Computer Networks 55 (2011) 759–778

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

A lightweight group-key management protocol for secure ad-hoc-network routing

Natalia Castro Fernandes *, Otto Carlos Muniz Bandeira Duarte

Grupo de Teleinformática e Automação (GTA), Universidade Federal do Rio de Janeiro (UFRJ), C.P. 68504-21945-970, Rio de Janeiro, RJ, Brazil

ARTICLE INFO

Article history:
Received 15 October 2009
Received in revised form 12 August 2010
Accepted 3 October 2010
Available online 27 October 2010
Responsible Editor: N. Agoulmine

Keywords:
Ad hoc networks
Security
Group key
Routing

ABSTRACT

Secure routing protocols for ad hoc networks use group keys for authenticating control messages without high energy consumption. A distributed and robust group-key management is, thus, essential. This paper proposes and specifies a protocol for distributing and managing group keys in ad hoc environments based on the Secure Optimized Link State Routing protocol (SOLSR). The proposed protocol manages group keys taking into consideration frequent network partitions/mergers and also reduces the impact of non-authorized users that try to illegitimately obtain the group key to use network resources. The analysis shows that our proposal provides high availability and presents low energy consumption for the two most important group events in ad hoc network: joining-node events and network-partition-merging events. Our protocol reduces both the number of control messages and the energy spent with cryptographic operations by up to three orders of magnitude when compared to contributory group-key agreement algorithms. The proposed protocol provides an efficient key management in a timely manner.

1. Introduction

Ad hoc networks are composed of self-organized wireless devices that cooperate to control the network and forward each other messages. The provision of secure routing to these networks faces specific vulnerabilities due to the absence of fixed infrastructure and the non-reliable users that want to use the network, but do not want to spend energy forwarding messages of other nodes. In addition, ad hoc networks are based on collaborative routing, which means that a node working in a malicious way may disrupt the whole network.

Protocols were proposed to provide routing and data forwarding security by restricting the group of nodes that can access the network. Secure Optimized Link State Routing protocol (SOLSR) [14], for instance, provides routing security through the use of a group key to identify the group membership. Routing control messages signed with a different group key are discarded. This kind of routing protocol demands a key management system that guarantees the property of forward secrecy, which means that the current group key cannot be used by any node to generate any future key. Thus, a node excluded from the network cannot obtain a future new group key based on its current group key. In contrast to forward secrecy, backward secrecy property is not needed in routing, because the group key only protects the control messages and a new node does not obtain any advantages from generating old group keys based on the current key. Most of the secure routing protocols rely on key management systems, but these systems are not specified by these protocols, being an open problem in ad hoc technology.

Key management is a challenge in ad hoc networks because it is not possible to guarantee the availability of a resource, such as a central authentication server, to all nodes at any time. Indeed, ad hoc networks are often partitioned and merged due to node mobility and link outages [9]. Besides, nodes frequently join and leave the network, which makes ad hoc network membership highly dynamic. Furthermore, ad hoc networks are usually composed of

* Corresponding author. Tel.: +55 21 2562 8635.
E-mail address: nataliacf@gmail.com (N.C. Fernandes).

1389-1286 © 2010 Elsevier B.V. Open access under the Elsevier OA license.
doi:10.1016/j.comnet.2010.10.003

energy-constrained devices and security must take into account energy consumption, avoiding frequent execution of complex cryptographic operations. Hence, group keys are usually preferred for ad hoc networks because symmetric cryptography is less energy consuming than asymmetric cryptography [6].

In this paper we propose and specify a protocol to manage the group key in ad hoc environments that use the Secure Optimized Link State Routing protocol. The proposed protocol, called efficient group-key management for secure routing (EGSR) [11], uses a few messages in the group-key distribution process to reduce energy consumption, especially in node-joining and partition-merging events. EGSR comprises three main procedures: group-key distribution, which updates the group key; the group-key gathering, which is used by nodes to join the network, merge network partitions, and initialize the network; and the round-leader management, which monitors and replaces the nodes responsible for a group-key distribution. In EGSR, the group key is periodically replaced to exclude non-authorized nodes which have the current group key but not a private key of an authorized node and to avoid the use of the same group key in more than some amount of data, especially when weak encryptions techniques are in use. Our proposal is compatible with ad hoc characteristics, such as the absence of infrastructure, the highly dynamic group membership, and the frequent network partitions. EGSR initialization phase avoids the use of special secrets that allows a node to initialize or join the network, but that can disrupt the entire network if exposed. Instead, in our protocol, all nodes only need a pair of public/private keys and a certificate given by the distributed certificate authority to start using the network.

The performed evaluation shows that EGSR reduces energy requirements for secure ad hoc networks based on group keys. Indeed, the analysis shows that the proposed protocol increases the energy efficiency with cryptographic operations in up to three orders of magnitude when compared to other proposals. Our protocol also presents a significant decrease in the control overhead that leads to high energy efficiency in the joining-node events and in the partition-merging events.

Aside from that, we analyzed EGSR with Petri nets to evaluate protocol characteristics. In addition, we analyzed the protocol robustness to the disclosure of the group keys and the converge delay of EGSR procedures. The analysis shows that EGSR achieves low complexity and low communication overhead, distributing the group key in a timely manner.

The remainder of the paper is structured as follows. In Section 2, we discuss related work, while in Section 3, we describe the system model and explain SOLSR model and its requirements. In Section 4, we show the details of the proposed protocol and, in Section 5, we show the analytical results. In Section 6, we present the conclusions.

2. Related work

Secure routing protocols for ad hoc networks require key management, because these protocols are typically based on cryptographic schemes to protect routing information. Usually, a distributed certificate authority [12,43,27,17] is adopted to achieve authentication and non-repudiation and, after that, authenticated nodes establish a secret group key to secure the communication. Hence, public key schemes are used to restrict and authenticate the group of users that can access the network, while secret key schemes are used to sign messages without great energy consumption. Standard approaches for group key management are based on centralized procedures, which are not well suited for ad hoc networks due to the low connectivity and the absence of infrastructure [31].

A distributed cryptographic scheme to establish a group key is the contributory key agreement [2,38,39,32], in which all nodes cooperate to form a new group key. Nevertheless, these protocols overcharge network with control messages to generate a new group key. Usually, contributory key agreement proposals are based on the Diffie–Hellman algorithm [10], which is a public key distribution mechanism. The contributory key agreement extends this algorithm to allow a group to share a key. Burmester–Desmedt (BD) [4] is a group-key agreement algorithm based on Diffie–Hellman, which generates the group key on rounds. The BD algorithm lists the nodes in a ring structure, which means that after node n, we find node 1. In the first round, each node selects a private secret, generates a public value according to this secret, and floods the network with this public value. In the second round, each node i uses its selected secret and the public value sent by nodes $i+1$ and $i-1$ to generate a new public value, which is also flooded in the network. Finally, in the last round, each node uses its secret and all the received values to generate the new group key. This is done through one exponentiation using the selected secret and n exponentiations with small exponents, where n is the number of nodes in the network. Hence, the group key is chosen based on the contribution of the whole network, which makes this algorithm robust against the choice of a weak group key.

