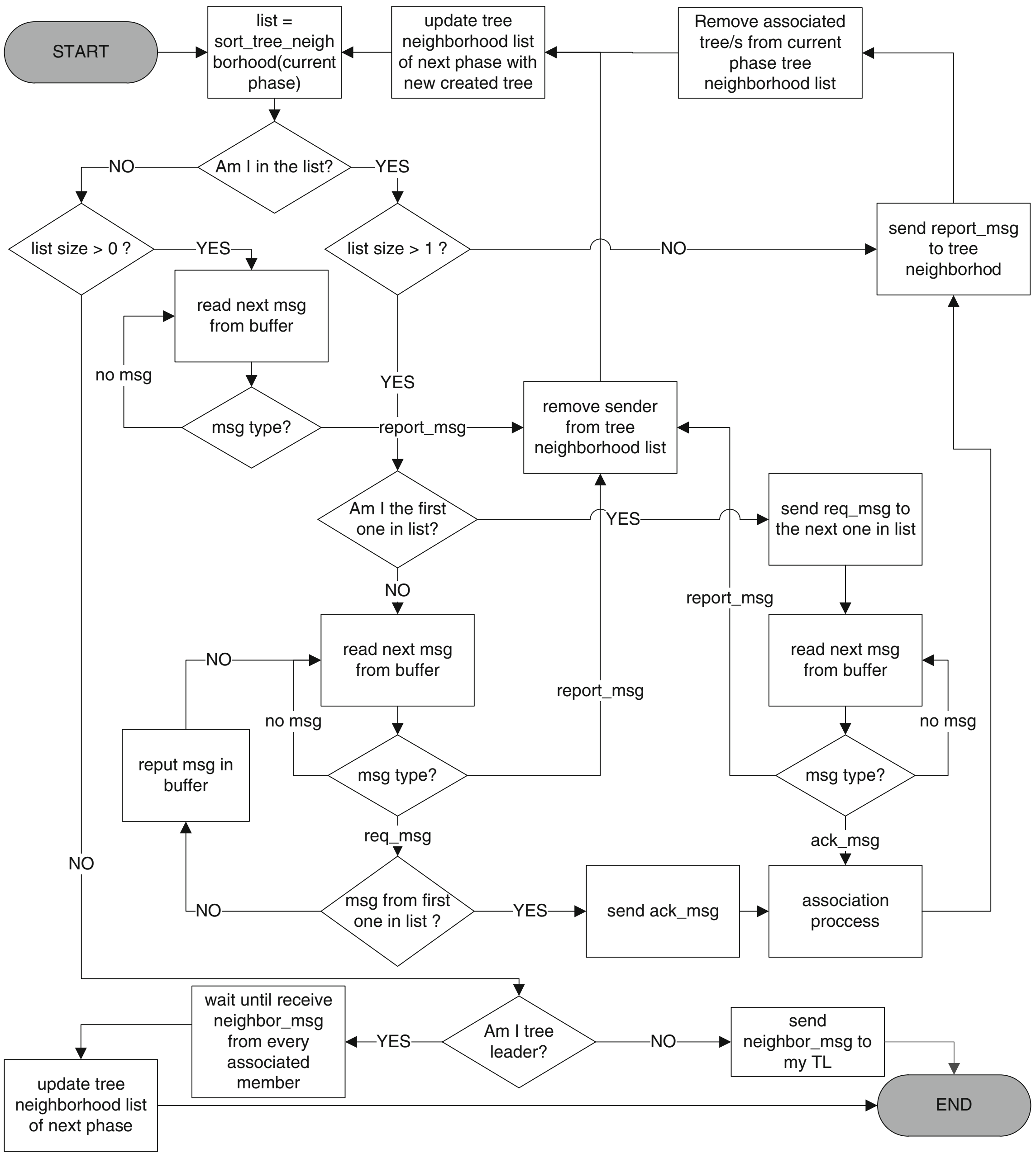

START
list = sort_tree_neighborhood(current phase)
update tree neighborhood list of next phase with new created tree
Remove associated tree/s from current phase tree neighborhood list
Am I in the list?
NO
YES
list size > 0 ?
YES
list size > 1 ?
NO
send report_msg to tree neighborhod
read next msg from buffer
no msg
YES
msg type?
report_msg
remove sender from tree neighborhood list
Am I the first one in list?
YES
send req_msg to the next one in list
NO
report_msg
read next msg from buffer
NO
read next msg from buffer
report_msg
no msg
no msg
reput msg in buffer
msg type?
msg type?
req_msg
ack_msg
NO
NO
msg from first one in list ?
YES
send ack_msg
association proccess
wait until receive neighbor_msg from every associated member
YES
Am I tree leader?
NO
send neighbor_msg to my TL
update tree neighborhood list of next phase
END

Fig. 2 Flowchart of a node during the creation of the secure group

3.2.1.3 Third iteration Finally (see Fig. 5), in iteration 3, [7, 5, 3, 1] acknowledges association to [6, 4, 2], a unique logical KEK tree is generated and the protocol is over.

Now the group of sensors has created a new logical binary tree of KEKs, the previous SEK is discarded and the new one is $K_{0,0}$.

4 Managing member leavings/losses/expulsions

When a sensor node leaves the group, in order to guarantee backward secrecy, the SEK (and consequently the compromised tree KEKs) must be updated. When the member voluntarily leaves (due to e.g. battery exhaustion), it notifies its leave to its sibling (the node hanging from the same node of the tree) or, if it does not exist, the leader (GL) of the sub-tree hanging from the first sibling node of the leaving member in a leaf-root way. The node in charge of initiating the rekeying process is called *rekeying master* (RM). If the member just crashes or it is compromised, any member detecting it broadcasts its leave in order to notify it to its RM. As aforementioned, detecting misbehavior, malfunction or impersonation of a node is another branch of research, complementary to the study of GKM, which is out

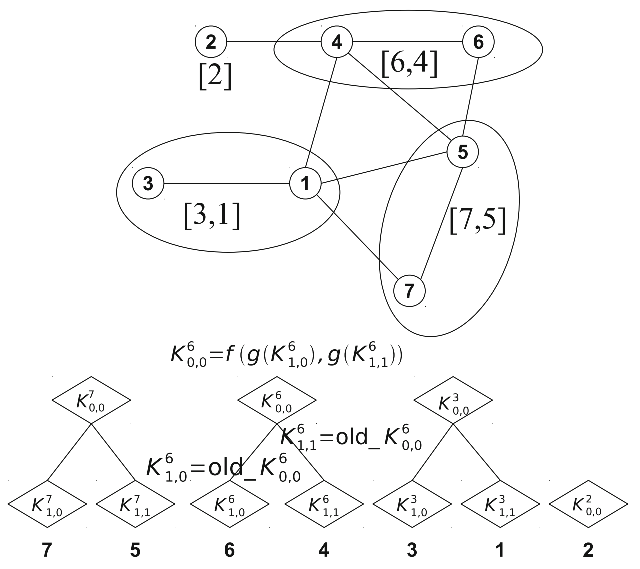

[2]
[6,4]
[3,1]
[7,5]
$K^6_{0,0}=f(g(K^6_{1,0}),g(K^6_{1,1}))$
$K^6_{1,1}$=old_$K^6_{0,0}$
$K^6_{1,0}$=old_$K^6_{0,0}$
7
5
6
4
3
1
2

Fig. 3 Creation of a secure group: 1st iteration

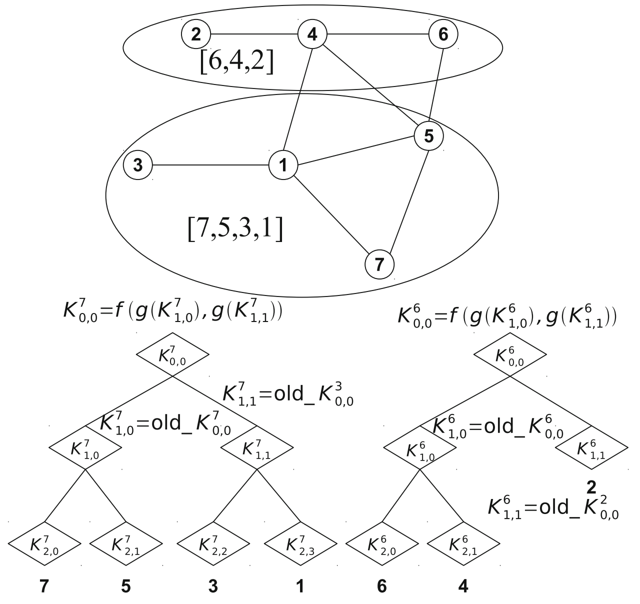

[6,4,2]
[7,5,3,1]
$K^7_{0,0}=f(g(K^7_{1,0}),g(K^7_{1,1}))$
$K^6_{0,0}=f(g(K^6_{1,0}),g(K^6_{1,1}))$
$K^7_{1,1}$=old_$K^3_{0,0}$
$K^7_{1,0}$=old_$K^7_{0,0}$
$K^6_{1,0}$=old_$K^6_{0,0}$
$K^6_{1,1}$=old_$K^2_{0,0}$
7
5
3
1
6
4
2

Fig. 4 Creation of a secure group: 2nd iteration

of the scope of this protocol. GKM provides with methods to safely expel the member once it has been detected.

Once the leaving process is started, the RM deletes every spare node and consequently moves itself or its subtree. Then, the RM updates its key and regenerates all the keys in its path to the root. Next, the RM securely communicates with every leader (GL) of the subtrees hanging from the sibling nodes of the leaving member by means of hop-by-hop encryption with the pre-established pairwise keys. After that, the RM sends to the GLs the needed new blinded keys to reconstruct the tree. After receiving the blinded keys, each involved GL sends them to the rest of members in its subtree

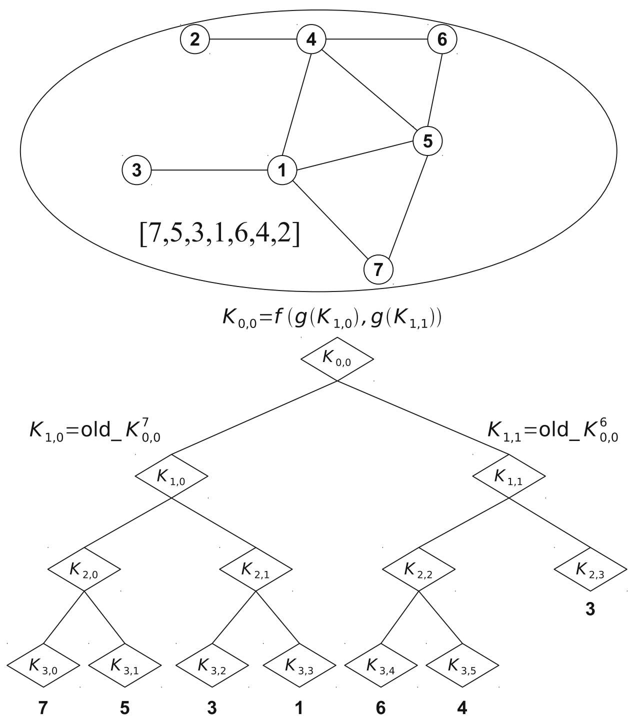

[7,5,3,1,6,4,2]
$K_{0,0}=f(g(K_{1,0}),g(K_{1,1}))$
$K_{1,0}$=old_$K^7_{0,0}$
$K_{1,1}$=old_$K^6_{0,0}$
7
5
3
1
6
4
3

Fig. 5 Creation of a secure group: 3rd iteration

encrypted with the subtree root key. Now every member in the tree regenerates the tree keys in its path to the root and a new SEK $K_{0,0}$ is assumed. At this moment all compromised keys have been updated and the leaving process is finished with a total of just $1 + 2(L - 1)$ messages, with $L = \log_2 N$ for the case of a balanced binary tree with N members or leaves.

