Pers Ubiquit Comput
DOI 10.1007/s00779-011-0365-5

ORIGINAL ARTICLE

RiSeG: a ring based secure group communication protocol for resource-constrained wireless sensor networks

**Omar Cheikhrouhou · Anis Koubâa ·
Gianluca Dini · Mohamed Abid**

Received: 10 December 2010 / Accepted: 5 March 2011

Abstract Securing group communication in wireless sensor networks has recently been extensively investigated. Many works have addressed this issue, and they have considered the grouping concept differently. In this paper, we consider a group as being a set of nodes sensing the same data type, and we alternatively propose an efficient secure group communication scheme guaranteeing secure group management and secure group key distribution. The proposed scheme (RiSeG) is based on a logical ring architecture, which permits to alleviate the group controller's task in updating the group key. The proposed scheme also provides backward and forward secrecy, addresses the node compromise attack, and gives a solution to detect and eliminate the compromised nodes. The security analysis and performance evaluation show that the proposed scheme is secure, highly efficient, and lightweight. A comparison with the logical key hierarchy is preformed to prove the rekeying process efficiency of RiSeG. Finally, we present the implementation details of RiSeG on top of TelosB sensor nodes to demonstrate its feasibility.

Keywords Secure group communication · Wireless sensor networks · Security · Key management · Group management

1 Introduction

Wireless sensor networks (WSNs) have emerged as a promising technology useful for a wide range of civilian applications such as environment monitoring, target tracking, healthcare services, etc. [1–4]. To note, a WSN is made up of several autonomous and compact devices called sensor nodes. The latter are densely spread in the monitored area, and wirelessly communicate in order to self-organize into a multi-hop network, collaborate in the sensing activity and forward the acquired information toward one or more users. WSNs are usually deployed for monitoring several types of data, and therefore, a sensor node is, generally, equipped with a diversity of sensors (temperature, humidity, light, etc.) [5]. In addition, sensor nodes charged with sensing the same data type may want to form a logical group, and consequently, data circulated in one group must not be revealed by nodes alien to that group. Group communication might be needed when the group controller wishes to send the same commands or requests to all group members. Similarly, group controller may wish to dynamically reprogram or retask group members, namely reset their trigger thresholds, recalibrate the sensors, etc. [6]. Moreover, group members may collaborate together to produce aggregated information. This collaboration requires a secure communication among group members.

O. Cheikhrouhou (✉) · M. Abid
CES Research Lab, National School of Engineers of Sfax,
University of Sfax, Sfax, Tunisia
e-mail: omar.cheikhrouhou@isetsf.rnu.tn

M. Abid
e-mail: mohamed.abid@enis.rnu.tn

A. Koubâa
CISTER Research Unit, Polytechnic Institute of Porto
(ISEP/IPP), Portugal, Al-Imam Mohamed Bin Saud University,
College of Computer Science and Information Systems,
Riyadh, Saudi Arabia
e-mail: aska@isep.ipp.pt

G. Dini
Dipartimento di Ingegneria della Informazione,
University of Pisa, Largo Lazzarino 1, 56100 Pisa, Italy
e-mail: g.dini@iet.unipi.it

1.1 Motivation

Several research works have addressed the secure group communication problem in WSNs. However, the proposed solutions consider a restrictive definition of a group. In fact, most of the related works have considered a group as being a set of nodes physically close to each other. Moreover, they consider the whole network as a single group managed by the base station. Nevertheless, grouping appears to be more general and sophisticated than such particular cases. Hence, this paper proposes to define a group as a set of nodes that sense the same data type and that are not necessarily close to each other. Thus, in a single network, it is possible to have several groups each of which managed by a sensor node playing the role of a group controller. As a matter of fact, there are several potential applications, such as home automation, environment monitoring in which several nodes are responsible for controlling diverse parameters, e.g., temperature, light, humidity, etc. Each set of nodes forms a group in which they communicate securely. This group formation concept gives flexibility in defining the security policy inside each group. As an illustration, one can cite the example of a WSN deployed to sense weather temperature and pollution rate produced by factories. Thus, while the temperature information can be used to deliver a paid service for users, the pollution rate information can be used to control factories and take decisions based on the sensed value (e.g., put taxes as a function of the pollution rate). The temperature information should then be delivered exclusively to the subscriber users. Therefore, an attacker may try to reveal information (in order not to pay subscription fees and get information for free), but he/she has no interest in injecting false temperature values. As a result, we have to apply confidentiality to the temperature group without having to care about authentication. Thus, messages exchanged between group members (sensor nodes) must be encrypted. However, in the case of pollution-related data, information can be sent clearly as it is not confidential information; yet the sensed value must be authenticated lest an attacker would try to decrease the real value of pollution rate. Therefore, it appears exclusively necessary to apply authentication in such a case. Thus, messages exchanged between group members must be authenticated using, for instance, a message authentication code (MAC). To summarize, dividing the network into multiple groups has some advantages, namely:

- Flexibility: the security services will be flexible and adaptive as it is possible to apply a security policy per group. For example, it will be possible to apply encryption for some information while apply authentication for the others.
- Security: a node pertaining to one group does not reveal information circulated in other groups. This increases the level of security inside the network as if a group is compromised the other groups remain secure.
- Scalability: dividing the network into groups promotes the network scalability. In fact, the burden task of maintaining network parameters (such as security parameters) at the base station is distributed among group controllers.

1.2 Contribution

In this paper, we propose a secure group communication mechanism for wireless sensor networks, whereby a group is defined as being a set of nodes collaborating to collect the same sensory information. The proposed scheme allows protecting data using a group key, which is shared among group members and maintained by the group controller. This key is updated whenever the group membership changes for the sake of providing forward and backward secrecy. One of the key contributions of this paper is the proposal of a logical ring topology that permits to alleviate the group controller task and render the rekeying process more scalable.

The remainder of this paper is organized as follows. In Sect. 2, we present works relevant to secure group communication. As for Sect. 3, it describes our network model and assumptions. Then, in Sect. 4, we present our secure group communication scheme. In Sect. 6 and Sect. 5, we expose the security analysis as well as the performance analysis. Then, in Sect. 7, we present the performance results of the proposed scheme when implemented in a real-world platform using TelosB motes. Finally, we end up by concluding and suggesting some further future works.

2 Related works

Group communication security in WSNs is a challenging issue that has been addressed throughout several research works [6–18]. In [6], the authors have proposed SLIMCAST: a secure level key infrastructure for multicast to protect data confidentiality via hop-by-hop re-encryption and mitigate the DoS-based flooding attack through an intrusion detection and deletion mechanism. The SLIMCAST protocol divides a group routing tree into levels and branches in a clustered manner. Communications among nodes in each level of each branch of the group tree are protected by a level key such that only the local level key is updated during a joining or a leaving process. The scheme presents a low communication overhead and power consumption and is also scalable. However, the performance is degraded (i.e., high power consumption) when membership changes are massive. In [17], the authors have proposed SeGCom a secure group communications mechanism for cluster-tree wireless sensor networks. The scheme uses µTESLA [19] to broadcast the group controller identity. However, µTESLA requires synchronization of nodes, which is a hard task to achieve in a WSN [20]. Moreover, the scheme did not explain how the authentication process is done and it presents an communication overhead. The authors in [11] have proposed to form a network with multiple base stations, each of which is responsible for dynamically forming a group composed of three types of sensor nodes classified according to their ability to communicate with the base stations. They have also proposed a scheme using a key tree to manage group members as they join or leave the group. However, the authors did not provide details as regards the group re-keying process. As the group key management presents the cornerstone of a secure group communication scheme, several papers have concentrated on the re-keying process. Re-keying occurs whenever a node joins or leaves the group. In LEAP (Localized Encryption and Authentication Protocol) [9], the authors have proposed a key management protocol for sensor networks that are designed to support in-network processing, while at the same time restricting the security impact of a node compromise to the immediate network neighborhood of the compromised node. LEAP supports the establishment of four types of keys for each sensor node—an individual key shared with the base station, a pairwise key shared with another sensor node, a cluster key shared with multiple neighboring nodes, and a global key shared by all the nodes in the network. For the update of the global key, LEAP assumes the use of a routing protocol in which the nodes are organized into a spanning tree. However, this assumption limits the deployment of the scheme. Moreover, the scheme rests on the µTesla scheme [19], which requires synchronization between nodes. In [8, 10], the authors have proposed an algorithm to compute a group key in a collaborative manner. The algorithm is based on the multi-party Diffie-Hellman protocol [21]. However, the proposed algorithm requires many exponentially complex operations, which turn it out to be unpractical for sensor networks. In [12–15, 18, 22], the authors have proposed a centralized group re-keying scheme based on a logical key-tree hierarchy for WSNs. The basic scheme is the logical key hierarchy (LKH) [12] proposed to reduce the rekeying messages' number from $O(n)$ to $O\log(n)$, using a tree structure for storing keys. The root of the tree serves as the key distribution center (KDC), while each leaf represents a node. Each leaf stores the set of keys belonging to its direct ancestors up to the KDC. The reason behind applying a tree structure is to increase the re-keying efficiency. However, the energy required for re-keying is approximately logarithmic in the group size. The main contribution of [13] consists of extending the LKH scheme in the context of directed diffusion [23], where the number of rekeying messages is still logarithmic in the group size. Dini et al. [14] have, in turn, improved key authentication by means of key chains, a mechanism derived from Lamport's one-time key and based on hash functions. Furthermore, Dini et al. [15, 22] have later extended the logical key-tree hierarchy into a key graph in order to efficiently support backward and forward security in systems comprising several, possibly overlapping, groups. However, the storage cost required by their scheme exceeds the available resources of a sensor node and, therefore, the scheme cannot be applied to groups with a resource-constrained group controller. The topological key hierarchy (TKH) scheme [18] allows to reduce the communication cost of the LKH rekeying messages delivery by mapping the logical key tree to the physical topology. The idea is to construct a key tree that reflects the physical topology of the network. However, TKH does not face with key authentication. In this paper, however, we propose a new secure group communication mechanism based on a logical ring topology, which allows for a scalable re-keying process. The scheme distributes the group management task among group members, thus, eliminating the need for a plentiful group controller. Moreover, the node compromise attack has been addressed, with a proposed solution to detect and discard the compromised nodes.