Another approach similar to BD is the Group Diffie-Hellman (GDH-3) [37]. This proposal focuses on networks in which some nodes are energy-constrained devices. Thus, assuming a network with n nodes, $n-1$ nodes execute a few exponentiations to obtain the key, while one node executes n exponentiations. In the first round, each node selects a private value. Then, node 1 sends the public value calculated based on the selected private value to node 2, which further calculates a new public value based on its selected private value and on the public value received from node 1. Node 2 sends this new public value to node 3, which repeat this procedure until the interactive mechanism reaches node $n-1$. In the second round, node $n-1$ uses the value received from node $n-2$ to calculate an exponentiation using its selected secret value as exponent. Then, node $n-1$ floods the network with the resulting value. In the third round, each node i factors out the value received from node $n-1$ with its own secret and sends the result to node n. Node n, which has a greater computational power than the other nodes, calculates and sends a special value to each node. Based on its selected secret and on the value received from node n, each node can calculate the group key.

In addition to the GDH.3 protocol, Steiner et al. propose an auxiliary mechanism, called Auxiliary Key Agreement (AKA), to adapt GDH.3 and some other protocols to events such as a joining node, a partition merging, etc. [38]. Indeed, a protocol suite including AKA, called CLIQUES, is proposed to support dynamic group operations in Diffie–Hellman-based group-key-agreement algorithms. AKA reduces both message and computation overhead to all group operations after the initial development of the group key. The main idea of AKA is that a controller node stores all the partial values, L_i, and manipulates them whenever there is a group operation, assuming that each node knows all the other nodes in the network. No mechanism to elect and replace this controller, however, is presented in CLIQUES, as well as no mechanism to control network membership is provided. Moreover, there is no mechanism to inform new nodes or new groups being merged to the current controller. If the controller node leaves the network without being replaced, the mechanisms proposed in AKA cannot be used and the initial group-key agreement must be re-executed.

Kim et al. propose the Tree-Based Group Diffie–Hellman (TGDH) protocol to reduce message and processing overheads in the contributory group-key agreement [16]. Each member of the network represents a leaf node in a binary tree and has a set of keys arranged from the leaf node up to the root node in this binary tree. As a consequence, all nodes share the root-node secret key, which is the group key. To accomplish membership changes, TGDH selects a special member, called sponsor, to update the keys whenever a new member joins or leaves the group. Also, this protocol balances the binary tree in network-partition-splitting, network-partition-merging, and node-joining/leaving events. According to Gangwar and Sarje [13], TGDH has moderate costs when the tree is fully balanced. Hence, the events of node join, partition split, and partition merging are costly, because they can imbalance the binary tree.

The use of trees on contributory key agreement with elliptic curve cryptography (ECC) is also being adopted to reduce energy costs. Kumar et al. propose a region-based group-key agreement protocol based on ECC, which uses the Group Elliptic Curve Diffie–Hellman protocol (GECDH) and the Tree-based Group Elliptic Curve Diffie–Hellman protocol (TGECDH). In this proposal, the group of nodes is broken into region-based subgroups, each one with a different leader. Each region-based subgroup has its own group key, and the leaders communicate using an outer group key. Thus, if there is a membership change on a specific group, only the key of that group is updated. Meanwhile, if a leader leaves the network, then both the group key of the leader group and the outer group key are updated [20]. Although this proposal is based on regions, there are no comments about how to divide the network into regions or how to elect the leaders. Moreover, the protocol assumes that data such as the number of nodes or the moment a member leaves the network is known, but no mechanisms to disseminate these data are provided. Other disadvantage, according to the authors, is that it takes critical time to generate a new contributory group key. Another proposal based on ECC is proposed by Li-Ping et al., in which a contributory group-key agreement protocol is developed over a circular hierarchical group model [25]. In this proposal, referred to as CH-ECC, the network is divided into h layers composed of subgroups of size c. Each subgroup has a different group key, while, based on these keys, the whole network is able to generate a group key. The CH-ECC, however, does not detail how to form the circular hierarchical groups in a decentralized way.

A challenge for contributory key agreement proposals are to establish the order of the nodes to form the group key and, for some schemes, such as GDH.3 and TGDH, also to specify which node is the 'n' node or the sponsor node. In addition, these mechanisms assume that all nodes know the routes to each other, which is a strong assumption if we take into consideration the case of a group key for secure routing in ad hoc networks. In this situation, nodes do not know any route, because they are not able to exchange control messages before establishing a group key. Hence, all contributory key agreement control messages must be flooded in the network.

Key pre-distribution schemes address the key distribution for networks composed of energy-constrained devices. In this approach, an administrator selects a pool of keys from the key space. Each node receives a random subset from the key pool before network deployment. Any pair of nodes able to find a common key within their subsets can use that key to establish a secure communication. After the stabilization of secure links, nodes select a group key [28,7]. Luo et al. propose a group-key management system based on key pre-distribution and on the contributory key agreement [28]. In this protocol, nodes must keep a list with all excluded nodes and pre-distributed keys of excluded nodes are discarded. This can imply in a connectivity problem, in case of many excluded nodes. Chan et al. propose the 'q-composite scheme', the 'multipath reinforcement scheme', and the 'random-pairwise keys scheme' to enable characteristics like node-to-node authentication and quorum-based revocation [7]. The main disadvantage of this kind of proposals for self-organized ad hoc networks is the premise of an administrator that must configure all the nodes. Also, the disclosure of some node secret keys can compromise the whole network.

Cluster-based and location-based protocols aim to build a scalable key management and to reduce the number of messages transmitted when a node join or leave the network [39,18,26,23,22]. Distributed, Efficient Clustering Approach (DECA) uses clusters to distribute keys in ad hoc networks [24]. The disadvantage of DECA and also other cluster-based proposals is their high-energy consumption for managing clusters if network membership often changes. Another approach to distribute group keys on multicast environments based on clusters is the Optimized Multicast Cluster Tree with Multipoint Relays (OMCT with MPR), whose main idea is to use information of OLSR protocol to elect the local controllers of the created clusters [3]. OMCT with MPRs assumes that routing control messages have been exchanged before the key distribution. In SOLSR, however, all routing control messages must be signed. Therefore, key distribution must be deployed before the exchange of routing control messages. Then, OMCT with MPRs is not useful to distribute a group key in the SOLSR protocol.