For the sake of clarity, Fig. 6 shows an example of the leaving process in which member 2 leaves a group of 8 members and member 5 assumes the role of RM and proceed with the group rekeying.

Notice that, in order to create the secure sessions, the rekeying messages must use hop-by-hop encryption by means of the pairwise keys. Hop-by-hop encryption is more costly than hop-by-hop authentication by means of the SEK, but, in our protocol, such operations are reduced to create secure channels with every leader (GL) of the subtrees hanging from the sibling nodes of the leaving member whenever a sensor node leaves the group or is expelled from it. As seen in Fig. 6, the number of these GLs is equal to the actual depth of the RM in the tree, which, in the case of a balanced tree, is only $\log_2 N$ being N the amount of members/sensors of the group.

5 Managing new member joinings

When the base station wants to insert a new node in the network, it preloads it with the previous unused hash of the

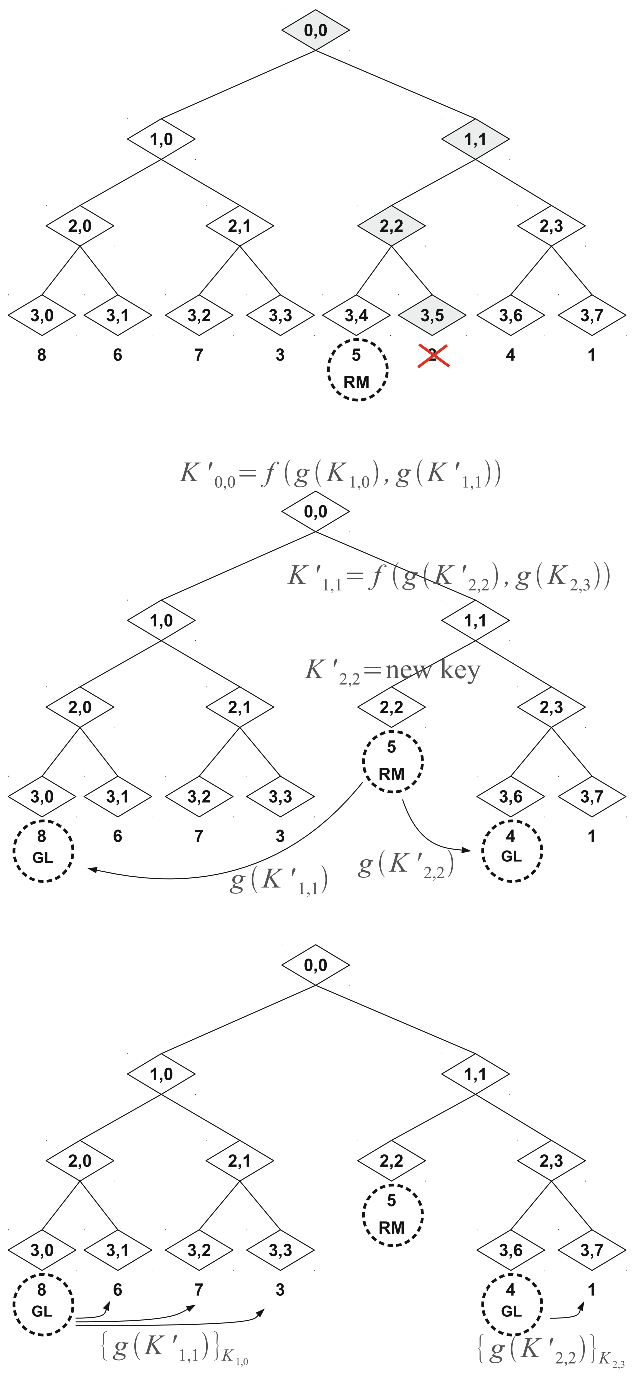

0,0
1,0
1,1
2,0
2,1
2,2
2,3
3,0
3,1
3,2
3,3
3,4
3,5
3,6
3,7
8
6
7
3
5
RM
2
4
1
$K'_{0,0}=f(g(K_{1,0}),g(K'_{1,1}))$
$K'_{1,1}=f(g(K'_{2,2}),g(K_{2,3}))$
$K'_{2,2}=$new key
GL
$g(K'_{1,1})$
$g(K'_{2,2})$
$\{g(K'_{1,1})\}_{K_{1,0}}$
$\{g(K'_{2,2})\}_{K_{2,3}}$

Fig. 6 An example of leaving process

group key. That is to say, assuming that the node is the i-th joining, the base station preloads it with K_g^{k-i} where K_g^{k-i+1} is the hash of K_g^{k-i} as in expression (1). When the new node is deployed it authenticates itself by means of broadcasting its preloaded key and the rest of nodes check the new join by computing the hash of the preloaded key. After the node is authenticated, the rekeying process is similar as it is for a member leaving. First, the node sends its blinded key to the chosen RM. Then, the RM adds a new leaf to the tree (for the new member) moving itself if necessary. After that, the RM updates its key and regenerates all the KEKs in its path to the root. Next, the RM securely sends then necessary blinded KEKs to the rest of member encrypted with the previous shared tree KEKs. After receiving the blinded keys, every member in the tree regenerates the tree keys in its path to the root and a new SEK $K'_{0,0}$ is assumed. Once again we illustrate such behavior with the example of Fig. 7 where member 9 joins the group and the RM is member 5.

6 Evaluation

The target of the following simulations is to show the goodness of the presented protocol for large groups of unattended devices in terms of protocol latency and required messages. As previously mentioned, a GKM protocol provides with a very efficient rekeying mechanism that makes up for the extra cost due to add security to the group creation. The originality of this work is to provide a method to create the secure group thus allowing very efficient rekeying in an unattended, low-cost and secure manner.

Once the secure group is established, the rekeying efficiency of our proposal is that for the tree-based algorithms, which has been widely studied [4, 13, 14]. The only main difference with these proposals is that in our proposal every time there is a rekeying, the cost if assumed by a different node (the corresponding RM) instead of a fixed one.

The main problem with tree-based algorithms is that, as time goes by, the tree can get unbalanced due to new joinings/leavings (especially in the presence of burst). Several proposal have dealt with improving the efficiency by providing batch rekeying, e.g. [28] instead of single rekeying. Batch processing in our proposal will lead to similar results. Consequently, since the rekeying process have been already thoroughly covered in the literature, this section just focus on analyzing the costs of creating the secure group. However, the costs of rekeying are presented in Sect. 8 in comparison with other GKM protocols in the literature.

Since the effect of the underlying technologies, shadowing and multipath is so specific to a given scenario, a specific evaluation couldn't be extrapolated to others. This is why we provided an evaluation at the application layer. In a real scenario, added cost due to the underlying technologies and specific environment should be added; link frames are usually retransmitted and sometimes there is a chance of outage nodes. Our protocol is resistant to this due to isolated nodes are treated as leaving members (they have no communication at all with any other member). Section 7

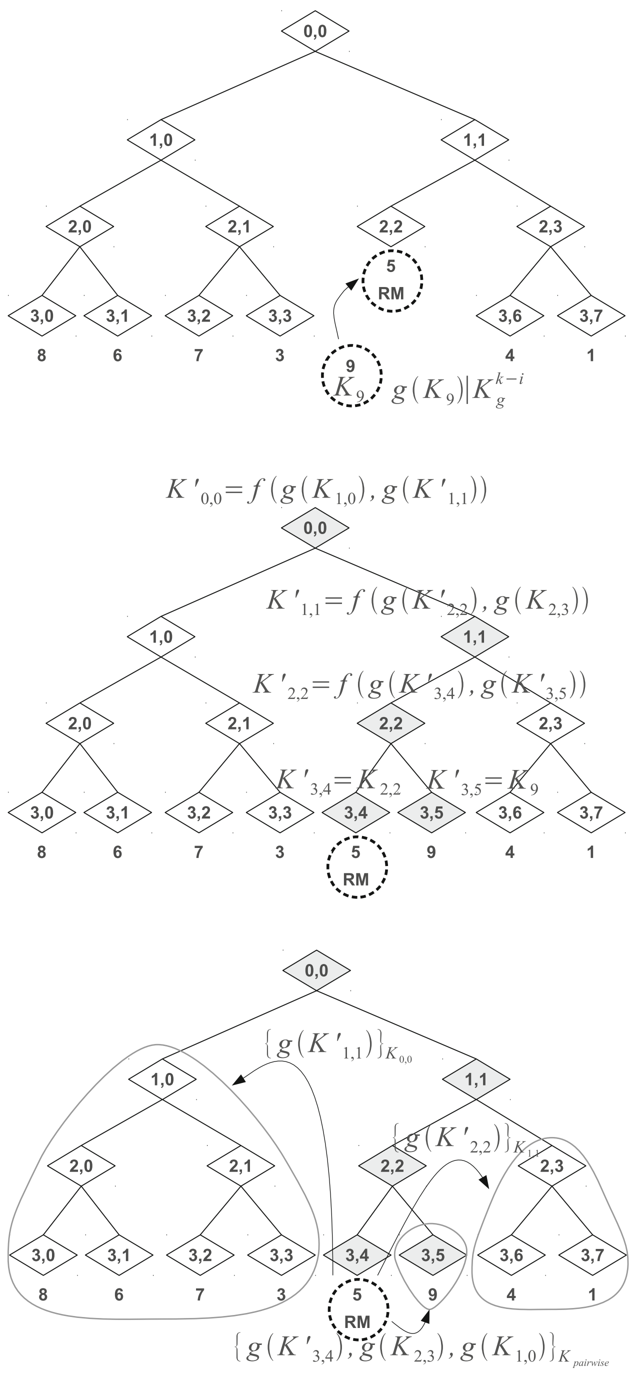

0,0
1,0
1,1
2,0
2,1
2,2
2,3
3,0
3,1
3,2
3,3
3,4
3,5
3,6
3,7
8
6
7
3
5
RM
9
4
1
K_9
$g(K_9)|K_g^{k-i}$
$K'_{0,0}=f(g(K_{1,0}),g(K'_{1,1}))$
$K'_{1,1}=f(g(K'_{2,2}),g(K_{2,3}))$
$K'_{2,2}=f(g(K'_{3,4}),g(K'_{3,5}))$
$K'_{3,4}=K_{2,2}$
$K'_{3,5}=K_9$
$\{g(K'_{1,1})\}_{K_{0,0}}$
$\{g(K'_{2,2})\}_{K_{1,1}}$
$\{g(K'_{3,4}),g(K_{2,3}),g(K_{1,0})\}_{K_{pairwise}}$

Fig. 7 An example of joining process

provides a methodology in order to match this evaluation with a given technology and scenario assumptions.