3 Network model, assumptions, and requirements

This section is devoted to the presentation of the network model to which the proposed secure group communication scheme is applied as along with the considered assumptions and requirements.

3.1 Network model

It is worth noting that a wireless sensor network maintained by a base station is considered in this study. As for the information within the network, is routed using a routing protocol such as Ad hoc On Demand Distance Vector routing algorithm (AODV) [24] or Dynamic Source Routing algorithm (DSR) [25]. In addition, the following types of nodes have also been considered:

- The base station (BS): is responsible for securing the whole network. It maintains a table containing the group controller addresses corresponding to each group. The BS also supervises the group controller

activity and maintains a *blacklist* containing the identity of compromised nodes. These nodes will not be allowed to join any group in the future and are, therefore, excluded from the network.
- The Group Controller (GC): is a node responsible for maintaining the security inside its group. It also stores a table containing the list of group members classified according to their joining time. The GC controls the group members' activity, and in the case of a compromised node, it sends a notification message to the BS. The latter adds the node to the *blacklist*. To note, no security property has been assumed for the GC.
- The End Device (ED): is a node which belongs to one or multiple groups. For each group, it maintains the next and previous hop (in the logical ring) addresses.

3.2 Assumptions

In the present work, the following assumptions have been formulated:

- The base station is secure and able to detect all compromised GC nodes. Detection of compromised GC nodes can be actually achieved by means of an intrusion detection system (IDS) such as [26–29].
- The GC can detect all compromised members as it has control over the members attached to its group. The GC may actually use the same IDS tools as the BS.
- Each node is identified by a unique address and can belong to more than one group.
- Each group has a unique group identifier, which represents the sensory information corresponding to this group. These group identifiers are known to all nodes. This can be done by loading the group identifiers to nodes at the deployment phase.
- The base station maintains a *blacklist* containing a list of compromised nodes together with their addresses. These nodes are prevented from joining any group and, therefore, excluded from the network.
- Each node periodically sends to its corresponding group controller a HELLO message confirming its presence. This enables to detect compromised nodes. Indeed, in case of a compromise attack, an attacker seizes a node from the sensor network, connects this node to his laptop, extracts the stored data, puts new data/behavior, and takes control over that node [30, 31]. This means that a compromise attack necessitates a certain period of time to be executed and, therefore, one might well assume that a node is compromised whenever it does not prove its presence, by sending some HELLO messages, during a threshold time period. This assumption seems logical, as an inactive node for a threshold time means that either the node is compromised or that it has failed. In both cases, the node is evicted from the group and, therefore, must be added to the *blacklist*.

3.3 Security requirements

In what follows, the requirements to be achieved by a secure group communication scheme have been presented:

- Nodes belonging to the same group must communicate securely and their exchanged information must not be revealed to non-member nodes even if they belong to the same network.
- A node may belong to more than one group. However, it must store a per-group profile containing the GC address, the group key, the next and previous node in the logical ring, etc.
- Compromised nodes must be ejected from the group as soon as they are detected.
- Nodes non-member of the group collaborate to route data. Yet, data must be confidential to each group (intermediate nodes forward data without being able to reveal their value).
- Both backward and forward secrecy must be achieved. Backward secrecy means that a node joining the group must not reveal previous exchanged information. Forward secrecy means that a node leaving the group must not reveal future exchanged information.
- Security parameters' maintenance such as the re-keying process must be lightweight and effective.

4 RiSeG: the logical ring based secure group communication scheme

In this section, we present our proposed secure group communication scheme. It is composed of two parts: (1) the logical ring management and (2) the group membership management.

4.1 Logical ring management

One of the most important challenges encountered when designing a secure group communication scheme is scalability. In fact, the re-keying process needed in the case of membership change represents an overhead as it requires, when using unicast, $O(n)$ messages to be sent by the GC, where n denotes the number of group members. In our work, this problem has been solved by constructing a logical ring topology. This logical ring permits to distribute and divide the task of sending a message to all members. Indeed, with the help of this logical ring topology, information is circulated from node to node until it reaches the information source. Therefore, the GC just needs to send $O(1)$ messages instead of $O(n)$ messages. The logical ring is constructed as follows. The ring initially contains the GC that plays the role of the ring head (Fig. 1a). Then, each new node is added to the ring queue (tail), upon request to join the group (Fig. 1b). The logical ring topology is maintained by the GC. Note that the GC maintains all the group members' addresses. Each node only maintains its next and previous hop addresses. For instance, in Fig. 1c, node N2 maintains the address of node N1 as its previous hop and the address of node N3 as its next hop. In the case of a joining process, the GC informs the newly joining node by its previous hop, which is the latest joined node, and informs the latter to update its next node address to this newly joining node. Taking the example of Fig. 1c, d, after the join of node N4, the GC sends to N4 the address of node N3 as its previous hop. Note the next node of N4 is the GC. Moreover, the GC sends a message to node N3 in order to update its next node to node N4.

In the case of a leaving process, the leaving node must also be removed from the logical ring. This means that the GC informs the leaving node's next node (respectively the leaving node's previous node) to change its previous (respectively next) hop address. For instance, if node N2 of Fig. 1c is leaving the group, the group controller informs N3 to change its previous hop to N1 and informs N1 to change its next node to N3.

4.2 Group membership management

In this section, we describe the necessary operations needed to maintain the group membership such as: the group creation, the group join, the group leave, the group controller switching, and the group controller leaving. Firstly, we begin by presenting the necessary parameters loaded in nodes at the pre-deployment phase.

4.2.1 Pre-deployment phase

As in [17], we propose to apply the key pre-distribution scheme proposed by Blundo et al. [32] in order to share a symmetric key between each pair of nodes. The network administrator chooses a t degree bi-variate polynomial over a finite field F_q: $f(x, y) = \sum_{i=0}^{i=t} \sum_{j=0}^{j=t} a_{i,j} x^i y^j$. The value of q is a prime number that is large enough to accommodate a cryptographic key. Then, the administrator loads in each node Ni the polynomial $f(x, Ni)$. The function f is symmetric. This means that, when two nodes Ni and Nj wish to share a pairwise key, each of them computes $K_{Ni,Nj} = f(Ni, Nj) = f(Nj, Ni)$.