Our proposal for managing group key in ad hoc networks, called efficient group-key management for secure routing (EGSR), significantly reduces the control-message overhead when compared with contributory key agreement protocols or cluster-based protocols. Besides, our protocol does not depend on the establishment of secrets before the network deployment, as occurs in the key pre-distribution schemes. Therefore, even if authorized nodes are hacked, network security is not completely compromised. EGSR deals with the challenges of frequent network access by non-authorized nodes that illicitly obtained the group key and network partitions. Instead of most of group-key agreement/distribution proposals, our protocol is able to identify all dynamic group events that affect group-key-based routing in ad hoc network, such as nodes joining the network, network initialization, and network partition merging. Additionally, EGSR specifies how to authenticate nodes and how to securely distribute the group key in the dynamic group events with a small overhead, assuming the existence of a distributed certificate authority. A distributed entity, such as the distributed certificate authority, is needed by most of group key management protocols to specify the list of authorized nodes and to support node authentication. The exceptions are the key pre-distribution schemes, which replace this entity by an administrator which install secrets in all authorized nodes before network deployment. The main advantage of our protocol is the low energy consumption, due to the small number of messages required for the key distribution, especially in joining-node and partition-merging events. In addition, EGSR group key distribution works without the knowledge of the available routes. This is important because secure routing protocols only accept control messages signed with the group key, and then, the group-key distribution must occur before the calculation of the routes.

3. System model

3.1. Network model

Our protocol works under the assumption of mobile nodes which collaboratively support network operation. Network partitions can occur at any time and nodes frequently join and leave the network. We define as group the set of nodes that can communicate through routes of one or more hops.

We assume that a distributed certificate authority controls network membership [43,27], which means that this entity knows which are the nodes that can access the network. By network membership, we mean all the nodes that are authorized by a third party, such as a certificate authority, to access the network. The certificate authority creates certificates for each authorized node, associating a public key Pk_i to an identity id_i. Each node knows its public and private keys as well as its certificate a priori. We also assume that this distributed certificate authority is able to notify authorized nodes whenever there is a change on the network membership, through the emission of a revoked certificate list. The revoked certificate list contains all certificates that are still valid, but cannot be used anymore.

Authorized nodes in the same group must share the same group key to exchange routing control messages. We assume that nodes run the Secure Optimized Link State Routing protocol (SOLSR) [14], which is an extension to provide security for the Optimized Link State Routing protocol (OLSR) [8].

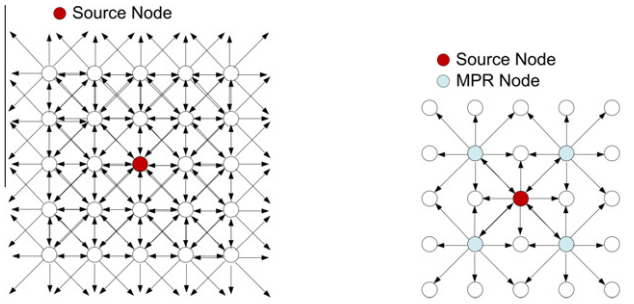
3.2. OLSR and SOLSR

OLSR is a pro-active and link-state-based routing protocol and, thus, its routing table is constructed based on information generated by node neighbors and also on all possible destinations in the network. To mitigate the overload effect caused by the control-message flooding events, OLSR limits the flooding procedure using the multipoint relay (MPR) mechanism. In this mechanism, every node selects its set of multipoint relay (MPR) nodes among the one-hop neighbors. The basic rule for the MPR selection procedure is that each node must reach all two-hop neighbors through its MPR set. Because selecting the best MPR set, which means to find the smallest set of one-hop neighbors that reaches all the two-hop neighbors, is a hard problem, this selection is done through heuristics. The RFC of OLSR [8] suggests an algorithm based on first selecting one-hop neighbors that are the only one to reach a specific two-hop neighbor. After that, the algorithm calculates the number of two-hop neighbors that each of the remaining one-hop neighbors reaches, excluding the two-hop neighbors that were already reached by the first selected MPR nodes. Then, the one-hop neighbor that reaches more two-hop neighbors is selected as MPR. Next, all the two-hop neighbors that this selected one-hop neighbor reaches are excluded of the two-hop neighbor set and the algorithm is run again until the two-hop neighbor set is empty.

The MPR nodes are responsible for forwarding routing messages in flooding events. Therefore, when a node sends a routing message which must be flooded, only its MPR will forward the message, reducing control message overload. This procedure is repeated by the MPR nodes of each MPR in a flooding, which guarantees that a message will reach the whole network. It is important noticing that the MPR nodes reduce the overhead in a flooding, but they usually do not eliminate all the redundancies in a flood event. An example of a flood event using MPR nodes is on Fig. 1.

Because each node in OLSR monitors its links with neighbors, every time a link failure or a new link is detected, the node floods the network with the current link state. Hence, all the nodes in the network can update their routing table. This way, OLSR correctly handles mobility in ad hoc networks.

SOLSR secures OLSR through two mechanisms: access control and message replay protection [14,40,1]. SOLSR assumes that all nodes authorized to access the network share a group key a priori. Then, all SOLSR messages are signed with this group key to perform the access control. As a consequence, non-authorized nodes cannot create or modify control messages in the network. SOLSR, however, does not specify how the group key is managed or distributed.



(a) Flood without MPR: all nodes retransmit the message.

(b) Flood with MPR: only MPR nodes retransmit the message.

Fig. 1. Example of the use of MPR nodes in a flood event. The number of nodes retransmitting the message and number of message copies each node receives are reduced.

The message replay protection of SOLSR prevents malicious nodes from replaying old control messages to damage routing. Thus, each node controls the difference between its own clock and the clocks of other communicating nodes. Therefore, the first time two nodes, A and B, need to communicate, they exchange their timestamps, T_{A_j} and T_{B_j}, to discover the difference between their clocks, $T_{A,B} = T_{A_j} - T_{B_j}$. When node A sends a message to node B, it sends the timestamp of the moment the message was sent, T_S. After node B receives the message, it checks Condition (1), given by

$$T_S - T_{B_i} - S \leqslant T_{A,B} \leqslant T_S - T_{B_i} + S, \tag{1}$$

where S is the maximum delay tolerance in the transmission and T_{B_i} is the current timestamp of node B. Since node A and node B are neighbors, S is chosen as a small value. If this condition does not hold, then node B classifies the message as a replay and discards the message. This mechanism prevents, for instance, malicious nodes from replaying old Hello messages from an authorized node to forge the presence of this authorized node elsewhere in the network. Because authorized nodes always check Condition (1) after receiving a control message and non-authorized nodes which does not have the current group key cannot modify the fields inside the signed control message, a replay of old Hello messages can always be detected by any authorized node in the network. As a consequence of the message replay protection, nodes know the difference between their clocks and the clock of their neighbors in SOLSR, which means that the network has a weak synchronization.