In the following we evaluate the process of creating the logical tree of KEKs (see Sect. 3.2) with the conditions in Sect. 6.1 and obtain as outputs: (1) the depth of the created tree (Sect. 6.2), which gives us an indicator of the posterior efficiency in terms of rekeying of the created secure group; (2) the required number of rounds for creating the secure group (Sect. 6.3); (3) the average and maximum number of received messages by any group member during the creation of the tree (Sect. 6.4); and (4) the necessary transmitted messages in order to create the group (Sect. 6.5). The expected results are: a tree that is almost balanced, a bearable time to create the group, and a limited or minimized amount of received/transmitted messages.

6.1 Simulation conditions

We have implemented an API of the node operation (see Fig. 2) that can be loaded in a real sensor mote or in a simulated device. However, since practical evaluation of large groups of sensor nodes (hundred to thousands) is out of our extent, we have adopted the latter approach. The API defines the exchange and processing of network layer packets; we have decided not to define an actual MAC/PHY layer since we wanted to evaluate the protocol independently of the underlying wireless physical medium. Nevertheless, we have provided in Sect. 7 a methodology to match this results in a physical technology such as IEEE 802.15.4.

For the simulation, we have adopted the following assumptions:

- All the group nodes have the same characteristics in terms of computational power and transmission/reception range. The range is assumed to be circular surface with radius ρ fixed to 20 m.
- The weight of a node is defined to be a unique identifier of a node and thus we have randomly chosen unique identifiers for every node; the highest the identifier, the greater the weight.
- We place from 10 to 1,000 sensor nodes according to a random uniform distribution over a square area that varies from 50 $\times$ 50 to 50 $\times$ 50 square meters (m^2).
- All the obtained values are averaged from 30 iterations of the protocol.

Before continuing with the evaluation of the protocol, for the sake of clarity, we define a normalized density δ as the density relative to the station range as in (4), with M the length of each of the sides of the square deploying area, and N the number of stations within that area. The normalized density gives us a reliable approximation of the number of stations within the range area of a specific station (including itself).

$$\delta = \frac{N \cdot \pi \cdot \rho^2}{M \times M} \tag{4}$$

Figure 8 shows an area of 200 $\times$ 200 m^2 containing numbers of sensor nodes (32, 100, 1,000), which correspond to densities of 800, 2,500 and 25,000 stations/km^2 respectively and normalized densities of approximately 1.00, 3.14 and 31.41. Blue circles show the range of a given

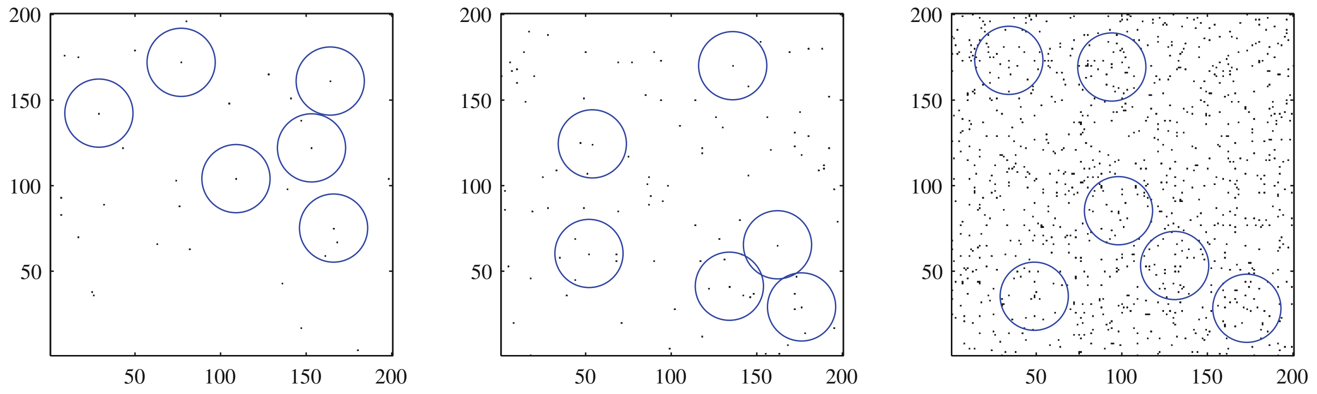


Fig. 8 32, 100 and 1,000 sensors over an area of 200 × 200 m^2

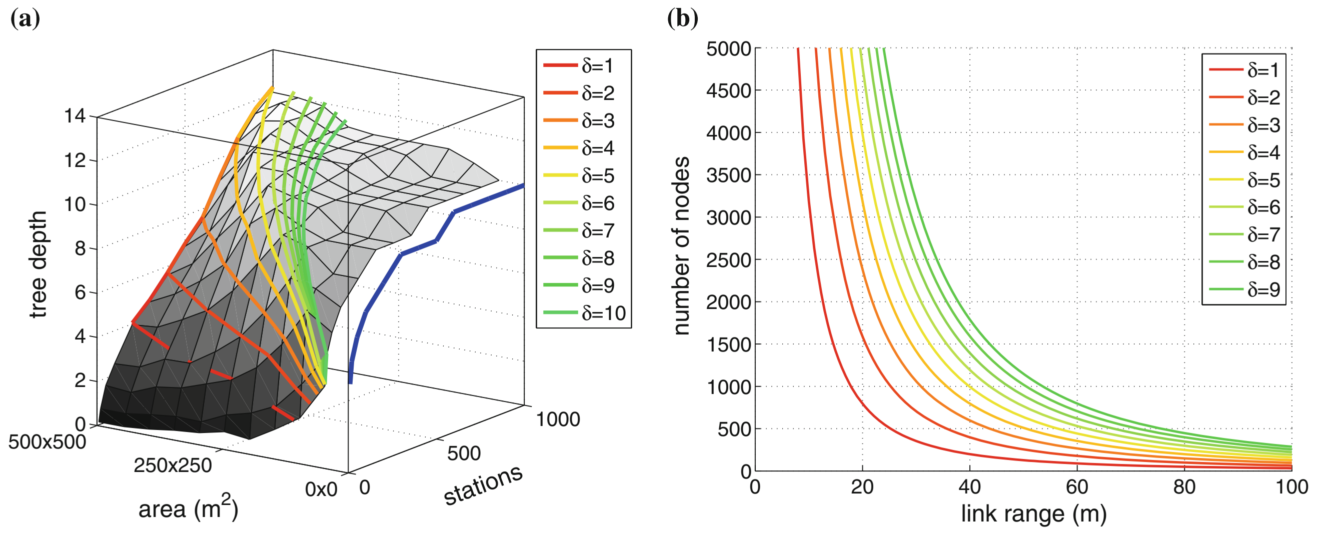


Fig. 9 Tree depth and optimal number of nodes. **a** Depth of the logical tree of KEKs. **b** Optimal number of nodes for fixed area of 1 km^2

sensor node and aid to notice the approximately number of sensors within its range (that actually correspond to δ).

6.2 Tree depth

Secure groups based on logical key trees reach maximum efficiency when the tree is balanced [14]. In such cases, the tree depth is $\log_2 N$, which is, in turn, the minimum number of messages needed for rekeying.

Figure 9a shows the tree depth achieved by the protocol with different numbers of nodes and over different areas. As per the figure, when a single logical tree is achieved, tree depth is always greater than the ideal depth $\log_2 N$ (represented as a blue line). Values of depth less than the optimal are generated when instead of a single group/tree, two or more isolated groups are created. Figure 9 clearly denotes that this situation is with high probability avoided for $\delta \geq 5$.

As expected, for $\delta \geq 5$, the tree depth increases logarithmically with the number of stations and the created logical trees are very close to being balanced (depth of $\log_2 N$). In fact, the created trees only differ from the ideal balanced tree by at most two levels (the achieved tree depth $\in [\log_2 N, 2 + \log_2 N]$). This expected behavior comes from the protocol design guidelines that prioritize trees with the least depth for association. This ensures that the smallest trees do not remain without association for more rounds than necessary and that differences in tree depth cannot increase beyond one or two levels.

$$N = \frac{A}{\pi \cdot \rho^2}\delta \qquad (5)$$

Figure 9b shows the number of nodes versus the node range radius ρ for a fixed area $A = 1$ km^2 as per expression (5). Since, as previously stated, we need a $\delta \geq 5$, if we randomly deploy 1,000 sensor nodes over such area, sensor nodes will need at least an average range of 40 m. On the