Moreover, for the signature purpose, we use the elliptic curve cryptography [33, 34], so that each node is preloaded with the domain parameters needed to compute and verify the (elliptic curve digital signature algorithm) ECDSA [33]. The domain parameters are the six-tuple $T = (p, a, b, G, n, h)$, where p is a prime number, a and b are two points from the primary field $F_p (a, b \in F_p)$

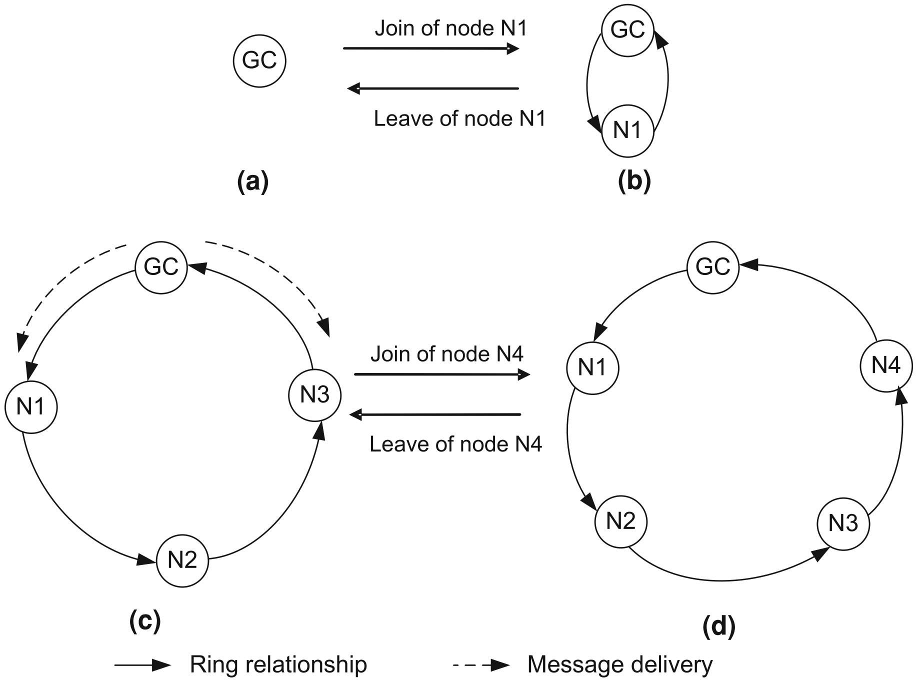


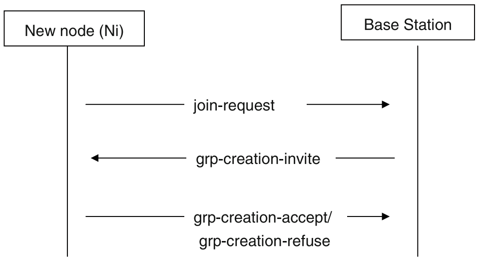
Fig. 1 Logical ring update in the case of a joining/leaving process

defining the curve, G a base point on the curve with order n and cofactor h.

4.2.2 Group creation

The group creation process is executed when a node wishes to join a non-existent group. In fact, when a node with identity Ni wishes to join a group identified by Gid, it sends a *join-request* message to the base station (Fig. 2). The *join-request* message contains the node identity (Ni), the group identifier (Gid) to which the node wishes to join, a fresh random number (*nonce*), and a message authentication code (MAC). The nonce allows to avoid replay attacks, while the MAC allows to avoid identity usurpation attacks (Sect. 5).

Upon receiving this message, the base station verifies the sender node's validity and the message authenticity. The validity of the sender node means that the node does not belong to the *blacklist* and is, therefore, considered as not compromised. The message authenticity is verified based on the MAC field. In fact, the received MAC is compared to the locally computed one using the pairwise key ($K_{BS,Ni}$), and the message is considered authentic if both MACs are equal. Otherwise, the base station ignores the request. After successfully verifying the message, the base station replies to the node Ni by sending a *grp-creation-invite* message. This message contains the node nonce ($nonce_{Ni}$), a new nonce generated by the BS ($nonce_{BS}$) and is also protected by a MAC. Therefore, the node is invited to become the GC of this new group. Once the node accepts to be a GC, it replies by sending a *grp-creation-accept* message. A node might refuse to become a GC, for instance, if it has not enough resources to achieve the GC task. In this case, it replies by sending a *grp-creation-refuse* message. Both messages contain the BS nonce in order to avoid any replay attack and are also protected by a MAC. Figure 2 illustrates the group creation process, where $MAC(m, K_{BS,Ni})$ is a message authentication code computed over the current message and using the pairwise key $K_{BS,Ni}$.



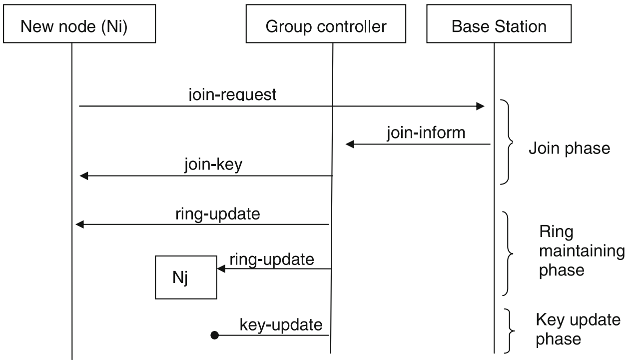
Ni→BS: join-request = Ni, Gid, nonce$_{Ni}$, MAC(m, K$_{BS, Ni}$)
BS→Ni: grp-creation-invite= Gid, nonce$_{Ni}$, nonce$_{BS}$, MAC(m, K$_{BS, Ni}$)
Ni→BS: grp-creation-accept= Ni, Gid, nonce$_{BS}$, MAC(m, K$_{BS, Ni}$)
or : grp-creation-refuse= Ni, Gid, nonce$_{BS}$, MAC(m, K$_{BS, Ni}$).

Fig. 2 Message exchanges in a group creation process

Following the new group creation, the GC and the BS agree on a sequence number *seqNbr*. This sequence number is incremented on each sent message enabling to avoid any replay attack, as will be explained later. Moreover, in order to sign subsequent key-update messages, the group controller needs a public/private key. For this purpose, the GC selects a random integer d in the interval $[1, n - 1]$ and then computes $Q = d \times G$. The tuple (d, Q), respectively, represents the GC's private and public keys.

4.2.3 Group join

The group join process is executed when a node wishes to join an existing group. Upon receiving a *join-request* message to a group that already exists, the base station verifies the sender node's validity along with the request authenticity. Hence, if the node is proved to be valid and its *join-request* message passes the authenticity test (the MAC is valid), the base station sends a *join-inform* message to the GC, informing it that a new node has joined the group (Fig. 3). The *join-inform* message contains the node identity (Ni) and is protected by a sequence number (*seqNbr*) to avoid any replay attacks, and by a MAC to avoid usurpation of the base station identity attacks (Sect. 5) After testing the validity of the message, the GC computes a new group key GK' and sends it out encrypted to Ni using the pairwise key ($K_{GC,Ni}$) in a *join-key* message. The *join-key* message contains also the GC public key Q, which will



---Join phase---
Ni→BS: join-request= Ni, Gid, nonce$_{Ni}$, MAC(m, K$_{BS, Ni}$)
BS→GC: join-inform= Ni, Gid, seqNbr, MAC(m, K$_{BS, GC}$)
GC→Ni: join-key= Gid, {GK'}_K$_{GC,Ni}$, Q, MAC(m, K$_{GC,Ni}$)
---Ring maintain phase---
GC→Ni: ring-update= Gid, nextNode, prevNode, MAC(m, K$_{GC,Ni}$)
GC→Nj: ring-update= Gid, nextNode, prevNode, MAC(m, K$_{GC,Nj}$)
---Key update phase---
GC→*: key-update= Gid, seqNbr,{GK'}_GK ,sign

Fig. 3 Message exchanges in a group join process serve for signature verification in subsequent *key-update* messages.

Moreover, the group controller updates the logical ring topology by sending *ring-update* messages. In fact, the GC sends to *Ni* a *ring-update* message containing the previous hop (we suppose *Nj*) as well as the next hop (the GC), and sends to *Nj* a *ring-update* message in order to update its next node to *Ni*. Then, the group controller launches the *key-update* process. Figure 3 summarizes the group join process.