Our protocol, EGSR, aims to solve the group key distribution and management in a complete distributed fashion. Accordingly, only nodes which can obtain a certificate out of the revoked certificate list are able to successfully run EGSR, obtain the current group key, send routing messages, and access the secure network.

3.3. Adversary model

We consider as adversary any non-authorized node or any authorized malicious node. Adversaries may behave in a malicious way, damaging network by creating, modifying or discarding messages. Also, non-authorized nodes that obtained the group key may behave properly, but consuming network resources, such as bandwidth. For these reasons, EGSR always tries to exclude non-authorized nodes from the network. Excluding a node means discarding all messages going to or coming from this node and preventing this node from receiving a group key.

Malicious authorized nodes are hard to detect and exclude in the network layer when using symmetric keys to sign routing control messages. Hence, routing attacks can be detected, but a malicious authorized node cannot be accused based only on the observation of routing control messages. The exclusion of this kind of node is not on the scope of this work.

We do not make assumptions about processing power of the adversaries. We assume that adversaries can steal group keys and can collude, but they are always minority in the network. This minority is important to guarantee that the ad hoc network has a high chance of trustful packet forwarding, independent of our protocol or of the malicious node actions.

4. The proposed scheme

The proposed protocol uses asymmetric cryptography to distribute the group key to all the nodes. The distribution is accomplished by three main procedures. The first one, the group-key distribution, is responsible for establishing a new group key in cases of node exclusion and periodical group-key replacement. It is worth mentioning that routing, unlike other applications, does not require group-key replacement when a node joins the network because confidentiality is not a goal. The second procedure deals with the challenges of node-joining, partition-merging, and network-initialization events. The third procedure treats the leader failure detection and the leader replacement. In EGSR, the group-key distribution is initialized in each round by a round leader. If the round leader fails, it is necessary to automatically replace the round leader to continue the group-key distribution.

4.1. Group-key distribution

The group-key distribution procedure is trigged to replace the group key in three cases: periodically, whenever

a node is excluded, or when a bad behavior is detected by a bad behavior detection system (BBDS) [29,42,41]. The group-key distribution is periodically executed to exclude non-authorized nodes which illicitly obtained the current group key but do not have a private key of an authorized node. For instance, an authorized user may reveal the group key to a non-authorized friend in order to allow his friend to access network resources. The group key distribution is also executed when a node is excluded to guarantee that the excluded node does not have the current group key. Hence, the distributed certificate authority revokes the certificate of the excluded node and sends the updated revoked certificate list to all nodes. This event causes the revocation of the current group key and the distribution of a new group key. After that, the excluded node is neither able to obtain the new group key nor to generate new control messages. Finally, the group-key distribution can be triggered by the BBDS, because when the BBDS sends an alert, it means that there is an adversary that should be excluded from the network. This adversary, however, cannot be identified in the routing layer due to the use of group keys, which do not authenticate users. If this adversary is not authorized, but has disclosed the group key, it will be purged after a new group-key distribution.

Fig. 2 illustrates the group-key distribution procedure. In EGSR, nodes are able to select a round leader in each group-key distribution, as we show in Section 4.5. The round leader initiates the group-key distribution through the broadcast of an Announcement message, which indicates the existence of a new group key. When the neighbors of the round leader listen to the Announcement message, they send an Order message asking the new group key. The round leader ends the process by sending to each neighbor a Response message, which contains the new group key encrypted with the public key of its neighbor. The neighbors that are multipoint relays (MPRs) of the leader further retransmit the Announcement message, and the two-hop neighbors of the leader select an MPR to obtain the new group key. The leader, its MPRs, the MPRs of MPRs of the leader etc. repeat this procedure, just as in a controlled flooding, to attain all the nodes of the network and to guarantee that all nodes will receive the new group key.

The messages used on the group-key distribution are in Fig. 3. The 'Signature with Private Key', 'Certificate', and 'New Group-key Encrypted with Neighbor Public Key' fields have variable size, depending on the hash function, cryptographic algorithms, and key size. The certificate and the message signature authenticate the sending node and guarantee the content integrity. Besides, the key distribution for each pair of nodes is only successful if both nodes prove that they have a valid certificate issued by the certificate authority and that they are not on the revoked certificate list. The 'Key Sequence Number' identifies the group key being distributed. The 'Current Round Leader', 'Next Round Leader', 'Group-Key Distribution Interval', and 'Distribution-Start Timestamp' fields are important for the round-leader selection procedure and the round-leader failure detection, as we explain later. The 'Distribution-Start Timestamp' field generated by the round leader, l_n, which we call $T_{b_i}(l_n)$, is updated hop-by-hop. Thus, based on $T_{b_j}(l_n)$ received from node j, node i estimates the time of the beginning of the group-key distribution procedure, $T_{b_i}(l_n)$, on its own clock according to

$$T_{b_i}(l_n) = T_{b_j}(l_n) - T_{j,i}, \tag{2}$$

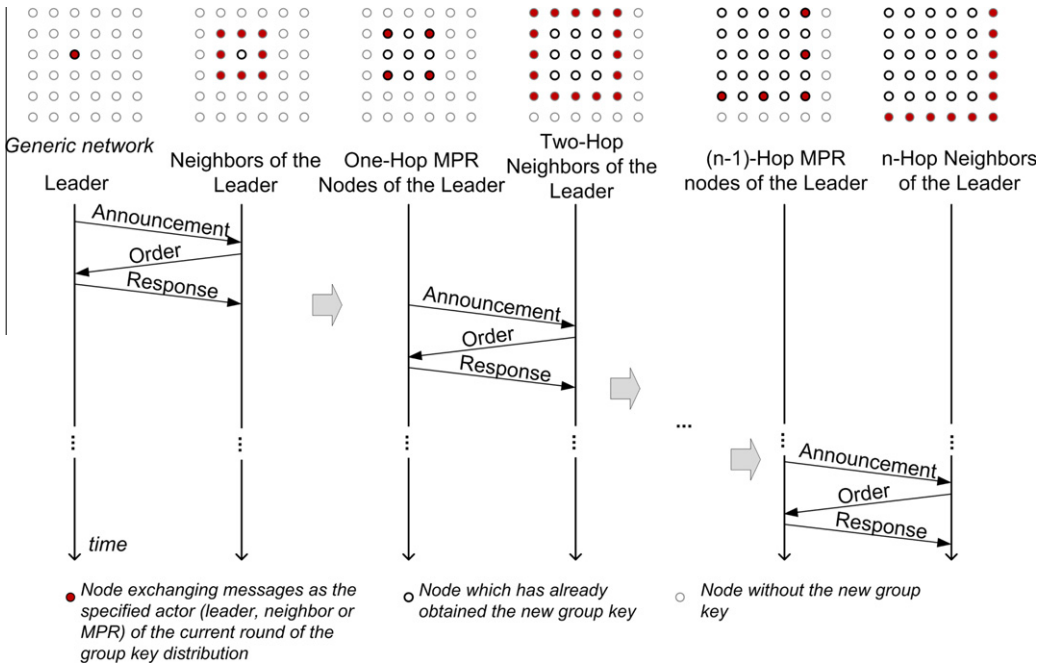


Fig. 2. Group-key-distribution procedure model and an example of the group-key distribution on a generic network.