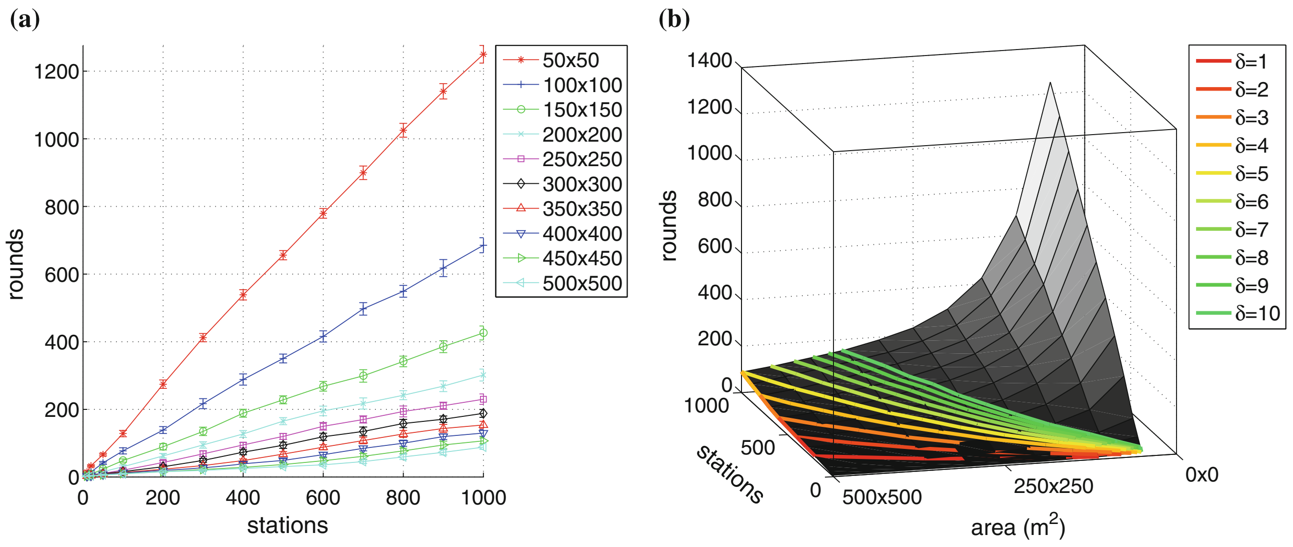


Fig. 10 Rounds in establishing the secure group. **a** Mean and variance. **b** 3D mean

other hand, for 4000 sensor nodes, the necessary range radius is reduced to 20 m. Therefore, as per expression (5) one can obtain a reliable approximation of the necessary amount of sensor nodes to deploy over a given area for a given node range. Or the other way around, an entity deploying the network should be aware of these since it provides it with a reliable estimation of the necessary transmission power for the devices under the given environment conditions. This prior deployment study will hopefully avoid wasting of precious energy.

6.3 Number of rounds

A round is defined as the necessary time taken by a node to receive messages, process them, consequently take a decision and send a message that announces this decision. The election of using rounds instead of directly evaluating spent time rely on the great variety of underlying technologies, protocols and/or setups that actually affect the necessary time. In fact, time highly depends on the frequency of operation, the modulation, the medium access control protocol and/or the error control method. Moreover, the lifetime of a sensor network is often split into equal periods of time made of an active time and an idle time. This period will determine the elapsed time between current and future operations besides of the really necessary time for processing them [1].

From the above reasoning we have adopted the approach of measuring time in terms of rounds, which, gives us the liberty to measure time regardless of the underlying specific features. Moreover, once obtained the right measure of elapsed time per round for a given scenario, the number of rounds provides us with an upper limit of the real time that the protocol spends in establishing a secure group.

Figure 10a shows the mean and variance with each combination of areas and numbers of stations. Figure 10b is a three-dimensional representation in which we have marked lines that show the different values of the normalized density δ. Both figures are obtained by simulation of the protocol under the conditions in Sect. 6.1. From these figures, one can denote that the execution time is a combination of two factors: (1) a linear growth $O(N)$ with the number of stations for a fixed area (that is, linear growth with density); and (2) a logarithmic growth $O(\log_2 N)$ with the number of stations for a fixed density. Obviously, the former, which is lineal, grows faster than the latter. This linear growth derives from the sequential behavior of the protocol regarding every neighbor. If the neighborhood is very large (high density) the sequence will also be very large. Consequently, from this point of view, it is preferable to have the lowest possible density but guarantying the group connectivity ($\delta \geq 5$). Therefore, in order to maintain the efficiency of the protocol we should control the deployment density. In other words, if we need to include more members in the group we will have to distribute the group across a larger area or use nodes with less transmission/reception range.

We shall now explain this behavior in greater detail by analytically getting to similar results. We define the normalized area as $\alpha = \frac{A}{\pi \cdot range}$, so we can express the normalized density as $\delta = \frac{N}{\alpha}$.

Following the protocol guidelines, in every phase, the depth of the created trees increases by 1 and the number of GLs is a half until only one remains (an only tree). As a result, the number of phases is equal to depth of the final tree, that assuming that the protocol leads to a balanced tree, will be $\log_2 N$.

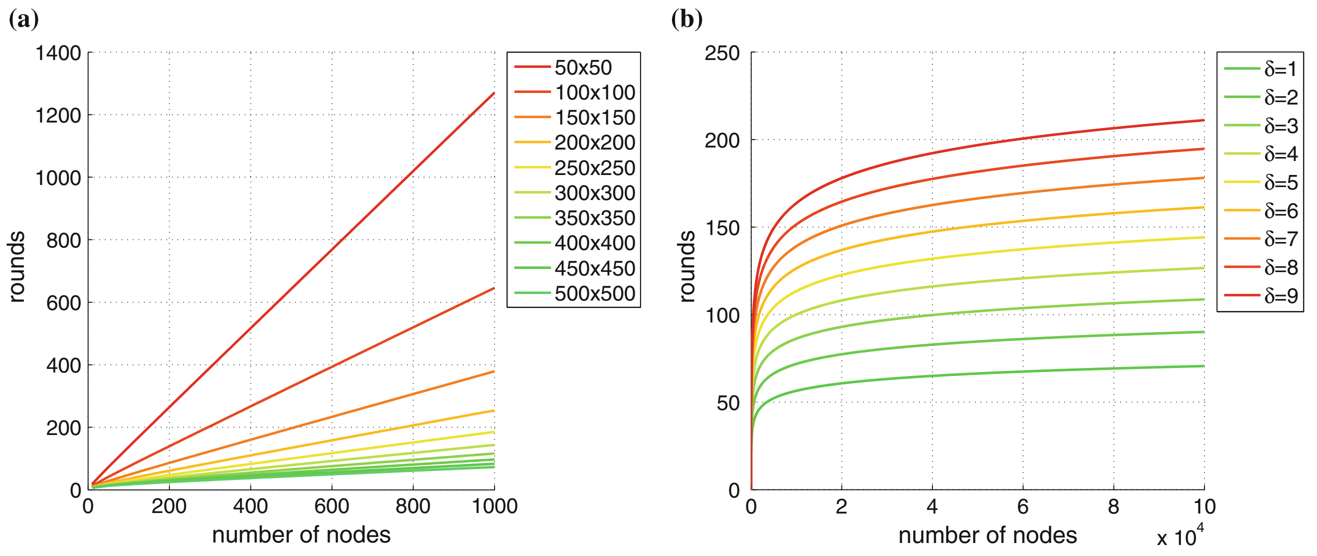


Fig. 11 Analytical approximation of the number of rounds used to construct the logical key tree. **a** Rounds for a fixed area, range = 20 m. **b** Rounds for a fixed density, range = 20 m

Following the protocol guidelines, in every phase, all the sets of neighboring GLs take actions sequentially. As a result, the last one associating will have to wait first for its v neighboring GLs. Considering that the number of rounds needed to complete an association process is 2 and that 1 more round is spent to report it to the neighboring GLs, we can assure that a total of $\beta\ v + 3$ rounds are needed to complete a phase, where $\beta \geq 1$ is the mean number of rounds that every neighbor GL waits for GLs in other neighborhoods. Thus, considering that the protocol leads to a balanced tree, the total rounds can be expressed as in (6).

$$r = \sum_{i=0}^{\log_2(N)-1} (v_i\beta + 3) \tag{6}$$

At the beginning of the protocol, consider phase 0, every member is a GL of a tree with an only member; consequently, every GL has an averaged neighborhood of δ GLs (including itself); thus its $v_0 = \delta$. In the next phase, phase 1, the GLs (now leading groups of two members) are a half, but they have at most a double of neighbors, that is to say, that they have approximately the same amount of neighboring GLs. As a result, for the sake of clarity, we assume v_1 continue being approximately $v_1 = v_0 = \delta$, and so on with every iteration. However, in the final phases, the total number of GLs can be less than the value of δ (until get just one GL) and so $v \neq \delta$; notice that there can be just two GLs at the last phase, 4 at the last but one, and so on. Considering that the protocol leads to a balanced tree, the number of neighboring GLs v can be analytically approximated as in (7) with $i \in [0, \log_2 N - 1]$ the current phase of the protocol.

$$v_i = \begin{cases} \delta & \text{if} \quad 2^i \geq \delta \\ 2^i & \text{if} \quad 2^i < \delta \end{cases} \tag{7}$$

From the previous reasoning, replacing (7) in (6), the total spent rounds can be denoted as in (8).

$$\begin{aligned} r &= 3\log_2 N + \sum_{i=0}^{\log_2 \delta - 1} \beta 2^i + \sum_{i=\log_2 \delta}^{\log_2 N - 1} \beta\delta \\ &= 3\log_2 N + \beta\delta\left(1 + \log_2 \frac{N}{\delta}\right) - \beta \end{aligned} \tag{8}$$

In order to compare (8) with the results obtained by simulation, we replace $\delta = N/\alpha$ in (8) in order to obtain a straight line with a slope of $m = \frac{\beta(1+\log_2 \alpha)}{\alpha}$ as in (9). Figure 11a represents (9) with the same parameters as in the simulation. As expected, the analytical results (Fig. 11a) are fairly similar to the ones obtained by simulation (Fig. 10a).

$$\begin{aligned} r &= 3\log_2 N + \beta\frac{N}{\alpha}(1 + \log_2 \alpha) - \beta \\ &= \underbrace{\frac{\beta(1 + \log_2 \alpha)}{\alpha}}_{m} N + 3\log_2 N - \beta \end{aligned} \tag{9}$$

Until now we have evaluated the rounds spent for creating a secure group over a fixed area varying the density (the number of deployed sensor nodes). However, it is also very important to analyze the scalability of the protocol for a fixed density over a varying area. With such a purpose, let us re-express (8) as in (10), which clearly denotes a logarithmic increase in the number of rounds when δ is a constant (see Fig. 11b). This conclusion is particularly relevant to our protocol because it indicates that the