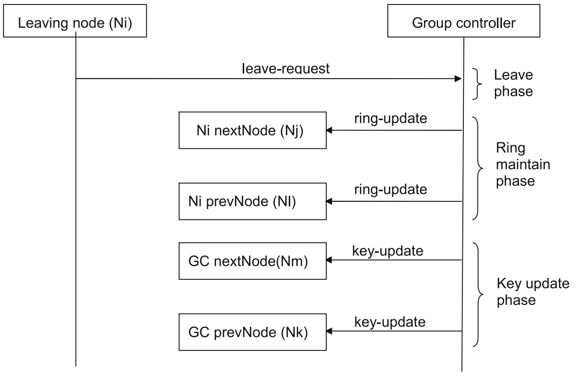
4.2.4 Group leave

The leaving process occurs when a node wishes to leave the group, breaks down, or is compromised. In the first case, the GC is informed through a *leave-request* message (Fig. 4). In the both remaining cases, the GC is informed through the inactivity of the leaving node, and the node is then considered as compromised. Consequently, the GC sends to the BS a notification to add this inactive node to the *blacklist*. For the sake of achieving forward secrecy, the GC must update the group key and the logical ring topology. For this reason, the group controller sends a *ring-update* message to both the *nextNode* and the *prevNode* of the leaving node, in order to, respectively, update their previous and next hops. To illustrate the leaving process, let us consider, for instance, the leave of node *N2* in Fig. 1. On receiving a leave-request message, the GC checks the message validity. If the message is valid, the GC sends two *ring-update* messages: one *ring-update* message to node *N3* to update its previous hop and one ring-update message to *N1* to update its next hop. Then, the group controller computes a new group key and sends it to its next and previous hops (*N1* and *N4*) in the logical ring. This key is sent encrypted using the pairwise key. Moreover, it is protected by a signature to verify its authenticity by the group members and to avoid that a node injects a false *key-update* message.

4.2.5 Key update

The key-update process is to be launched after each join or leave process or when the *GC* wishes to update the group key for security purposes. The *key-update* message contains the group identity (*Gid*), the encrypted new group key (GK') and is protected by a sequence number along with a signature. The sequence number ensures the freshness, and the signature ensures the authentication of the *key-update* message.

To protect the group key from eavesdropping, the GC protects it by means of encryption. In the case of a join operation, the GC can use the current group key to encrypt the new one, and then broadcasts the *key-update* message to all members. However, in a leave operation, the leaving node knows the current group key and, therefore, this key cannot be used for encryption as this would break the requirement of forward secrecy. Therefore, there is no choice but to use pairwise keys for encryption.



---Leave phase---

Ni→GC: leave-request= Ni, Gid, seqNbr, MAC(m, $K_{GC,\ Ni}$)

(Or the GC detects the inactivity of Ni)

---Ring maintain phase---

GC→Ni’s nextNode: ring-update= Gid, nextNode, prevNode, MAC(m, $K_{GC,Nj}$)

GC→Ni’s prevNode: ring-update= Gid, nextNode, prevNode, MAC(m, $K_{GC,Nl}$)

---Key update phase---

GC→GC’s nextNode: key-update= Gid, seqNbr,{GK’}_$K_{GC,Nm}$,sign

GC→GC’s prevNode: key-update= Gid, seqNbr,{GK’}_$K_{GC,Nk}$,sign

Fig. 4 Message exchanges in a group leave process

To alleviate the group key distribution task, the GC will use the ring topology. Actually, as the group controller maintains a double-direction ring topology, it sends the message in both directions. On receiving a key-update message, a node first, verifies the sequence number and the signature fields. The sequence number must be greater than the current one, otherwise the message will be considered as old and already processed and must consequently be ignored. In the case of a valid sequence number and a valid signature, the node processes the message in the following way. If the key-update message was received from the previous node (respectively, next node), the node decrypts the group key using the pairwise key shared with the previous node (respectively, next node), and then re-encrypts the group key using the pairwise key shared with the next node (respectively, previous node), and, finally, transmits the message to the node.

4.2.6 Group controller switching

When the group controller wishes to leave the group controller responsibilities, it sends the group management information to the upstream node. If the latter accepts to be a GC, it sends a *GC-confirm* message to the base station indicating that it is the new GC in order to update its table. Noteworthy, the base station maintains a table indicating the GC address of each group. All messages are sent securely using the pairwise keys. If the GC's upstream node refuses to be the new group controller, it forwards the group management information to the next hop in the logical ring. This process of forwarding the group management information message will be repeated until a node accepts to be the new GC, otherwise the message reaches its origin (the current group controller), in which case, the group will be destroyed.

4.2.7 Group controller leaving

The normal operation performed by the GC consists in switching its functionality to another node before leaving the group, hence, its leaving is similar to that of any normal node. However, the actual problem is what occurs when the GC is compromised or crashed. To overcome this problem, two solutions are conceivable: either to make a backup GC or store the group management information in the base station. In the former solution, a normal group member maintains a copy of the group management information, and in the case of GC compromise, this node takes the role of the GC. As for the latter solution, the base station elects a group member to which it sends the group management information.

5 Security analysis and discussion

This section is allotted to discuss the merits of the different cryptographic tools used in the proposed scheme and analyze its security. In the design of our scheme, a nonce has been applied for the purpose of preventing replay attacks, along with a message authentication code (MAC) intended to avoid impersonation attacks, as well as a signature aiming at providing authentication of the rekeying messages.

The proposed secure group communication scheme provides the following security services:

- Replay attack robustness: in the proposed scheme, intercepted messages cannot be replayed by an attacker as all sent messages are proved to be fresh through a *nonce*. In addition, attackers cannot modify the value of the *nonce* as the message is protected by a MAC.
- Impersonation attack robustness: all sent messages are protected by a MAC computed over the identity of the sender node. This prevents attackers from gaining access to a group during the group creation and group join processes.
- Authentication of the rekeying messages: *key-update* messages carry a signature computed by the GC. This signature proves that the key is sent by the GC and, therefore, precludes an attacker from injecting a fake group key.
- Backward and forward secrecy: when a new node joins the group, the group controller generates a new key and delivers it to the group members. Therefore, the new node has no means to decrypt the previously exchanged messages. Moreover, when a node leaves the group, the group controller generates a new key. This key will be sent by unicast and, therefore, the leaving node will be unable to decrypt the future sent messages.
- Mutual authentication: our scheme achieves mutual authentication since not only the base station authenticates the requesting node, but also the node authenticates the base station. The authentication of messages sent by the base station is critical. In fact, if we do not authenticate the grp-creation-invite message, an attacker can impersonate the base station by sending this message even when the group exists. This scenario will disturb the network operation as there will be a creation of multiples copies of the same group, each of which is composed of a single node.
- Node compromise robustness: based on monitoring the activity of nodes, our scheme can detect compromised nodes and is, therefore, able to discard them from the network.

6 Performance analysis

This section is allotted to present the analytical performance evaluation of the proposed scheme. The performance evaluation does not consider the base station as it is powerful and does not present constrained resources. The performance evaluation criteria are the storage cost, the computation cost and the communication cost. It is worth starting by presenting the different notations used throughout this section in Tables 1 and 2

6.1 Storage cost

The storage cost is computed as the number of bytes that the sensor node (group controller or group member) have to store. Generally, this storage cost is introduced by the storage of different parameters and keys necessary to the function of the RiSeG scheme. The proposed secure group communication scheme does not require much memory overhead. In fact, due to Blundo et al.'s key distribution technique, each sensor node has to store a polynomial function which occupies $(t + 1)logq$ storage space, where t stands for the degree of the polynomial and $log(q)$ represents the size of the keys [35]. Moreover, each member has

Table 1 Computation cost parameters

Parameter	Signification
C_{enc}	The computation cost needed to compute the encryption of the group key
C_{dec}	The computation cost needed to compute a decryption operation on the group key
C_{sign}	The computation cost needed to generate a signature
C_{verif}	The computation cost needed to verify a signature
C_{kg}	The computation cost needed to generate a group key
C_{kc}	The computation cost needed to compute the pairwise key
C_{mac}	The computation cost needed to compute a MAC

Table 2 Communication cost parameters

Parameter	Signification
$\|m\|$	The size in bits of the message *m*
mjr	Join-request message
mjk	Join-key message
mji	Join-inform message
mci	Group-creation-invite message
mcd	Group creation decision message i.e. Grp-creation-accept or Grp-creation-refuse message
mku	Key-update message
mru	Ring-update message
mlr	Leave-request message
e_{tx}	The energy dissipated for the transmission of 1 bit
e_{rx}	The energy dissipated for the reception of 1 bit
T_{tx}	Time needed for the transmission of 1 bit
hop	Average number of hop between two group members

to store the ECC domain parameters $T = (p, a, b, G, n, h)$ [33, 36].