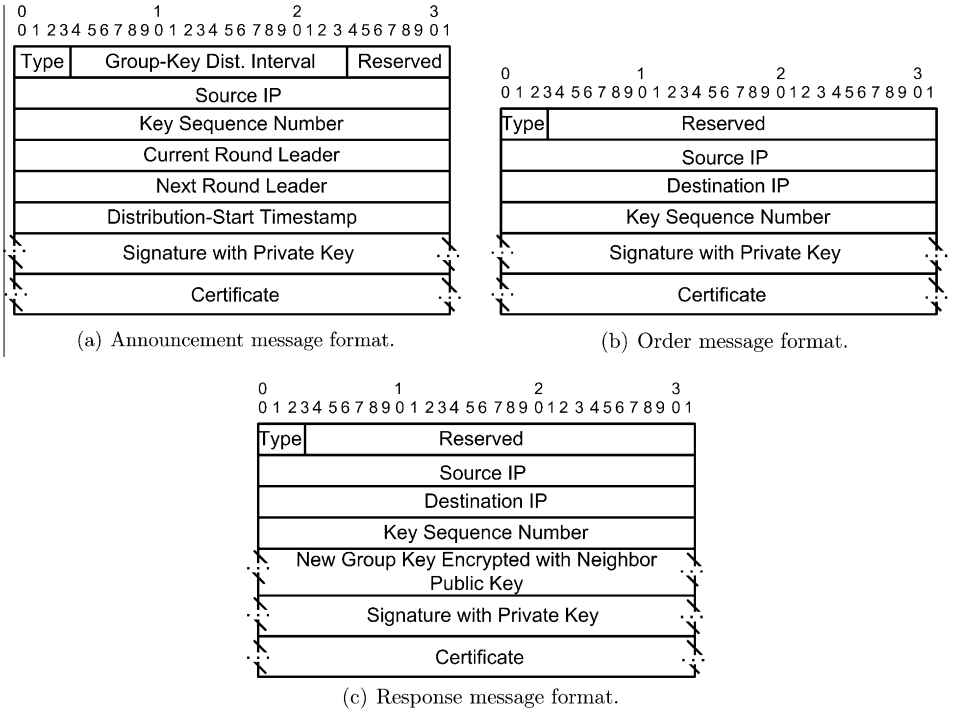


Fig. 3. Group-key distribution messages.

where $T_{j,i}$ is provided by SOLSR and indicates an estimative of the difference between the clock of node *j* and the clock of node *i*. Hence, all nodes know approximately when the current group-key distribution has begun and when the next group-key distribution should start.

Due to the group key distribution procedure, even if a non-authorized node has an old group key, it cannot obtain the new one. As the new group key is randomly chosen and is encrypted with the public key of the destination node, non-authorized nodes cannot derive the new group key from an old group key nor spoof the group-key distribution to obtain it.

4.1.1. Using the new group key

Nodes must begin to use the new group key approximately at the same time. Therefore, a node *i* calculates the expected time to start using the new group key, $T_{w_i}(l_n)$, given by

$$T_{w_i}(l_n) = T_{b_i}(l_n) + T_n * H_{max}. \quad (3)$$

In this equation, $T_{b_i}(l_n)$ is the approximate time when the group-key distribution procedure began according to node *i*, given by Eq. (2), T_n represents an upper bound of the maximum delay an MPR takes to transmit the new group key to its neighbors, and H_{max} represents the number of hops between the round leader and the farthest node, which is obtained with data collected by SOLSR.

Node *i* starts to use the new group key after $T_{w_i}(l_n)$, although it accepts messages signed with the old or the new group key in the period given by $T_{w_i}(l_n) - \alpha$ and $T_{w_i}(l_n) + \alpha$, where α represents the delay tolerance. After $T_{w_i}(l_n) + \alpha$, messages not signed with the new group key are discarded. If a node *j* does not receive the group key before $T_{w_j}(l_n) + \alpha$, it will obtain the new key based on the procedures described in Sections 4.2 and 4.5, which deal with partition-merging events and round-leader failures. Indeed, if the node does not obtain the group key, it will conclude that the current leader has failed and will choose new leaders until the node becomes the leader and chooses another key or the node will detect that its neighbors have a different group key and will start a partition merging procedure. Due to α and to these procedures EGSR only needs a weak synchronization provided by SOLSR timestamp exchange.

4.2. Gathering the current group key

The group-key distribution procedure treats excluded nodes, but not joining nodes. When an authorized node joins the network, it must obtain the current group key. Similarly, when two network partitions restore a common link, they must establish a common group key to guarantee that all the routing control messages are accepted by the nodes of both partitions. It is important to notice that if an authorized node leaves the network or if a network partition occurs, there is no need for key replacement. These nodes were not suspected of malicious actions and were not excluded, so they can own the current group key, because they are not expected to damage the network.

We propose three procedures to a node or a group of nodes obtain the current group key: the joining-node procedure, the partition-merging procedure and the network-initialization procedure. The joining-node procedure allows authorized nodes which do not hold the group key

to obtain it with any other authorized node. The partition-merging procedure joins network partitions caused by connection problems [9] or when nodes leave the network. Therefore, these two procedures to an authorized node gather the current group key make EGSR robust to nodes that frequently join and leave the network, delays in the group-key distribution procedure, and link losses. The network-initialization procedure organizes the nodes and establishes a group key when nodes are joining to create a new network.

Any authorized node that does not have the group key can obtain it with any other authorized node. In the end of the process, nodes will have a common group key to be used in secure routing.

4.2.1. Joining-node procedure

A node can join the network if it was previously authorized and has obtained a valid certificate. Fig. 4 shows the joining-node procedure when node *B* joins the network and obtains the group key with node *A*, which belongs to the group and has the current group key. After this procedure, node *B* can exchange timestamps with its neighbors and send/receive routing control messages, such as the periodic Hello messages of SOLSR. Tables 1 and 2 show notations used in this paper.