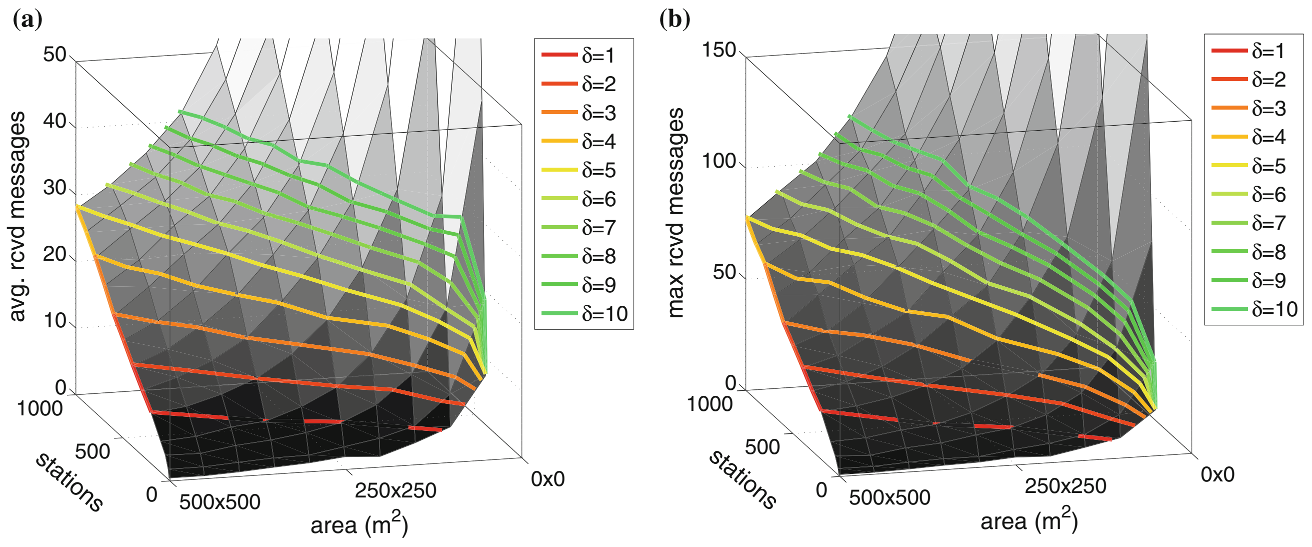


Fig. 12 Number of received messages during the creation of the group. **a** Average received msgs. **b** Maximum received msgs by a node

protocol is also scalable in terms of covering a wider area deploying more sensor nodes but keeping a fixed density.

$$r = (3 + \beta\delta)\log_2 N + \beta\delta(1 - \log_2 \delta) - \beta \qquad (10)$$

It is important to note that limiting δ should not make more difficult the process of establishing the group key. If the density is too high, we can reduce it by: (1) expanding the group over a larger area; (2) reducing the number of members; and (3) choosing, as aforementioned, nodes with less transmission power (range) before the deployment in order to avoid the waste of precious energy.

As previously stated at the beginning of this section, the aim of using round measurements is to provide an upper limit to the time spent for the creation of the group regardless of the underlying technologies, protocols, setups, etc. For example, in 802.15.4 [18] a PPDU frame is at most 133 bytes and the transmission rate is at least 20 Kbps, thus a frame would be transmitted in at most $\frac{133\times8}{20\times10^3} = 53.2$ ms. Assuming, a sensor mote with at least a 4 MHz CPU and that we can parallelize transmission/reception tasks with processing, we could select a very over-sized value of a round of e.g. 1 s, which will be time enough to receive a few frames, process them and send a message. As a result, as denoted in Fig. 11b, it takes only 160 rounds (just 2 or 3 min) to create a very large unattended group of 100,000; and this is a perfectly assumable time for low-rate networks such as WSNs.

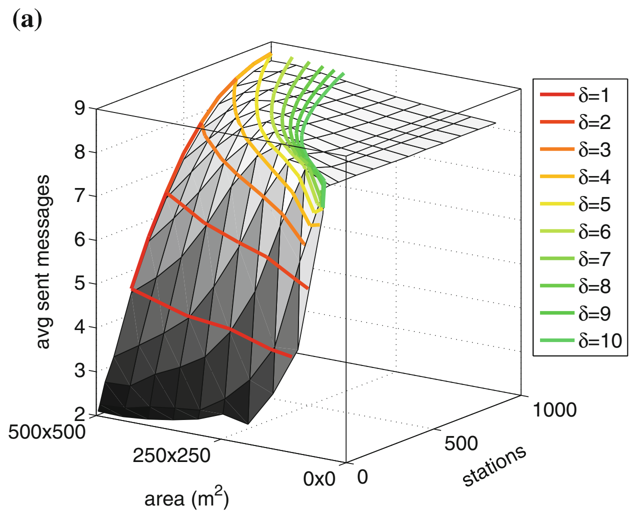
6.4 Received messages

The cost of receiving a wireless message is less than an typically half of the cost of transmitting it, e.g. 28.6 μJ/byte against 59.2 μJ/byte for Mica2dot sensor mote at 3 V and 5 dBm transmit power [35]. As a result, we have also analyzed it. Figure 12a, b represent the average and maximum number of messages received by a node. One can clearly denote that the number of necessary average received messages for appropriate deployment densities ($\delta \in [6, 10]$) is approximately between 20 and 40 messages even during the creation of a large secure group of 1,000 members; and the maximum received messages is approximately between 80 and 120 messages. Simulation results, once again, denote a bearable cost in terms of received messages.

These figures are deliberately focused on the values around standard values of normalized density δ in order to show the goodness of the protocol. However, the actual complete shape would have a fast growth with δ as the reflected results in Fig. 10b. And this is important because it clearly denotes that this protocol can lead to exhaustion for very dense networks. This is the expected behavior since one message is received (either a *req_msg* or an *ack_msg* or a *report_msg*) in every round, and these messages are the result of a decision. As decisions are taken sequentially in every neighborhood then, at a first stage, rounds have a linear dependency with δ. As a consequence, one can conclude that the protocol is not optimized to high density networks but to standards deployment where every node has an average from 6 to 10 nodes within its range; actually desired values for multihop sensor networks (unattended networks preclude the existence of a base station) [24], by the way.

6.5 Sent messages

One of the main aims of this proposal is to minimize the number of messages that a given node must send. This is very important because the transmission of a bit is by far



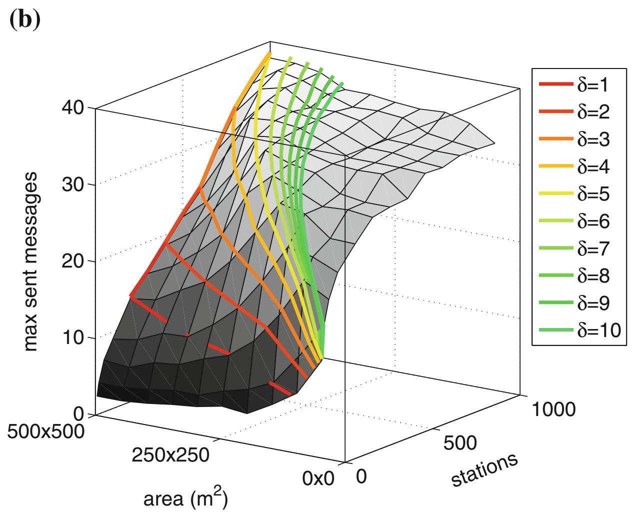


Fig. 13 Number of transmitted messages during the creation of the group. **a** Average sent messages. **b** Max. messages sent by a node

(up to three orders of magnitude more than processing [24]) the more energy consumption process.

Figure 13a shows the average number of messages sent by a node during the execution of the protocol. It can be clearly seen that the protocol converges to an upper limit of eight messages. We can explain this behavior analytically because the number of nodes sending messages (current GLs) is reduced by half during each iteration. In this case, there is a special initial phase in which each of the N nodes sends a message; associations are then repeated until a single logical key tree is constructed. Since associations are configured in pairs, the number of GLs is reduced by half in each iteration.

If we denote μ as the average number of messages sent by a GL during each iteration, we can denote the average number of messages $\overline{M}$ sent by a node as in (11). If we apply the limit for large values of N to this expression, we can reformulate it as in (12). Since, for the case of a balanced tree, $\mu \simeq 3.5$, one clearly denote that expression in (12) clearly reflects the same behavior as the simulated results (Fig. 13a).

$$\begin{aligned}\overline{M} &= \frac{1}{N}\left[N + N\mu + \frac{N}{2}\mu + \frac{N}{4}\mu + \cdots + \mu\right] \\ &= 1 + \mu \sum_{i=0}^{\log_2 N-1}\left(\frac{1}{2}\right)^i = 1 + 2\mu - 2\frac{\mu}{N}\end{aligned} \qquad (11)$$

$$\lim_{N\to\infty} \overline{M} = 1 + 2\mu \qquad (12)$$

Figure 13b shows the behavior of the increase in the maximum number of messages sent by a member of the group for different areas and numbers of nodes. We can deduce from the figure that the maximum number of sent messages increases with $\log_2 N$. Note that the node that sends the most messages is the final GL; it will therefore send messages during each of the $\lceil \log_2 N \rceil$ iterations. Since μ messages are sent per iteration, we can determine that the node will send $\lceil \log_2 1000 \rceil \cdot 3.5 \approx 35$ messages for 1,000 members (with $\mu \simeq 3.5$), which matches the simulation results in Fig. 13b.

At this point it is useful to recall that: (1) with a trivial solution, the key server uses N messages to establish the group key and N messages for rekeying; (2) with LKH, the key server uses N messages to establish the group key and $O(\log_2 N)$ messages for rekeying; and (3) with our proposal, the node that sends the most messages only sends $\mu * \log_2 N$ when the group key is established and, as with any other tree-based algorithm, $O(\log_2 N)$ messages for the rekeying process. Moreover, with our protocol every node in the group only sends an average of $1 + 2\mu$ messages due to the distribution of the GKM tasks which is a very valuable property in UWSNs.

7 Obtaining energy costs in a typical 802.15.4 scenario

The aim of this section is to show the proposed protocol's goodness in terms of energy consumption in a real scenario. With such purpose, we have adopted the IEEE 802.15.4 [18] technology, which is one of the most extended technologies for sensor networks. Next we defined the scenario assumptions.