In addition, a group member has to store the group key, the address of the *nextNode* and *prevNode*, as well as the GC address and public key Q for each group it belongs to. The GC also stores the members' addresses that belong to its group and the pair of public/private key (Q, d). As for the base station, it has to store the GC address corresponding to each group as well as the *black list* containing the list of compromised nodes. The following equations summarize the storage cost at each entity, for a group of n nodes:

1. Group controller stores:
   (a) The ring topology $= n * sizeof(ID)$
   (b) The Blundo polynomial share $= (t + 1) * log(q)$
   (c) The ECC domain parameters $T = (p, a, b, G, n, h)$, and the pair of public/private key (Q, d)
2. End device stores:
   (a) The next hop and previous hop (in the ring) addresses $= 2 * sizeof(ID)$
   (b) The Blundo polynomial share $= (t + 1) * log(q)$
   (c) The ECC domain parameters $T = (p, a, b, G, n, h)$, and the GC public key Q

6.2 Computation cost

The computation cost can be measured in terms of time, use of CPU or energy dissipation. In fact, these parameters are related and each one can be deduced from the other. For instance, the energy dissipation can be deduced from the time as follows: Energy = Power × Time, where Power represents the CPU power when it is in its active state and Time represents the computing time. In the present analysis, the term cost is used in its general form and we have not specified the unit (which can be second, Joule or number of CPU cycles).

The computation cost of the RiSeG scheme during each phase can be computed as the sum of the computation cost of the main operations executed during this phase. The main operations required in the RiSeG scheme are presented in Table 1 and they are namely: the encryption/decryption operation, the signature generation/verification operation, the generation of a key, and the MAC operation.

The number of required operations regarding each group membership process are as follows. In the group creation process, the joining node computes the pairwise key shared with the BS, a MAC to send the *join-request* message, a MAC to verify the received *grp-creation-invite* message, and a MAC to send a grp-creation-accept message. In total, the joining node consumes $C_{\mathrm{kc}} + 3C_{\mathrm{mac}}$.

In the group join process, the joined node computes $2C_{\mathrm{kc}} + 3C_{\mathrm{mac}} + C_{dec}$, as it computes two pairwise keys ($K_{BS,Ni}$ and $K_{GC,Ni}$), three MAC, and one decryption operation for the decryption of the group key. The group controller computes four MACs, three key computations, one key generation to generate the group key and one encryption. In total, the group controller computes $4C_{\mathrm{mac}} + 3C_{\mathrm{kc}} + C_{kg} + C_{enc}$.

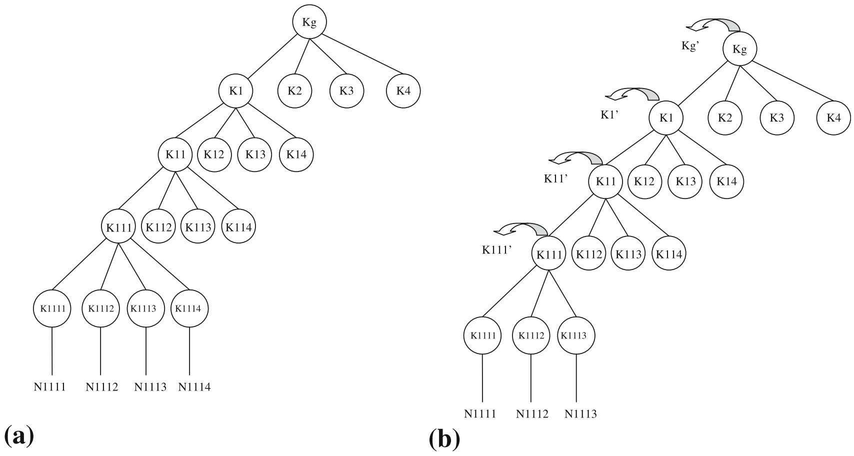
As for the group leave process, the leaving node computes $C_{\mathrm{kc}} + C_{\mathrm{mac}}$ and the group controller $3C_{\mathrm{kc}} + 3C_{\mathrm{mac}}$.

For the key-update process, in case of a join, the GC performs $C_{enc} + C_{sign}$ and other group members $C_{verif} + C_{dec}$, and in case of a leave, the GC performs $C_{kg} + 2C_{\mathrm{kc}} + 2C_{enc} + C_{sign}$ and other members perform $2C_{\mathrm{kc}} + C_{enc} + C_{dec} + C_{verif}$.

So, we can conclude that the RiSeG scheme is lightweight in terms of computation cost.

Table 3 Performance evaluation of RiSeG

	Communication cost	Computation cost
Group creation	*Ni*: $|mjr| \times e_{tx} + |mci| \times e_{rx} + |mcd| \times e_{tx}$	*Ni*: $C_{\mathrm{kc}} + 3C_{\mathrm{mac}}$
Group join	*Ni*: $|mjr| \times e_{tx} + |mjk| \times e_{rx} + |mru| \times e_{rx}$ GC: $|mji| \times e_{rx} + |mjk| \times e_{tx} + 2|mru| \times e_{tx}$	*Ni*: $2C_{\mathrm{kc}} + 3C_{\mathrm{mac}} + C_{dec}$ GC: $4C_{\mathrm{mac}} + 3C_{\mathrm{kc}} + C_{kg} + C_{enc}$
Group leave	*Ni*: $|mlr| \times e_{tx}$ GC: $|mlr| \times e_{rx} + 2|mru| \times e_{tx}$	*Ni*: $C_{\mathrm{kc}} + C_{\mathrm{mac}}$ GC: $3C_{\mathrm{kc}} + 3C_{\mathrm{mac}}$
Key update	*Case of join:*	*Case of join:*
	GC: $|mku| \times e_{tx}$	GC: $C_{enc} + C_{sign}$
	Ni: $|mku| \times e_{rx}$	*Ni*: $C_{verif} + C_{dec}$
	Case of leave:	*Case of leave:*
	GC:$2|mku| \times e_{tx}$	GC: $C_{kg} + 2C_{\mathrm{kc}} + 2C_{enc} + C_{sign}$
	Ni: $|mku| \times e_{rx} + |mku| \times e_{tx}$	*Ni*: $2C_{\mathrm{kc}} + C_{enc} + C_{dec} + C_{verif}$



(a)

(b)

Fig. 5 LKH tree. **a** Partial view of LKH tree. **b** LKH tree update after the leaving of node *N*1114

6.3 Communication cost

The main factor of the communication cost is the energy dissipation. The communication cost is computed using the same approach as TKH [18]. Actually, the communication cost in terms of energy dissipation is computed as the size of sent/received messages multiplied by the energy dissipated for the sent/receive of one bit. The different messages used in the RiSeG scheme are presented in Table 2.

In the group creation process, the group controller consumes $|mjr| \times e_{tx} + |mci| \times e_{rx} + |mcd| \times e_{tx}$ as it sends a *join-request* message, receives a *grp-creation-invite* message and finally sends a *grp-creation-accept* message.

In the group join process, the joining node *Ni* consumes $|mjr| \times e_{tx} + |mjk| \times e_{rx} + |mru| \times e_{rx}$ while the GC consumes $|mji| \times e_{rx} + |mjk| \times e_{tx} + 2|mru| \times e_{tx}$.

As for the leaving process, the leaving node Ni consumes $|mlr| \times e_{tx}$ if the node sends a leave-request message or 0 in the case of a silent leaving, while the GC consumes $|mlr| \times e_{rx} + 2|mru| \times e_{tx}$.

Regarding the key-update process in a join case the GC consumes $|mku| \times e_{tx}$ while the joining node *Ni* consumes $|mku| \times e_{rx}$, and in a leave case, the GC consumes $2|mku| \times e_{tx}$ and the group members consume $|mku| \times e_{rx} + |mku| \times e_{tx}$.

Table 3 summarizes the computation and communication costs of the RiSeG scheme.