The joining-node procedure uses two messages: Join and Accept. The Join message, depicted in Fig. 5(a), signals to node *A* that node *B* is a joining node. The Join message contains the new node certificate and also a signature with node *B*'s private key, because node *B* must prove to node *A* that it is an authorized node. The Accept message, described in Fig. 5(b), informs the current parameters of node *A*'s partition: the group key, which node *A* uses to send routing control messages, the number of nodes in the partition, used in the partition-merging procedure, and the relevant data for the next group-key distribution.

Table 1
Notations used in the procedure descriptions.

Notation	Meaning/Action
$?M$	Receive message M
$?_1\ M$	First reception of message M
$!M$	Send message M
$\overline{X}$	Not X
$C_1 \wedge C_2$	C_1 and C_2
$C_1 \vee C_2$	C_1 or C_2
$[C_1]\|[C_2]$	If condition C_1 holds, then do C_2
C_A	Certificate of node A
C_B	Certificate of node B
Gk	Network group key
Gk_{Pt_1}	Group key of partition 1
Gk_{Pt_2}	Group key of partition 2
Pk_A	Public key of node A
Pk_B	Public key of node B
pk_A	Private key of node A
pk_B	Private key of node B
L_{rev}	Revoked certificate list
L_{act}	Active node list obtained with SOLSR

4.3. Partition-merging procedure

We need to identify each group of nodes in the network with different group keys to detect and merge network partitions. EGSR uses the signature of the SOLSR Hello messages as partition identifiers. A partition-merging scenario is detected if a signature of a Hello message of a neighbor was not generated with the current group key. Hence, if two neighbors have different group keys and there are more than T_H seconds since the last group-key update and T_P seconds since one of these nodes started a partition-merging procedure, then these nodes are in different partitions and should start the partition-merging procedure. The period of T_H seconds guarantees that a group-key update is not happening at that moment and the

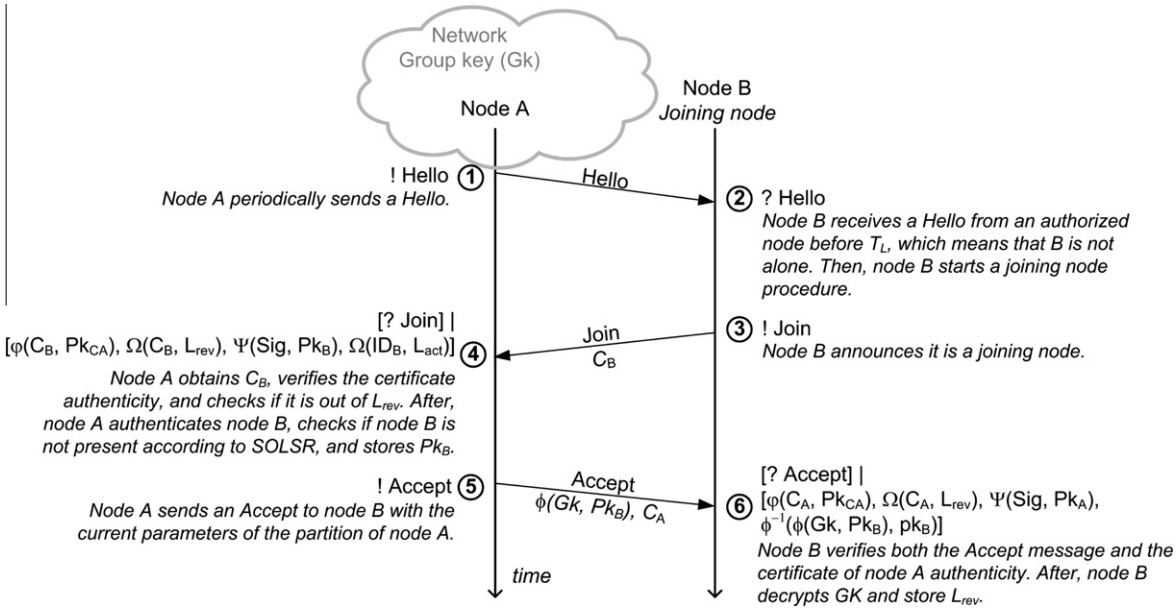


Fig. 4. Message exchange over time in the joining-node procedure, assuming that node *A* is already on the network and node *B* is the joining node.

Table 2
Notations used to cryptographic operation descriptions in the procedures.

Notation	Meaning/Action
$\phi(M,Pk)$	Encrypt M using public key Pk
$\phi^{-1}(M,pk)$	Decrypt M using private key pk
$\vartheta(M,Gk)$	Encrypt M using group key Gk
$\vartheta^{-1}(M,Gk)$	Decrypt M using group key Gk
$\varphi(C,Pk_{CA})$	Check certificate C using the public key of the certificate authority Pk_{CA}
$\Psi(Sig,Pk)$	Check signature of the message, Sig, using the public key Pk
$\theta(Sig,Gk)$	Check if signature of the message, Sig, was not generated using the group key Gk
$\Omega(E,L)$	Check if element E is not on list L

period of T_P seconds avoids malicious nodes from trying to exhaust the battery of a node with frequent false partition-merging procedures.

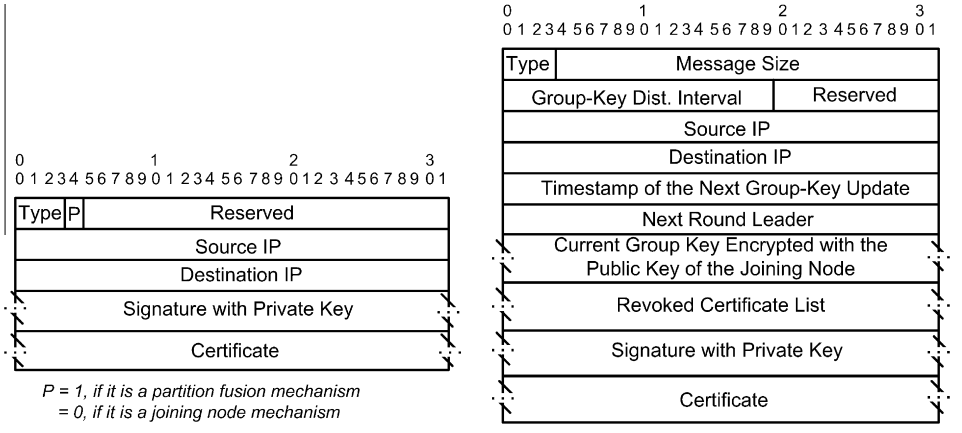
Fig. 6 describes the partition-merging procedure. First, the nodes that detected the partition exchange the messages Join, Associate and Confirmation, depicted in Fig. 5(a), 7(a), and 7(b), to obtain the data about each other and about each partition. Each node executes only one partition merging procedure at a time. Hence, if many nodes send the Join message to node A, node A will send the Associate only for the first communicating node, to avoid unnecessary message overhead. After this message exchange, both nodes know the group key and the number of nodes of the other partition. The node in the smallest partition, which is node B in our example, then announces itself as immediate round leader and distributes the group key of partition 1 through the flood of the Partition message, described in Fig. 7(c). In this message, node B warns its partition about the partition-merging procedure and advertises the new group key and the group-key distribution data of partition 1. Therefore, after all nodes in partition 2 receive this message, they can exchange routing control messages with nodes of partition 1 and, also, they are able to detect failures in the leader selection of the next group-key distribution, as we show in Section 4.5.