Figure 14 shows an example of frame structure using symmetric cryptography. At PHY layer, we have assumed popular 2.4 GHz operation of IEEE 802.15.4 leading to a 5 bytes (10 4-bit symbols) long synchronization header and a 1 byte (or 2 symbols) long PHY header. These fields are followed by a variable length (up to 127 bytes) PHY payload. At MAC layer, we use short addressing (2 bytes per address) since we can assume that a give node will not

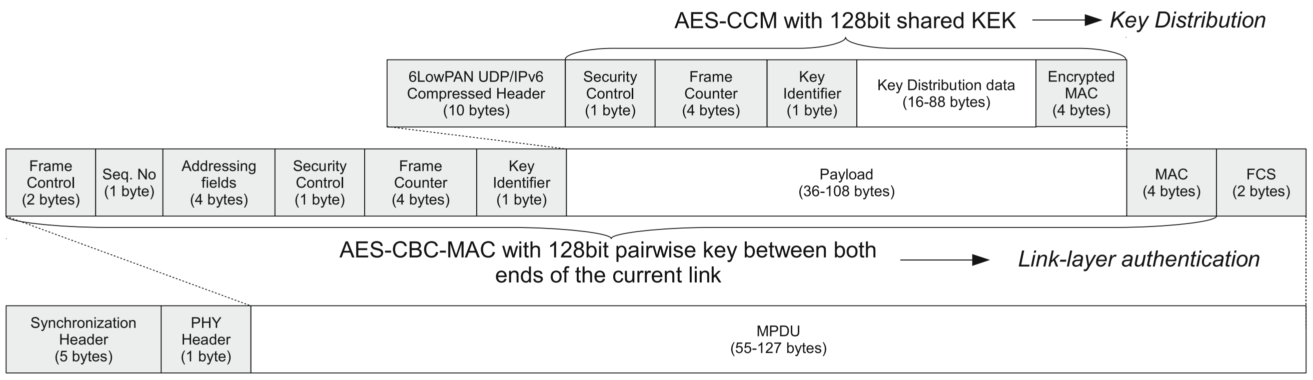

AES-CCM with 128bit shared KEK
Key Distribution
6LowPAN UDP/IPv6 Compressed Header (10 bytes)
Security Control (1 byte)
Frame Counter (4 bytes)
Key Identifier (1 byte)
Key Distribution data (16-88 bytes)
Encrypted MAC (4 bytes)
Frame Control (2 bytes)
Seq. No (1 byte)
Addressing fields (4 bytes)
Security Control (1 byte)
Frame Counter (4 bytes)
Key Identifier (1 byte)
Payload (36-108 bytes)
MAC (4 bytes)
FCS (2 bytes)
AES-CBC-MAC with 128bit pairwise key between both ends of the current link
Link-layer authentication
Synchronization Header (5 bytes)
PHY Header (1 byte)
MPDU (55-127 bytes)

Fig. 14 The proposed frame format

have more than 2^{16} one-hop neighbors. Further, we have assumed a secure deployment where AES-CBC-MAC with 128 bits keys is used for authenticating the radio link. Moreover, we have adopted 6LowPAN [17] with compressed UDP/IPv6 headers of 10 bytes. For key management, the transmitted KEKs are securely sent using AES-CCM (Encryption + Authentication) with 128 bits shared keys. Consequently, as shown in Fig. 14, from 16 to 88 bytes can be used for key management data with symmetric cryptography. A similar approach may be done for asymmetric cryptography. For more details about the frame format fields, we refer the reader to [18].

Obviously, the energy costs for transmitting a given frame will rely not only on the actual frame length but also on the average number of transmissions $\overline{N}_{tx}(\overline{\gamma}, F)$ needed to ensure a successful reception for a given average reception signal-to-noise ratio (SNR) $\overline{\gamma}$ and a frame length of F bytes.

We next assume that an ACK packet can be successfully transmitted in a single attempt. This assumption relies on the fact that ACK packets are much smaller and hence much likelier to go through, and on temporal channel correlation which ensures that, if the data packet experienced a good channel, the return path should experience the same beneficial channel conditions. Therefore, $\overline{N}_{tx}(\overline{\gamma}, F)$ can be approximated as in (13), where PEP(F) is the packet error probability (PEP) of a frame of F bytes.

$$\overline{N}_{tx}(\overline{\gamma}, F) \approx \frac{1}{1 - \mathrm{PEP(F)}(\overline{\gamma})} \qquad (13)$$

Usually, in order to characterize the average number of retransmissions, three wireless operating conditions are distinguished [33]: (1) fast fading where the Nakagami-m channel varies from symbol to symbol (ergodic over packet); (2) block fading where the channel remains constant over a packet but changes from packet to packet (ergodic over retransmission window); and (3) fading where the channel remains constant over time but varies in space (non-ergodic). Details of how to analytically model the three conditions in IEEE 802.15.4 are presented in [3].

We will however subsequently only consider the former case with block channel coder, although the analysis and insights for the other cases would be very similar. In this case, the average number of transmissions under fast Nakagami-m fading conditions with block channel coder can thus be expressed [3] as in (14), where k is the uncoded number of information bits in the code word and t the number of errors which can be corrected by the code. The constants η and θ depend on the choice of modulation and modulation order M, but are defined as $\eta = (1/(max(2, \log_2 M)\sqrt{\pi}))\Gamma(m + 1/2)/\Gamma(m + 1)$, $\theta = \sin^2(\pi/M)$ for M-PSK. The value of c is obtained as $c = 1/((t + 1)\mathrm{B}(\mathrm{t} + 1, \mathrm{J} - \mathrm{t}))$ being $\mathrm{B}(\mathrm{x}, \mathrm{y})$ the Beta function and J the word length in bits.

$$\overline{N}_{tx}(\overline{\gamma}, F) \approx \left(1 - c\left(\eta\left(\frac{m}{m + \theta\overline{\gamma}}\right)^{m}\right)^{t+1}\right)^{\frac{-F}{8k}} \qquad (14)$$

In 802.15.4 at 2.4 GHz every 4-bit symbol ($k = 4$) is encoded into a sequence of 32 chips ($J = 32$) that are actually transmitted over the air; that is to say that l bytes will actually require to send $2l$ 4-bit symbols that finally leads to transmit $2l$ 32-bits chip sequences or $8l$ bytes. The minimum distance between two chip sequences is 12 bits and thus we can assume $t = 5$. As per [18], the 2.4 GHz PHY shall employ direct sequence spread spectrum (DSSS) with offset quadrature phase-shift keying (O-QPSK) for chip modulation. Therefore, we can assume $M = 4$. As in O-QPSK, average reception SNR should be at least 12.5dB in order to consider a neighbor within the transmitter range (bit error rate less than 5 %), we define $\overline{\gamma} = 12.5$dB as the most restrictive case. Finally, as stated in [27], the Nakagami factor can be approximated as a deterministic factor $m \approx 1.19$ for this scenario assuming outdoor measurements.

For details about how to analyze the effect of shadowing on $\overline{\gamma}$ and thus to the density deployment we refer the authors to [3]. Shadowing will increase the probability of some nodes being isolated from the rest of the group. It worths mentioning here, that this will not break the protocol. Quite

Table 1 Energy costs with Mica2dots ATmega128L

Tx (τ)	Rx (ρ)	AES128 enc.	AES128 dec.	RSA1024 sig.	RSA1024 ver.
$59.2\,\frac{\mu J}{\text{byte}}$	$28.6\,\frac{\mu J}{\text{byte}}$	$1.62\,\frac{\mu J}{\text{byte}}$	$2.49\,\frac{\mu J}{\text{byte}}$	304 mJ	11.9 mJ

the opposite, it will create several secure groups of interconnected devices, which is the desired behavior.

Next we present real estimations of the energy costs when the sensor network is made up of Berkeley/Crossbow Mica2dots with Atmel ATmega128L 8-bit micro-controller, which is a popular platform for WSN research. This platform was analyzed in [35] and the authors obtained energy consumption for individual cryptographic algorithms and data transmission/reception with less than a 5 % deviation by measuring the current drawn from the power supply. Table 1 extracts the useful data for this analysis, that is to say the costs of transmission, reception, AES-128 encryption, AES-128 decryption, RSA-1024 signature creation and RSA-1024 signature verification.

It is interesting to note that the power required to transmit 1 byte is more than one order of magnitude greater than the cost of encrypting/decrypting it using symmetric cryptography. This is not the case of asymmetric cryptography that not only creates bigger frames (besides the added signature length, output is a multiple of the key) but also has 3-order of magnitude greater power consumption. This fact suggests to use symmetric cryptography whenever it is possible.

With asymmetric cryptography, we can neglect transmission costs with respect to signature creation and verification. Therefore, we can obtain a lower bound of the cost of transmitting a key management packet as the cost of creating the signature, and of reception as the cost of verifying it.

Quite the opposite, with symmetric cryptography, cryptographic operations incur negligible costs with respect to sending/transmitting data. Consequently, and taking into account that key management data must be a multiple of 16bytes because of the use of 128-AES-CCM, the cost of transmitting/receiving the necessary key management data with symmetric cryptography can be obtained as in (15) and (16), being L the amount of bytes to be sent, $\varpi = 45 + 16 \cdot L \bmod 80$ the size of the last frame in bytes, τ the cost of transmitting 1 byte and *rho* the cost of receiving 1 byte.

$$\varepsilon_{tx}(L) = \left(\left\lfloor \frac{L}{80} \right\rfloor 125\overline{N}_{tx}(\overline{\gamma}, 125 \cdot 8) + \varpi \overline{N}_{tx}(\overline{\gamma}, \varpi \cdot 8) \right) \tau \tag{15}$$

$$\varepsilon_{rx}(L) = \left(\left\lfloor \frac{L}{80} \right\rfloor 125\overline{N}_{tx}(\overline{\gamma}, 125 \cdot 8) + \varpi \overline{N}_{tx}(\overline{\gamma}, \varpi \cdot 8) \right) \rho \tag{16}$$

It can be clearly denoted the weak impact of our proposal in a real scenario. During the creation of the group, every key management data packet will fit in a single frame. As a result the cost of sending and receiving a packet will be respectively $\varepsilon_{Tx}(16) \approx 3.61\,\text{mJ}$ and $\varepsilon_{Rx}(16) \approx 1.75\,\text{mJ}$. For example the average energy cost per node when creating the secure group of 1,000 nodes under good density conditions (from 6 to 10 neighbor nodes) can be approximated by the amount of necessary received and transmitted bytes (see Figs. 12b, 13b) that is to say that it will be less than $40\varepsilon_{Rx}(16) + 8\varepsilon_{Rx}(16) \approx 98.72\,\text{mJ}$ per node with a global impact of just 98.72 J.

As explained in Sects. 5 and 4, when a node joins, leaves or it is expelled from the group, the specific node chosen as RM sends $\lceil \log_2 N \rceil$ messages, with a varying number of KEKs, that are actually forwarded by the GLs, as a result the nodes spending more energy are the RM and the GLs (a total of $\lceil \log_2 N \rceil$). This represent a cost of $\sum_{i=1}^{\lceil \log_2 N \rceil} \varepsilon_{Tx}(16 * i) \approx 97.47\,\text{mJ}$ for each of them, and a global impact ε_G as in (17) of approximately 1.97J.

$$\varepsilon_G = \varepsilon_{Tx}(16) + \sum_{i=1}^{\lceil \log_2 N \rceil} (2\varepsilon_{Tx}(16 * i) + \varepsilon_{Rx}(16 * i)) + (N - \lceil \log_2 N \rceil - 1)\varepsilon_{Rx}(16) \tag{17}$$

Assuming that every sensor of the network has a packet rate Υ and a rate of new joinings/leavings of Φ, the percentage of the consumption of the rekeying process with respect to the global operation of the WSN is then:

$$\frac{\Phi \varepsilon_G}{N\Upsilon(\varepsilon_{Tx}(16) + \varepsilon_{Rx}(16))} \tag{18}$$

which, under the same conditions as before, is approximately $0.368 \frac{\Phi}{\Upsilon}$. Obviously, if the packet rate is much higher than the dynamism of the group, as we expect in a WSN, the global impact of rekeying is very low. For example, assuming a packet rate of one packet (e.g. a measure) every 30 s and a rate of new/leaving nodes of 1 node every day, the costs of rekeying is only 0.13 ‰ of the global operation.