6.4 Comparison with LKH

In order to highlight the RiSeG advantages in the WSN context, a comparison with the LKH scheme appears worth establishing. The choice of the LKH scheme is justified as follows. The LKH is a well-known scheme and several schemes such as LKHW [13], S2RP [14], LARK [22], TKH [18], etc. derive from it. So, we made the comparison with the basic scheme. Moreover, in other systems [8–11],

Table 4 Performance comparison between RiSeG and LKH schemes

Scheme	Storage cost	Key update computation cost	Key update communication cost	Key-update latency
LKH	• GC: $n \times \|ID\| + \frac{(d^{(h+1)}-1)}{(d-1)}$ keys	• GC: $h \times C_{kg} + d \times h \times C_{enc}$	• GC: $[(h-1) \times d + (d-1)] \times \|mku\| \times e_{tx} \approx h \times d \times \|mku\| \times e_{tx} = d \times log_d(n) \times \|mku\| \times e_{tx}$	• Without multicast support: Latency = $[(d-1)+d^2+\cdots+d^h-1] \times\|mku\| \times T_{tx} + hop \times \|mku\| \times T_{tx} = [(d^{(h+1)}-1)/(d-1)-3] \times \|mku\| \times T_{tx} + hop \times\|mku\| \times T_{tx} \approx (d/(d-1)) \times n \times \|mku\| \times T_{tx}$
	• ED: h keys	• ED: $\frac{(h+1)}{2} \times C_{dec}$	• ED: $\frac{(h+1)}{2} \times \|mku\| \times e_{rx}$	• With multicast support: Latency = $[(d-1)+d+d+\cdots+d-1] \times \|mku\| \times T_{tx} + hop \times \|mku\| \times T_{tx} \approx h \times d \times \|mku\| \times T_{tx}$
RiSeG	• GC: $n \times \|ID\| + (t+1) \times log(q) + T + (d, Q)$	• GC: $C_{kg} + 2C_{\mathrm{kc}} + 2C_{enc} + C_{sign}$	• GC: $2\|mku\| \times e_{tx}$	• Initiated with 2 messages: Latency = $\frac{n}{2} \times hop \times \|mku\| \times T_{tx}$
	• ED: $2\|ID\| + (t+1) \times log(q) + T + (d, Q)$	• ED: $2C_{\mathrm{kc}} + C_{enc} + C_{dec} + C_{verif}$	• ED: $\|mku\| \times e_{rx} + \|mku\| \times e_{tx}$	• Initiated with 3 messages: Latency = $\frac{n}{4} \times hop \times \|mku\| \times T_{tx}$

grouping is instead a network topology management tool, e.g., nodes are grouped according to their physical/network proximity. However, RiSeG consider an application-defined grouping. This means that sensor grouping is defined according application needs, e.g., nodes belonging to the same type or concurring to the same task or service. It follows that in RiSeG nodes in the same group may be not neighboring from a network point of view. In contrast, in [8–11] neighboring nodes belong to the same group (also called cluster) even though this topology has no meaning from the application point of view. For these reasons, we believe that RiSeG is not comparable to [8–11].

As the LKH scheme exclusively presents the rekeying process, the comparison is made with regards to the following parameters: the storage cost, communication cost, computation cost, and latency of the key-update process. However, a brief overview of the LKH scheme seems plausible to start with in the first place. Actually, the idea of the LKH [12] scheme is to construct a logical key hierarchy tree maintained by the group controller. The LKH tree is composed of key encryption keys (*KEKs*) shared between the group controller and sub-groups of the network, pairwise keys (called Individual Key *IK*) shared between the group controller and each group member, and group key (called Encryption Key *EK*) shared between all nodes in the network. The *KEKs* role is to deliver the group key, in a secure manner (using encryption), to these sub-groups. Consequently, the LKH scheme replaces several unicast rekeying messages by a single multicast message, which permits to reduce the number of rekeying messages from $O(n)$ to $O(log(n))$. However, the LKH introduces additional computation and storage costs, especially at the GC level. Hence, the LKH appears to be inappropriate for a homogeneous WSN where the GC is a sensor node with constrained resources.

We consider in Fig. 5, a logical key hierarchy tree with height $h = 4$ and degree $d = 4$ (number of nodes is $d^h = 4^4 = 512$). In LKH, the GC has to store $\frac{(d^{(h+1)}-1)}{(d-1)}$ keys $\approx (\frac{d}{d-1}) \times n$ keys, in addition to the node identity ($n.|ID|$). Group members have to store the keys on the path to the root, so h keys. In case of a leaving, the group controller must renew all the keys on the path of the leaving node to the root and then deliver them to the appropriate nodes. So, the GC needs to change h keys, including the group key. To deliver these h keys, the GC needs to send $(h-1) \times d + (d-1)$ messages, where $(h-1) \times d$ messages are sent by multicast and $(d-1)$ messages are sent by unicast. For group members, the number of rekeying messages received depends on its position on the LKH tree. For example, according to Fig. 5, after the leave of node $N1114$, node $N1111$ will receive h rekeying messages and node $N4111$ will receive a single rekeying message (only

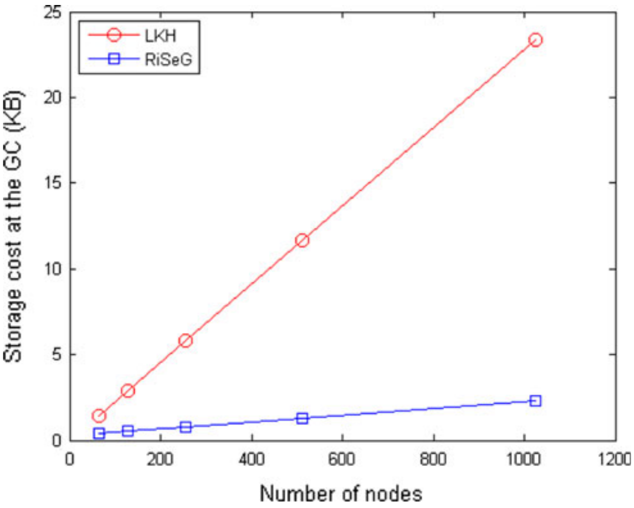


Fig. 6 Storage cost comparison

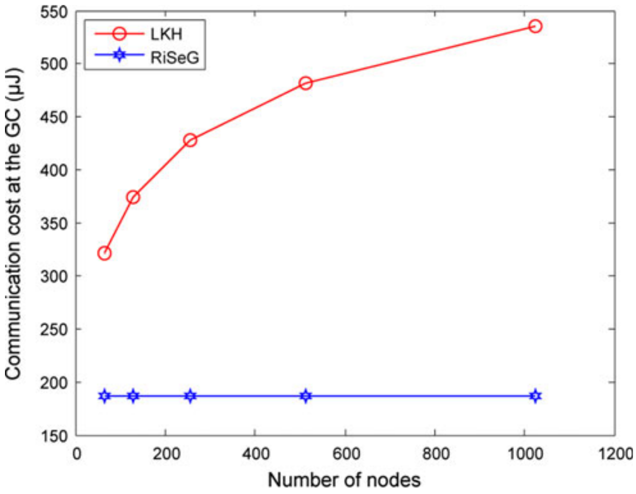


Fig. 7 Communication cost comparison

Kg' is updated). So, the average of the communication cost of a group member is $\frac{(h+1)}{2} \times |mku| \times e_{rx}$. There is also an additional communication cost related to the forwarding of the rekeying messages destined to other nodes. This cost depends on the position of the group member in the physical topology of the network.

Table 4 gives a performance comparison between RiSeG and LKH schemes.

Moreover, Fig. 6 depicts the storage cost needed for each scheme (RiSeG Vs LKH) on varying the number of group members. The key size is set to 128 bits and the identity of nodes is set to 16 bits. For the LKH scheme, the ariety of the tree is set to 4. For the RiSeG scheme, the degree of the Blundo polynomial t is set to 8 and $log(q)$ is equal to the key size (128 bits), and for the ECC parameters, we use the specification *secp160r1* defined in [36]. So that, p, a, b, G, n, d, Q are of size 160 bits. Note that, the number of keys that must be stored at the GC is $O(n)$ in the LKH scheme and $O(1)$ in the RiSeG scheme. However, as in both schemes the GC must store the identity of group members, the storage cost is linear to the number of group members.

According to Fig. 6, for n=1024, the LKH requires about 23.3 Kbytes and the RiSeG requires 2.2 Kbytes memory. If, we suppose that keys are stored on ROM memory, for TelosB motes, which have 48 Kbytes of ROM, the LKH consumes more than 50% of the available ROM, while the RiSeG consumes only 4.5% of the ROM.

Concerning the communication cost, the unit communication costs are set to $e_{tx} = 0.209$ and $e_{rx} = 0.226$ µJ from the characteristics of the CC2420 transceiver used in the Xbow's MICA-Z and TelosB sensor nodes [37]. As shown in Fig. 7, the communication cost to be consumed by the group controller during the key-update process in the RiSeG scheme is independent on the number of group members. However, the LKH communication cost at the GC is logarithmic to the number of group members and reaches 535 µJ when $n = 1{,}024$.