It is worth mentioning that EGSR does not authenticate every node in a partition merging procedure, because all nodes with the group key are trusted. Instead of the group key distribution procedure, here there is no need to authenticate all the nodes to check if all belong to the group. Then, as depicted in Fig. 6, the Partition message is signed with the group key of partition 2 instead of private key of node B and the new group key is encrypted with the group key of partition 2. Hence, the partition-merging procedure has low energy consumption.

Since many partition-merging procedures can occur at the same time, EGSR uses a decision process based on three rules to avoid loops. First, if a node detects a partition, it starts the procedure sending a Join message only if its IP is greater than the other node IP. This avoid that both nodes start the procedure at the same time. Second, if there are more than one partition merging occurring at the same time, the Partition messages of the smaller partitions are always discarded and the group key of the partition with more nodes is adopted. Third, if partitions have the same size, the partition with the leader with the greatest IP address will predominate and the Partition messages of the other partitions are discarded. Therefore, after the partition-merging procedure, all partitions are expected to share a unique group key of the greatest partition.

4.4. Network-initialization procedure

The network-initialization procedure guarantees that nodes organize themselves in the beginning of the network and, after a period of time, all nodes share a unique group key in a distributed way. The worst case scenario occurs when a group of nodes joins the new network simultaneously. A node assumes it is in the network initialization phase if, after listening the medium for a period T_L, the node does not receive any Hello message from an authorized node. Thus, this node chooses a group key and starts to send Hello messages. After that, all arriving nodes must just accomplish the joining-node procedure to receive the group key and access the network. Nevertheless, if more



(a) Join message format. (b) Accept message format.

Fig. 5. Messages used in the joining-node procedure of EGSR.

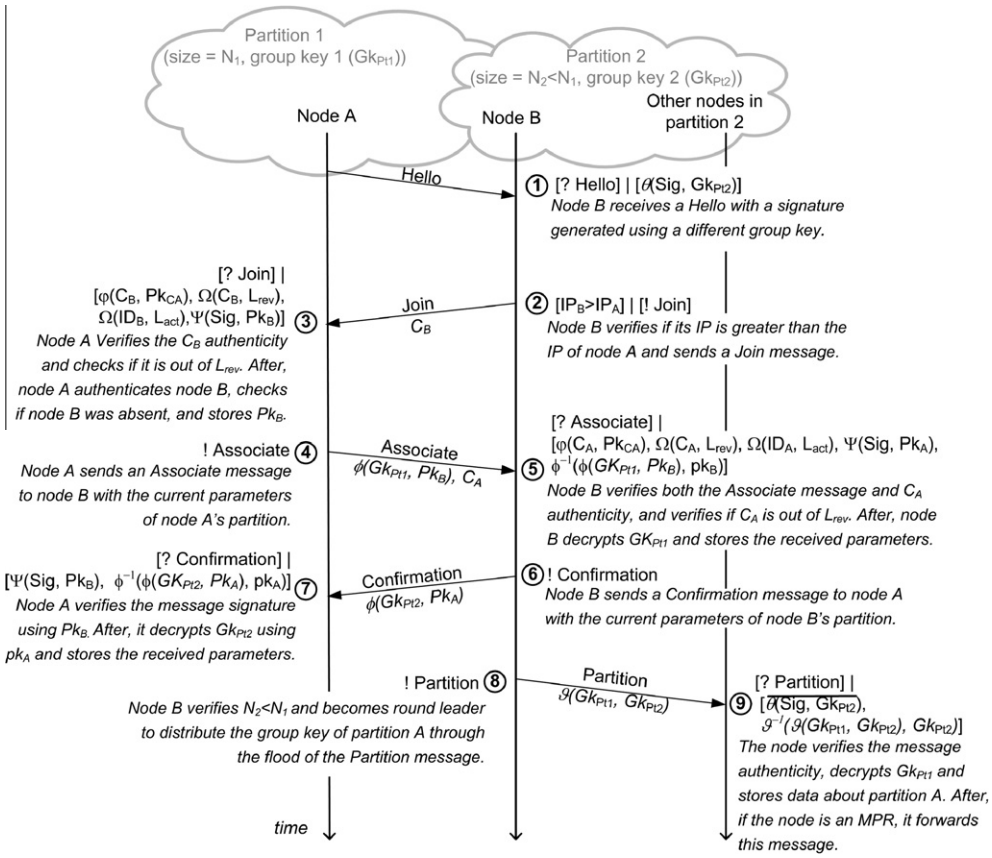


Fig. 6. Message exchange over time in the partition-merging procedure.

than one node starts the network-initialization procedure at the same time, each node will choose a different group key. These nodes realize that they are actually neighbors and have different keys when they receive the Hello messages sent by the other nodes. As a consequence, they will start partition-merging procedures. Indeed, the nodes will arrange themselves in small partitions with one hop neighbors and these small partitions will further merge, as shown in Fig. 8. The partition-merging procedure is repeated until the whole network attains a unique key.

4.5. Round-leader management

The round leader randomly chooses a group key and starts the group-key distribution procedure. Therefore, the leader plays a special role in each group-key-distribution round and is a single point of failure, which could disrupt the entire group-key management. Furthermore, if a malicious node is chosen as leader, it could choose weak group keys to damage the network security. To avoid these problems, we propose two procedures: the round-leader selection and the round-leader replacement.

The round-leader selection follows a rule to avoid that colluding malicious nodes are always the round leaders. Also, this procedure avoids extra message overhead, because the nodes reach to the same decision based only on the already stored data of SOLSR. Indeed, SOLSR proactively lists all the possible destinations in the network, which correspond to the list of all active nodes. Although SOLSR routes can often vary, this list is considered stable and can be used to select the round leaders with a really low error rate. This means that all nodes will probably choose the same node as next round leader.

A round leader selects the next round leader by creating a circular ordered list based on the active-node IP address list provided by SOLSR. The round leader selects its successor in this list as the next round leader. As a consequence, a malicious node cannot easily choose another malicious node as round leader, because all nodes can verify if the next round leader was correctly chosen. Moreover, if a malicious node is selected as round leader and it chooses a weak group key, the nodes that run a bad behavior detection system (BBDS) can detect this and warn the next round leader to start a new group-key distribution.