A similar approach can be carried in order to find the costs associated to this and other proposals in this and another scenarios. We have however computed the costs as in this section for every other proposal in the comparison provided in the next section. Obtained costs are shown in Table 2 in braces.

8 Comparison with other protocols

We will next compare the communication costs of our protocol with other widely known proposals with a similar goal. We have chosen representative proposals based on

Table 2 Comparison with other protocols

	[21]	LKHW [10]	GDH [32]	TGDH [19]	[15]	Our proposal
Topology	Clusters	Tree	Ring	Tree	Tree	Tree
Unattended	No	No[a]	Yes	Yes	Yes	Yes
Decentralized	Yes	No	Yes	Yes	Yes	Yes
Asymmetric cryptography	Yes	No	Yes	Yes	Yes	No
Total msgs. for the group creation	*clusters*:$O(N \log N)$ $\{O(57.47\ \text{J})\}$ + *GDH*: $2\,(N-1)$ $\{631.17\ \text{J}\}$ + *key dist.*: $<2\,(N-1)$ $\{<631.17\ \text{J}\}$	N $\{5.36\ \text{J}\}$ + direct fusion process	ring creation $+ 2\,(N-1)$ $\{631.17\ \text{J}\}$	$2N \log_2 N$ $\{6{,}318\ \text{J}\}$	$<8\ N\{<2{,}527.2\ \text{J}\}$	$<8\ N\{<42\ \text{J}\}$
Rounds for the group creation	*clusters*:$O(N \log N)$ + *GDH*: $2\,(N-1)$ + *key dist.*: $<2\,(N-1)$	1 + direct fusion process	ring creation $+ 2\,(N-1)$	$\sum_{i=1}^{\log_2 N} \frac{N}{2^i} 4 = 4N\left(1-\frac{1}{2^{\log_2 N}}\right)$	From $O(\log_2 N)$ to $O(N)$[b]	From $O(\log_2 N)$ to $O(N)$[b]
Total msgs. for batch rekeying	$<4\,(N-1)\{<1{,}262.34\ \text{J}\}$	$\log_2 N\{1{,}870.88\ \text{mJ}\}$	$2\,(N-1)\{631.17\ \text{J}\}$	$\log_2 N$[c] $\{3{,}474\ \text{mJ}\}$	$2(\log_2 N-1)$ $\{5{,}686.2\ \text{mJ}\}$	$2(\log_2 N-1)$ $\{1971.97\ \text{mJ}\}$
Total msgs. when forced rekeying	$<4\,(N-1)\{<1{,}262.34\ \text{J}\}$	$1+\log_2 N$ $\{1{,}874.49\ \text{mJ}\}$	$2\,(N-1)$ $\{631.17\ \text{J}\}$	$1+\log_2 N$[c] $\{3{,}159\ \text{mJ}\}$	$1+2\,(\log_2 N-1)$ $\{6{,}002.1\ \text{mJ}\}$	$1+2\,(\log_2 N-1)$ $\{1{,}968.36\ \text{mJ}\}$

[a] Unless a powerful sink or CH is also on the deployed network

[b] See expressions (9) and (10)

[c] In fact it is 2 multicast/broadcast messages but containing the same amount of info as $\log_2 N$ messages with one key per message

clusters [21], logical trees of keys [10, 15, 19] or ring structures [32]. All the proposals achieve not only the creation of a secure group but also how to secure it when the membership changes. Table 2 summarizes the main features of these proposals. Moreover, we provide with a quantitative comparison of the aspects that have the deepest impact in the performance of a GKM protocol: the total messages for the group creation, the rounds for the group creation, the sent messages for batch rekeying and the sent messages when forced rekeying. Energy costs in braces have been obtained, just for reference purposes, by applying the methodology in Sect. 7 to a group of 1,000 sensor nodes. These costs account for both sending and receiving the messages.

The main target of providing such table is to provide a comparison between the most important aspects that affect the efficiency of a GKM protocol. As a result, in Table 2, we classify the presented proposals into:

- Topology: if the group is organized in clusters, trees of keys or rings structures. Cluster based proposals first split the management domain into smaller areas or clusters, each one leaded by a clusterhead; once the clusters are created, in decentralized proposals, the clusterheads share the rekeying task when the membership changes. Proposals based on logical trees of keys create a logical backbone of shared keys that actually makes rekeying easier and more efficient. Proposals based on ring structures rely on group key agreement algorithms where the future groupwise key passes through each and every node before is agreed; its main drawback for large groups is that getting to a ring structure is usually not that easy.
- Unattended: specifies if the group is able by itself to self-organize and maintain security without attended operation (e.g. a base station).
- Decentralized: whether there is an only entity or node in charge of managing security or it is shared by a set of members.
- Asymmetric cryptography: reflects if the proposal requires the use o asymmetric cryptography in order to achieve group security. Note that its use is costly and many times not affordable for WSNs.
- Total msgs. for the group creation: approximately presents the required total number of transmitted messages in order to initially create the secure group or to entirely recreate it (backbone reset). Here we measure all the messages in the network and not the average messages per node. In the case of decentralized unclustered proposals, obtaining the average messages per node is fairly straightforward: it is just the total messages divided by the number of nodes N (the average sent messages for our proposal is detailed in Sect. 6.5). In the case of clustered proposals, almost all of the total transmitted messages are sent by the clusterheads and then obtaining the average would not be a representative parameter.
- Rounds for the group creation: as already defined in Sect. 6.3, the rounds provide us with an estimation of the required time in order to completely create a secure group that does not depend on the underlying technologies and protocols.
- Total msgs. for batch rekeying: the necessary transmitted messages for standard (policy required or periodic) updates of the group key.
- Total msgs. when forced rekeying: required total transmitted messages when a rekeying is forced because of a member joining or leaving the group.

Table 2 clearly shows that the protocols based on tree topologies are the ones performing best in terms of necessary messages for rekeying: they achieve an efficiency of $O(\log_2 N)$ instead of $O(N)$ in terms of necessary transmitted messages. On the other hand, centralized tree-based protocols such as LKHW are the fastest in terms of group creation since they have a centralized manager that actually imposes the tree structure instead of a set of nodes that have to cooperate to create the secure group. Obviously, when computing costs, there are important differences between proposal using symmetric and asymmetric cryptography during the message exchange.

Assuming a (required) decentralized approach for an unattended network, regarding to the creation of the secure group, we can conclude that: (1) our protocol under best conditions (the deployment area increases as new nodes are added—fixed density) spends only $O(\log_2 N)$ rounds, while in the worst conditions (fixed area and thus the density increases as new nodes are added) gets to $O(N)$ which is a similar value than the obtained with the other decentralized approaches; (2) the total transmitted messages for the group creation is, as for the other approaches, $O(N)$, although the cost are much shared by the group members and a single entity will send at most $O(\log_2 N)$ messages (see Sect. 6.5).

On the other hand, regarding the cost of rekeying, from Table 2 we can clearly denote that, besides using lighter symmetric cryptography, as the other tree-based proposals, the necessary transmitted messages are limited to only $O(\log_2 N)$.

Summarizing, TGDH is the only proposal that achieves the same global performance as ours. However, [15] and this proposal are able to create the secure group faster, with less messages and less energy consumption. As a result, they are able to better recover against a mass disconnection (new group creation). Moreover, this proposal, as it does not rely at all on asymmetric cryptography, has substantially reduced the energy consumptions (see example of costs in braces). As a result, we can conclude that our proposal is perfectly suited to UWSN and especially for those made of low-end devices with computational constraints.

9 Conclusions

In this document we have presented a GKM protocol targeted to UWSNs. In keeping with the special requirements of these networks, it has been designed to be performed by a group of unattended low-end devices cooperating in a completely distributed and decentralized manner. Therefore, the main target of the protocol has been to minimize the added costs due to GKM. In this work, we have presented the protocol's operation when a new secure group is formed, as well as when it resolves membership changes.

Our protocol achieves the scalability of tree-based GKM solutions while being completely distributed and not relying on asymmetric cryptography. We have reported simulation results of the initial establishment of the secure group. We have matched these results with an analytical study of the protocol behavior. The results prove that the protocol has no deadlocks, that the logical key tree formed tends to be balanced, that the bandwidth spent in terms of sent and received messages only grows with the logarithm of the number of members of the group, and that the secure group can be established in a limited number of rounds. Moreover we have provided a methodology in order to estimate the realistic energy costs with a given physical technology. With all these results we can conclude that the protocol is perfectly suited to large groups of sensor nodes.

Thus far, we have analyzed the number of necessary transmitted messages but not the cost of relaying these messages, closely related to the ad hoc routing protocol. A cross-layer solution integrating the management of the secure group with the routing algorithm seems to be the appropriate way of minimizing the relaying costs, and this is now the current and future research line of this proposal.

Moreover, how different policies of weight assignments could actually affect the network lifetime is part of a future line. A function to choose a random unique identifier for every member every time the group is to be created or recreated is fairly straightforward, but it remains to study the effect of assigning the weights depending on node capabilities, such as battery left.

Acknowledgments This work has been partially supported by the Spanish *Comisión Interministerial de Ciencia y Tecnología* (CICYT) with the project P2PSEC (TEC2008-06663-C03-01), the Spanish *Ministerio de Ciencia e Innovación* with the CONSOLIDER project ARES (CSD2007-00004), the Spanish *Ministerio de Industria, Energía y Turismo* with the AVANZA funds for ITEA2 project DiCoMa (TSI-020400-2011-55) and the *Generalitat de Catalunya* with the grant 2009 SGR-1362 to consolidated research groups awarded to the Information Security Group of the *Universitat Politècnica de Catalunya*.