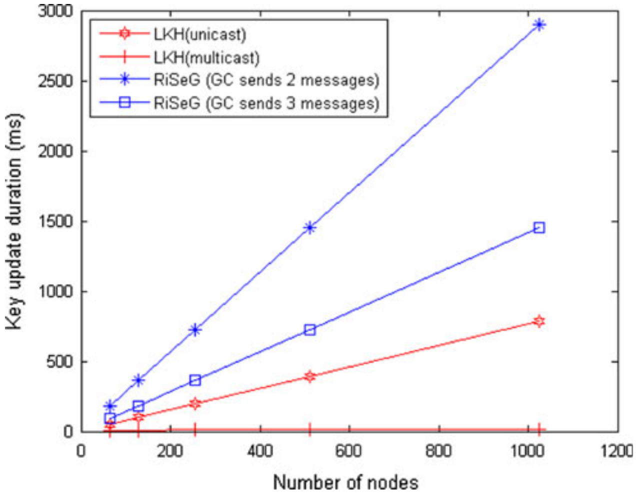


Fig. 8 Key update duration comparison

Figure 8 shows the variation of the key-update latency when varying the number of nodes. From the telosb datasheet [38], the transmit data rate is 250 kbps, so, T_{tx} is equal to 4 µs (1/250). The key-update message length is set to the size of keys (128 bits). Using unicast, LKH and RiSeG key update duration is proportional to the number of nodes in the group. However, in LKH with multicast routing support, several unicast rekeying messages are replaced by a single multicast message, and the key update duration is $O(log(n))$. Yet, the multicast routing support would add additional overhead for the construction and maintenance of the routing table.

Note that RiSeG outperforms the LKH in the following aspects:

- It requires less storage cost.
- It reduces computation and communication cost at the GC.
- It does not require multicast routing support.
- It alleviates the GC task of maintaining the group and the rekeying process.

Table 5 RiSeG messages size

Message	Size (bytes)
mjr	12
mjk	41
mji	12
mci	14
mcd	12
mku	62
mru	10

7 Implementation

In this section, a prototype of the RiSeG scheme is presented to show the feasibility of the proposed scheme and to give performance of the scheme under real WSN platform.

RiSeG has been implemented in the TinyOS [39] operating system using the nesC [40] language. For encryption, we used the AES algorithm [41] with key size of 128 bits (16 bytes). For MAC computing, we used *MMH* interface that is provided in *TinyOS-contrib/crypto* modules [42]. This interface is an implementation of the Multilinear-Modular-Hashing function [43] that provides a 32 bits MAC. For the signature, we chose to use the elliptic curve cryptography (ECC) [34], as it is adapted for resource-constrained sensor nodes (fast computation, small key size, compact signature, etc.) [44]. We have used an existing implementation of ECC [44]. However, as the implementation is done in TinyOS-1.x, we ported the code to TinyOS-2.x to operate with our code. The message exchanged have the structure presented in Fig. 9. This structure is defined in TinyOS-2.x as *message_t*. We have also defined a structure for RiSeG messages in Fig. 9. The field type (8 bits) indicates the type of the message riseg, and the field data (variable length) contains the specific RiSeG message. The different RiSeG messages are presented in Table 5 with their respective size.

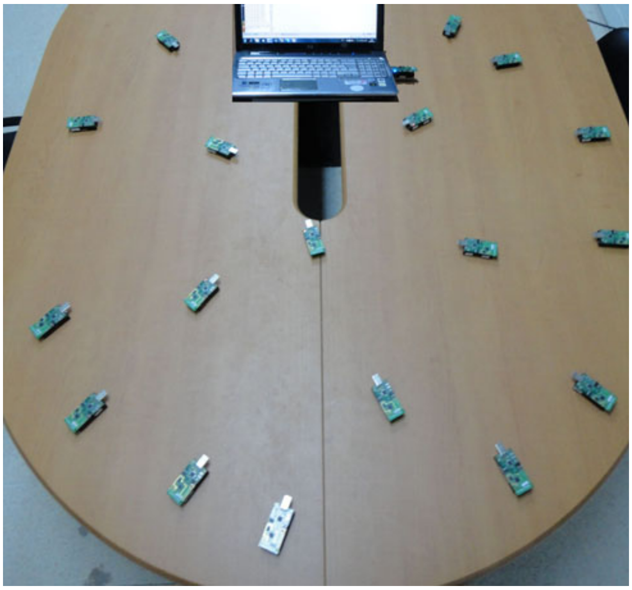


Fig. 10 Test-bed topology

RiSeG was tested on a real-world plateform using the Telosb motes [38]. Telosb mote has a 8 MHz microcontroller, 10 Kbytes of RAM memory, and 48 Kbytes of ROM memory. The testbed is formed by 20 nodes that are geographically closed to each other as shown in Fig. 10. The group controller is the node attached to the laptop in order to collect data, other nodes are end devices.

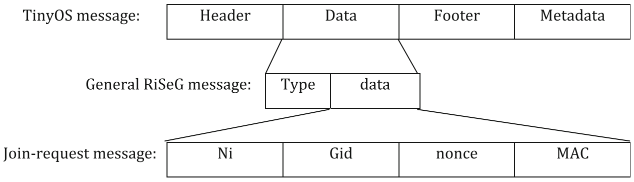


Fig. 9 TinyOS and RiSeG message structure

The following results are obtained.

7.1 Memory consumption

- For the base station, the compiled RiSeG code consumes 24,390 bytes in ROM and 5,744 bytes in RAM. These values represents respectively, 50% of ROM and 57% of RAM.
- For the end device, the compiled RiSeG code consumes 35,694 bytes in ROM and 6,448 bytes in RAM. Note that the code supports also the code of the group controller. These values represents, respectively, 72% of ROM and 64% of RAM.

Table 6 Execution time on Telosb motes

	Time (ms)
Group creation	180
Group join	700
Key update per node	~400

Table 7 Energy consumption on TelosB motes

	Time (ms)	Energy (mJ)
Encryption/decryption	230	2.76
MAC computation	0.8	9.6×10^{-3}
Blundo key computation	1	12×10^{-3}
Signature generation	3170	38.04
Signature verification	4040	48.48

7.2 Execution time

The execution time of the major RiSeG components RiSeG has also been measured and reported in Table 6. This measurement has been achieved thanks to the LocalTime<TMilli> interface provided by TinyOS. Besides the printf library has also been used to print performance parameters through the serial port of the laptop. The execution time of the group creation process has been measured as the time elapsed between the sending of the join-request message at the joining node level and the receiving of the grp-creation-accept message at the base station level. Concerning the group join execution time, it has been measured as the time elapsed between the sending of the join-request message and the receiving of the join-key message. As regards, the key-update process, the value cited in Table 6 represents the average time taken by a group member to forward a message in the ring. This time include the following operation: the reception of the message, the computation of the pairwise key shared with message sender, the decryption of the key-update message, the computation of the pairwise key shared with the next receiver, the message encryption, and finally message sending.

7.3 Energy consumption

This subsection presents the energy consumption of the main operations of the RiSeG scheme. The energy consumption is deduced by multiplying the CPU power by the computation time that is measured according to our implementation in TelosB motes. According to the TelosB datasheet [38], the CPU power consumption in its active state is 12 mW with a 3 V voltage (12 mW = 4 mA × 3 V).

The execution times of different security operations presented in Table 7 corresponds to data length of 128 bits, which represents the key size length. As already mentioned, we applied the AES algorithm for the encryption/decryption operation, and we applied the ECDSA algorithm for the signature generation and verification. The Blundo scheme has also been used for pairwise key computation. Other operations like key generation, random number generation are of the order of micro-second and, hence, energy consumption is negligible.

8 Conclusion

In this paper, RiSeG: a logical Ring based Secure Group Communication Protocol for wireless sensor networks has been proposed. A group has been considered as being a set of nodes cooperating to sense the same information. The proposed scheme is lightweight and effective thanks to the application of a logical ring topology. In addition, the scheme protects against node compromise attacks and provides both forward and backward secrecies. Moreover, the real-world implementation first proved that RiSeG is applicable to WSNs and also showed that the performance results in terms of execution time, energy consumption and memory consumption are satisfactory. RiSeG scheme may behave less well in large scale networks as it may introduce longer latencies when the number of nodes grows and when neighbor nodes in the logical ring are physically far from each other. We are planning to tackle this issue to improve the scalability of RiSeG. However, we argue that RiSeG is efficient for small to medium scales networks, as shown in Fig. 8. Besides, we intend to integrate the proposed scheme in IEEE 802.15.4/ZigBee and 6LowPAN networks, as these protocols do not support group security.