References

1. Akyildiz, I., Su, W., Sankarasubramaniam, Y., & Cayirci, E. (2002). Wireless sensor networks: A survey. *Computer Networks, 38*(4), 393–422.
2. Balenson, D. M., McGrew, D., & Sherman, A. (1999). Key management for large dynamic groups: One-way function trees and amortized initialization. In: *IRTF SMUG meeting*. Internet draft. Draft-balenson-groupkeymgmt-oft-00.txt.
3. Bartoli, A., Hernández-Serrano, J., Soriano, M., Dohler, M., Kountouris, A., & Barthel, D. (2011). Secure lossless aggregation over fading and shadowing channels for smart grid m2m networks. *IEEE Transactions on Smart Grid, 2*(4), 844–864. doi:10.1109/TSG.2011.2162431.
4. Brown, J., Du, X., & Guizani, M. (2009). Efficient rekeying algorithms for wimax networks. *Security and Communication Networks, 2*(5), 392–400. doi:10.1002/sec.124.
5. Camtepe, S., & Yener, B. (2007). Combinatorial design of key distribution mechanisms for wireless sensor networks. *IEEE/ACM Transactions on Networking, 15*(2), 346–358. doi:10.1109/TNET.2007.892879.
6. Canh N. T., Truc, P., Hai, T. H., Hung, L. X., Lee, Y. K., & Lee, S. (2009). Enhanced group-based key management scheme for wireless sensor networks using deployment knowledge. In: *6th IEEE consumer communications and networking conference, 2009 (CCNC 2009)* (pp. 1–5). doi:10.1109/CCNC.2009.4784870.
7. Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. *ACM Computing Surveys, 41*, 15:1–15:58. doi:10.1145/1541880.1541882.
8. Chen, J., Kher, S., & Somani, A. (2006). Distributed fault detection of wireless sensor networks. In: *Proceedings of the 2006 workshop on dependability issues in wireless ad hoc networks and sensor networks, DIWANS '06* (pp. 65–72). New York, NY: ACM. doi:10.1145/1160972.1160985.
9. da Silva, A. P. R., Martins, M. H. T., Rocha, B. P. S., Loureiro, A. A. F., Ruiz, L. B., & Wong, H. C. (2005). Decentralized intrusion detection in wireless sensor networks. In: *Proceedings of the 1st ACM international workshop on quality of service & security in wireless and mobile networks, Q2SWinet '05* (pp. 16–23). New York, NY: ACM. doi:10.1145/1089761.1089765.
10. Di Pietro, R., Mancini, L., Law, Y. W., Etalle, S., & Havinga, P. (2003). Lkhw: A directed diffusion-based secure multicast scheme for wireless sensor networks. In: *International conference on parallel processing workshops* (pp. 397–406).
11. Du, W., Deng, J., Han, Y. S., Varshney, P. K., Katz, J., & Khalili, A. A pairwise key predistribution scheme for wireless sensor networks. *ACM Transaction on Information and System Security, 8*(2), 228–258. doi:10.1145/1065545.1065548.
12. Eschenauer, L., & Gligor, V. D. (2002). A key-management scheme for distributed sensor networks. In: *Proceedings of the 9th ACM conference on computer and communications security, CCS'02* (pp. 41–47). New York, NY: ACM. doi:10.1145/586110.586117.
13. Hardjono, T., & Dondeti, L. R. (2003). *Multicast and group security*. Norwood, MA: Artech House, Inc. ISBN: 1580533426.
14. Harney, H. (1999). Logical key hierarchy protocol (lkh). Internet draft. Draft-harney-sparta-lkhp-sec-00.

15. Hernández-Serrano, J., Pegueroles, J., & Soriano, M. (2008). Shared self-organized gkm protocol for manets. *Journal of Information Science and Engineering, 24*(6), 6.
16. Huang, Y., & Lee, W. (2003) A cooperative intrusion detection system for ad hoc networks. In: *Proceedings of the 1st ACM workshop on security of ad hoc and sensor networks* (p. 147). New York, NY: ACM.
17. Hui, J., & Thubert, P. (2011). RFC 6282—Compression format for IPv6 datagrams over IEEE 802.15.4-based networks. *Internet Engineering Task Force (IETF)—Standards Track*. http://datatracker.ietf.org/doc/rfc6282/.
18. IEEE Computer Society. (2006). Wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (WPANs). In: *IEEE standards publication delivered in electronic form*.
19. Kim, Y., Perrig, A., & Tsudik, G. (2004). Tree-based group key agreement. *ACM Transactions on Information and System Security, 7*(1), 60–96. doi:10.1145/984334.984337.
20. Krishnamachari, B., & Iyengar, S. (2004). Distributed bayesian algorithms for fault-tolerant event region detection in wireless sensor networks. *IEEE Transactions on Computers, 53*(3), 241–250. doi:10.1109/TC.2004.1261832.
21. Li, X., Wang, Y., & Frieder, O. (2002). Efficient hybrid key agreement protocol for wireless ad hoc networks. In: *Proceedings of the eleventh IEEE international conference on computer communications and networks, 2002*. (pp. 404–409).
22. Li, X. S., Yang, Y. R., Gouda, M. G., & Lam, S. S. (2001). Batch rekeying for secure group communications. In: *WWW '01: Proceedings of the 10th international conference on World Wide Web* (pp. 525–534). New York, NY: ACM Press. doi:10.1145/371920.372153.
23. Liu, D., & Ning, P. (2003). Establishing pairwise keys in distributed sensor networks. In: *CCS '03: Proceedings of the 10th ACM conference on computer and communications security* (pp. 52–61). New York, NY: ACM Press. doi:10.1145/948109.948119.
24. Lopez, J., & Zhou, J. (2008). *Wireless Wireless sensor network security*. Amsterdam, The Netherlands: IOS Press.
25. Meng, J., Li, H., & Han, Z. (2009). Sparse event detection in wireless sensor networks using compressive sensing. In: *43rd Annual conference on information sciences and systems (CISS 2009)*.
26. Mittra, S. (1997). Iolus: A framework for scalable secure multicasting. *SIGCOMM Computer Communication Review, 27*(4), 277–288. doi:10.1145/263109.263179.
27. Molisch, A., Balakrishnan, K., Chong, C., Emami, S., Fort, A., Karedal, J., et al. (2006). IEEE 802.15.4a channel model-final report. *IEEE P, 15*, 802–1504.
28. Pegueroles, J., & Rico-Novella, F. (2003). Balanced batch lkh: new proposal, implementation and performance evaluation. In: *Proceedings of the eighth IEEE international symposium on Computers and Communication, 2003 (ISCC 2003)* (Vol. 2, pp. 815–820). doi:10.1109/ISCC.2003.1214218.
29. Pegueroles, J., Rico-Novella, F., Hernández-Serrano, J., & Soriano, M. (2003a). Adapting GDOI for balanced batch-LKH "draft-ietf-msec-gdoi-batch-lkh-00.txt". Internet draft. Work in progress.
30. Pegueroles, J., Rico-Novella, F., Hernández-Serrano, J., & Soriano, M. (2003b). Improved lkh for batch rekeying in multicast groups. In: *IEEE International conference on information technology research and education (ITRE)*. New Jersey: E.E.U.U.
31. Perrig, A., Szewczyk, R., Tygar, J. D., Wen, V., & Culler, D. E. (2002). Spins: security protocols for sensor networks. *Wireless Networks, 8*(5), 521–534. doi:10.1023/A:1016598314198.
32. Steiner, M., Tsudik, G., & Waidner, M. (1996). Diffie-hellman key distribution extended to group communication. In: *CCS '96: Proceedings of the 3rd ACM conference on computer and communications security* (pp. 31–37). New York, NY: ACM. doi:10.1145/238168.238182.
33. Tse, D., & Viswanath, P. (2005). *Fundamentals of wireless communication*. Cambridge University Press. doi:10.2277/0521845270.
34. Wallner, D., Harder, E., & Agee, R. (1998). Key management for multicast: Issues and architectures. *RFC, 2627*.
35. Wander, A. S., Gura, N., Eberle, H., Gupta, V., & Shantz, S. C. (2005). Energy analysis of public-key cryptography for wireless sensor networks. In: *PERCOM '05: Proceedings of the third IEEE international conference on pervasive computing and communications* (pp. 324–328). Washington, DC: IEEE Computer Society. doi:10.1109/PERCOM.2005.18.
36. Wang, T. Y., Chang, L. Y., Duh, D. R., & Wu, J. Y. (2008). Fault-tolerant decision fusion via collaborative sensor fault detection in wireless sensor networks. *IEEE Transactions on Wireless Communications, 7*(2), 756–768. doi:10.1109/TWC.2008.060653.
37. Wang, Y., Wang, X., Xie, B., Wang, D., & Agrawal, D. P. (2008). Intrusion detection in homogeneous and heterogeneous wireless sensor networks. *IEEE Transactions on Mobile Computing, 7*, 698–711. doi:10.1109/TMC.2008.19.
38. Zhang, Y., Meratnia, N., & Havinga, P. (2010). Outlier detection techniques for wireless sensor networks: A survey. *IEEE Communications Surveys Tutorials, 12*(2), 159–170. doi:10.1109/SURV.2010.021510.00088.
39. Zhu, S., Setia, S., & Jajodia, S. (2006). Leap+: Efficient security mechanisms for large-scale distributed sensor networks. *ACM Transactions on Sensor Network, 2*(4), 500–528. doi:10.1145/1218556.1218559.

Author Biographies

Juan Hernández-Serrano was born in Salamanca (Spain) in 1979. He received the M.S. degree in Electrical Engineering emphasis in Telecommunication in 2002, and the Ph.D. degree in 2008, both from the Universitat Politècnica de Catalunya (UPC). In 2002 he joined the Information Security Group (ISG) within the Telematics Services Research Group (SERTEL) at the Department of Telematics Engineering of the UPC. He currently works as assistant professor at the Castelldefels School of Technology (EPSC) in the UPC. His research interests include security for large deployment of sensor networks, autonomous cognitive networks, smart grids, digital forensics and e-voting.

Juan Vera-de-Campo doing his Ph.D. on Security in Peer-to-peer networks at the Information Security Group of the Universitat Politècnica de Catalunya, Barcelona. His fields of interest include resource-constrained devices, distributed systems, peer-to-peer networks, document location and security, especially the protection of the privacy of the users in distributed environments.

Josep Pegueroles was born in Tortosa (Spain) in 1974. He received the M.S. degree in Electrical Engineering emphasis in Telecommunication in 1999, and the Ph.D. degree in 2003, both from the Universitat Politècnica de Catalunya (UPC). In 1999 he joined the Information Security Group (ISG) within the Telematics Services Research Group (SERTEL) at the Department of Telematics Engineering of the UPC. He currently works as assistant professor at the Telecommunications Engineering School in Barcelona (ETSETB). His research interests include security for multimedia networked services and secure group communications.

Carlos Gañán is a PhD student in Telematics at the Universitat Politècnica de Catalunya (UPC). He is an active researcher in security for vehicular ad hoc networks, secure video streaming, reputation and identification in peer-to-peer networks, and certification in PKI. Nowadays, he is working on the development of collaborative certificate status checking mechanisms focusing on guarantying an always-on availability regardless the current status of the certificate emitters.