References

1. Hanson M, Powell H, Barth A, Ringgenberg K, Calhoun B, Aylor J, Lach J (2009) Body area sensor networks: challenges and opportunities. Computer 42(1):58–65
2. Zhou Y, Fang Y, Zhang Y (2008) Securing wireless sensor networks: a survey. IEEE Commun Surv Tutorials 10(3):6–28
3. Vassis D, Belsis P, Skourlas C, Pantziou G (2010) Providing advanced remote medical treatment services through pervasive environments. Personal Ubiquitous Comput 14:563–573, doi:10.1007/s00779-009-0273-0
4. Steed A, Milton R (2008) Using tracked mobile sensors to make maps of environmental effects. Personal Ubiquitous Comput 12:331–342, doi:10.1007/s00779-006-0104-5
5. Li N, Yan B, Chen G, Govindaswamy P, Wang J (2010) Design and implementation of a sensor-based wireless camera system for continuous monitoring in assistive environments. Personal Ubiquitous Comput 14:499–510, doi:10.1007/s00779-009-0271-2

6. Huang J-H, Buckingham J, Han R (2005) A level key infrastructure for secure and efficient group communication in wireless sensor network. In: Proceedings of the first international conference on security and privacy for emerging areas in communications networks. IEEE Computer Society, Washington, DC, USA, pp 249–260. Available: http://www.portal.acm.org/citation.cfm?id=1128018.1128494
7. Liu D, Ning P, Du W (2008) Group-based key predistribution for wireless sensor networks. ACM Trans Sen Netw 4(2):1–30
8. Boujelben M, Cheikhrouhou O, Abid M, Youssef H (2009) A pairing identity based key management protocol for heterogeneous wireless sensor networks. In: International conference on network and service security, N2S '09
9. Zhu S, Setia S, Jajodia S (2003) Leap: efficient security mechanisms for large-scale distributed sensor networks. In: 10th ACM conference on computer and communications security. ACM Press, New York, pp 62–72
10. Tubaishat M, Yin J, Panja B, Madria S (2004) A secure hierarchical model for sensor network. SIGMOD Rec 33(1):7–13
11. Thepvilojanapong N, Tobe Y, Sezaki K (2003) A proposal of secure group communication for wireless sensor networks. In: The 23th computer security (CSEC) group meeting, IPSJ, Tokyo, Japan, pp 47–52
12. Wong CK, Gouda M, Lam SS (2000) Secure group communications using key graphs. IEEE/ACM Trans Netw 8(1):16–30
13. Pietro RD, Mancini LV, Law SEYW, Havinga PJM (2003) Lkhw: a directed diffusion-based secure multicast scheme for wireless sensor networks. In: ICPPW '03: proceedings of the 32nd international conference on parallel processing workshops. IEEE Computer Society Press, pp 397– 406
14. Dini G, Savino I (2006) S^2RP: a secure and scalable rekeying protocol for wireless sensor network. In: Proceedings of the IEEE international conference on mobile adhoc and sensor systems (MASS 06), Vancouver, Canada, pp 457–466
15. Dini G, Giurlanda F (2010) Scalable rekeying in dynamic multigroups. In: Proceedings of the IEEE symposium on computers and communications, Riccione, Italy, pp 22–25 (to appear)
16. Cheikhrouhou O, Koubaa A, Gaddour O, Dini G, Abid M (2010) Riseg a Logical Ring based Secure Group Communication Protocol for wireless sensor networks. In: The International conference on wireless and ubiquitous systems (ICWUS 2010), Sousse, Tunisia
17. Gaddour O, Koubaa A, Abid M (2009) Segcom: a secure group communication mechanism in cluster-tree wireless sensor networks. In: First international conference on communications and networking, ComNet 2009, pp 1–7
18. Son J-H, Lee J-S, Seo S-W (2010) Topological key hierarchy for energy-efficient group key management in wireless sensor networks. Wirel Pers Commun 52(2):359–382
19. Perrig A, Canetti R, Tygar JD, Song D (2002) The tesla broadcast authentication protocol. RSA CryptoBytes 5:2–3
20. Lasassmeh S, Conrad J (2010) Time synchronization in wireless sensor networks: a survey, pp 242–245
21. Diffie W, Hellman M (1979) Privacy and authentication: an introduction to cryptography. In: Proceedings of the IEEE, pp 397–427
22. Gianluca D, Maria SI (2010) Lark: a lightweight authenticated rekeying scheme for clustered wireless sensor networks. ACM Trans Embedded Comput Syst (to appear)
23. Intanagonwiwat C, Govindan R, Estrin D (2000) Directed diffusion: a scalable and robust communication paradigm for sensor networks. In: Proceedings of the 6th annual international conference on Mobile computing and networking, MobiCom '00, pp 56–67
24. Perkins C, Royer E (1997) Ad-hoc on-demand distance vector routing. In: Proceedings of the 2nd IEEE workshop on mobile computing systems and applications, pp 90–100
25. Johnson DB, Maltz DA (1996) Dynamic source routing in ad hoc wireless networks. In: Mobile computing. Kluwer, Dordrecht, pp 153–181
26. Farooqi AH, Khan FA (2009) Intrusion detection systems for wireless sensor networks: a survey. In: Communication and networking, pp 234–241, doi:10.1007/978-3-642-10844-0-29
27. Sun B, Osborne L, Xiao Y, Guizani S (2007) Intrusion detection techniques in mobile ad hoc and wireless sensor networks. IEEE Wirel Commun (5):56–63
28. Mostarda L, Navarra A (2008) Distributed intrusion detection systems for enhancing security in mobile wireless sensor networks. Int J Distrib Sen Netw 4(2):83–109
29. Li G, He J, Fu Y (2008) Group-based intrusion detection system in wireless sensor networks. Comput Commun 31(18):4324–4332
30. Abraham J, Ramanatha KS (2008) Energy efficient key management protocols to securely confirm intrusion detection in wireless sensor networks. Wirel Sensor Actor Netw II:149–160, doi:10.1007/978-0-387-09441-0-13
31. Song H, Xie L, Zhu S, Cao G (2007) Sensor node compromise detection: the location perspective. In: IWCMC '07: proceedings of the 2007 international conference on Wireless communications and mobile computing. ACM, New York, pp 242–247
32. Blundo C, De Santis A, Vaccaro U, Herzberg A, Kutten S, Yong M (1998) Perfectly secure key distribution for dynamic conferences. Inf Comput 146(1):1–23
33. Certicom Research (2009) Standards for efficient cryptography-SEC 1: elliptic curve cryptography, http://www.secg.org/download/aid-780/sec1-v2.pdf
34. Hankerson D, Vanstone S, Menezes A (2004) Guide to elliptic curve cryptography, ser. Springer Professional Computing. Springer, Berlin
35. Liu D, Ning P, Li R (2005) Establishing pairwise keys in distributed sensor networks. ACM Trans Inf Syst Security (TISSEC), pp 41–77
36. Certicom Research (2010) Standards for efficient cryptography-SEC 2: recommended elliptic curve domain parameters, http://www.secg.org/download/aid-780/sec2-v2.pdf
37. Texas Instruments Inc. (May, 2007) Single-Chip 2.4GHz IEEE 802.15.4 Compliant and ZigBee(TM) Ready RF Transceiver. Available at: http://www-s.ti.com/sc/ds/cc2420.pdf
38. Telosb datasheet: http://www.ece.osu.edu/bibyk/ee582/telosMote.pdf
39. TinyOS (2010) http://www.tinyos.net
40. Nes C (2010) A programming language for deeply networked systems, http://www.nescc.sourceforge.net
41. National Institute of Standards and Technology (2001) NIST FIPS PUB 197 Specification for the Advanced Encryption standard (AES). National Institute of Standards and technology
42. tinyos-2.x-contrib: http://www.tinyos.cvs.sourceforge.net/viewvc/tinyos/tinyos-2.x-contrib/contrib.html
43. Halevi S., Krawczyk H (1997) Mmh: software message authentication in the gbit/second rates. In: FSE '97: proceedings of the 4th international workshop on fast software encryption. Springer, London, pp 172–189
44. Liu A, Ning P (2008) Tinyecc: a configurable library for elliptic curve cryptography in wireless sensor networks. In: Proceedings of the 7th international conference on Information processing in sensor networks, ser. IPSN '08. IEEE Computer Society, Washington, DC, pp 245–256 [Online]. Available: http://www.dx.doi.org/10.1109/IPSN.2008.47