According to the round-leader selection, the round leader is chosen before the moment of the next group-key distribution. Therefore, the round leader might be unavailable in the next group-key distribution, compromising the group-key management. EGSR avoids this problem with the proposed self-adaptive round-leader replacement procedure, which detects a leader failure and replaces the leader.

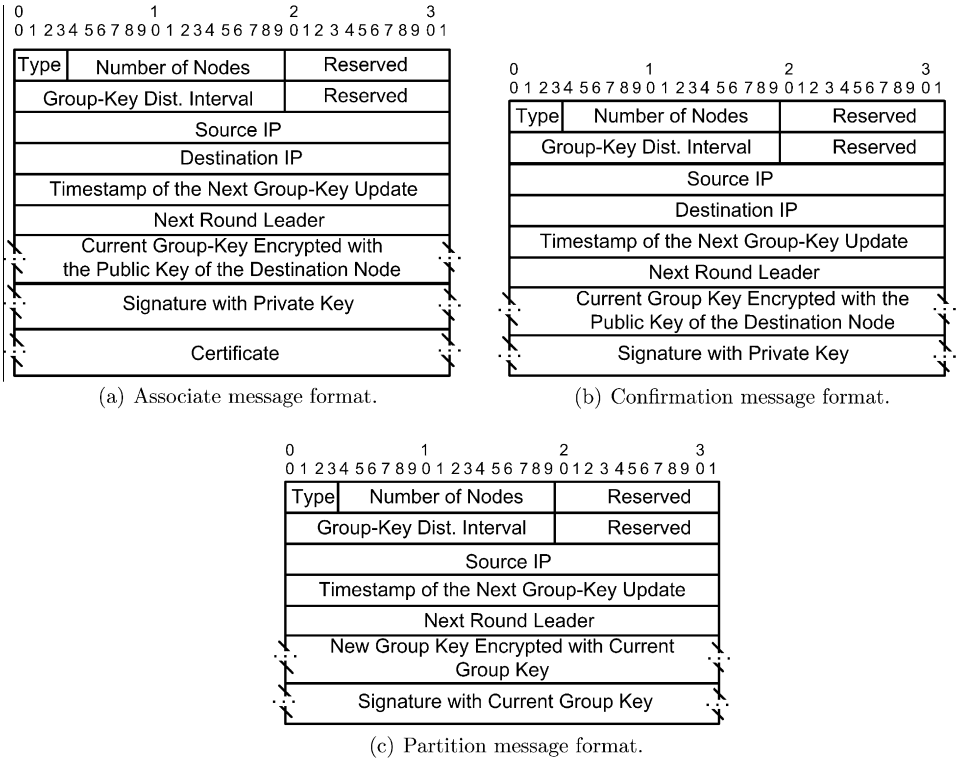


Fig. 7. Messages used in the partition-merging procedure of EGSR.

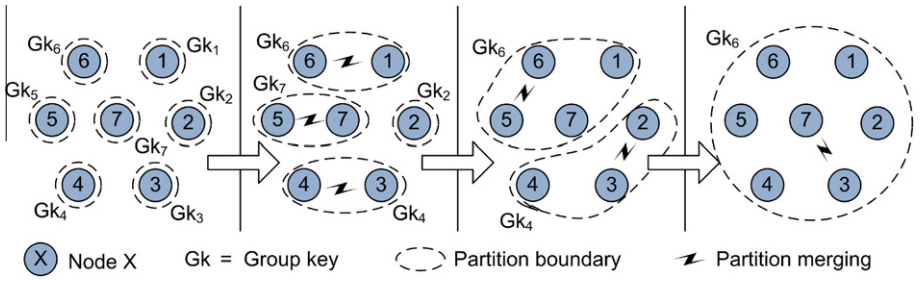


Fig. 8. Network-initialization procedure when seven nodes join the network almost simultaneously.

A node j detects that the round leader l_n failed when a group-key distribution is pretended to start, but no neighbors sent the Announcement Message after the expected time to receive the new group key, $T_{k_j}(l_n)$. Based on the hop-by-hop delay to receive the new group key, node j estimates this expected time, $T_{k_j}(l_n)$, which we define by

$$T_{k_j}(l_n) = T_s + T_n \cdot H_{l_n j} + \delta, \tag{4}$$

where δ is the delay tolerance and $H_{l_n j}$ is the number of hops from the round leader l_n up to node j. The variable T_n is an estimate of the maximum delay for group-key distribution from an MPR to its neighbors and T_s is the expected time for the start of the group-key distribution procedure, which is given by

$$T_s = T_{b_j}(l_{n-1}) + T_G, \tag{5}$$

where $T_{b_j}(l_{n-1})$ is the time the last group-key distribution began, which is given by Eq. (2), and T_G is a protocol parameter to establish the interval between automatic group-key replacements, which is given by the Accept message in the group-key distribution. The round leader is considered absent if the new group key is not received up to $T_{k_j}(l_n)$.

When a node detects that round leader l_n has failed, it selects the next round leader, l_{n+1}, in the circular ordered list of the active nodes. A node j calculates a new expected time to receive the key, $T'_{k_j}(l_{n+1})$ based on the time to the node that will be the next leader, l_{n+1}, notices that the current round leader has failed and the delay to node j receive the key generated by the round leader l_{n+1}. Thus, in order to calculate the expected time to receive the group key after a leader failure, we propose the expression given by

$$T'_{k_j}(l_{n+1}) = T_s + \left(T_n \cdot H_{l_{n+1},l_n} + \delta\right) + \left(T_n \cdot H_{l_{n+1}j} \cdot T_n + \delta\right). \quad (6)$$

The time to start using the new group key, $T_{w_j}(l_{n+1})$, is recalculated according to Eq. (3) to the new round leader. The round-leader replacement procedure is accomplished for a node when it obtains the new group key. Then, if i leaders fail, then each node chooses the new leader, l_{n+i}, and estimates the maximum delay to receive the new group key by

$$T'_{k_j}(l_{n+i}) = T_s + \sum_{j=1}^{i} T_n \cdot H_{l_{n+j},l_{n+j-1}} + T_n \cdot H_{l_{n+i,j}} + (i+1)\cdot\delta. \quad (7)$$

Eq. (3) is then used to estimate the time to start using the new group key.

If the network experiences connection losses and congestion, the group key can get a great delay to be delivered and the round-leader failure procedure can be wrongly evoked. In this case, if a node obtains different group keys, but still has not start to use any of these keys, it chooses as the new group key the one sent by the oldest round leader and updates its estimated delay to deliver the group key, T_n.

5. Protocol analysis

5.1. Petri net analysis

State machine models of EGSR procedures are developed to validate the protocol characteristics, as shown in Fig. 9. These state machines were converted into a single Petri net and we used the ARP tool, version 2.3 [30] to

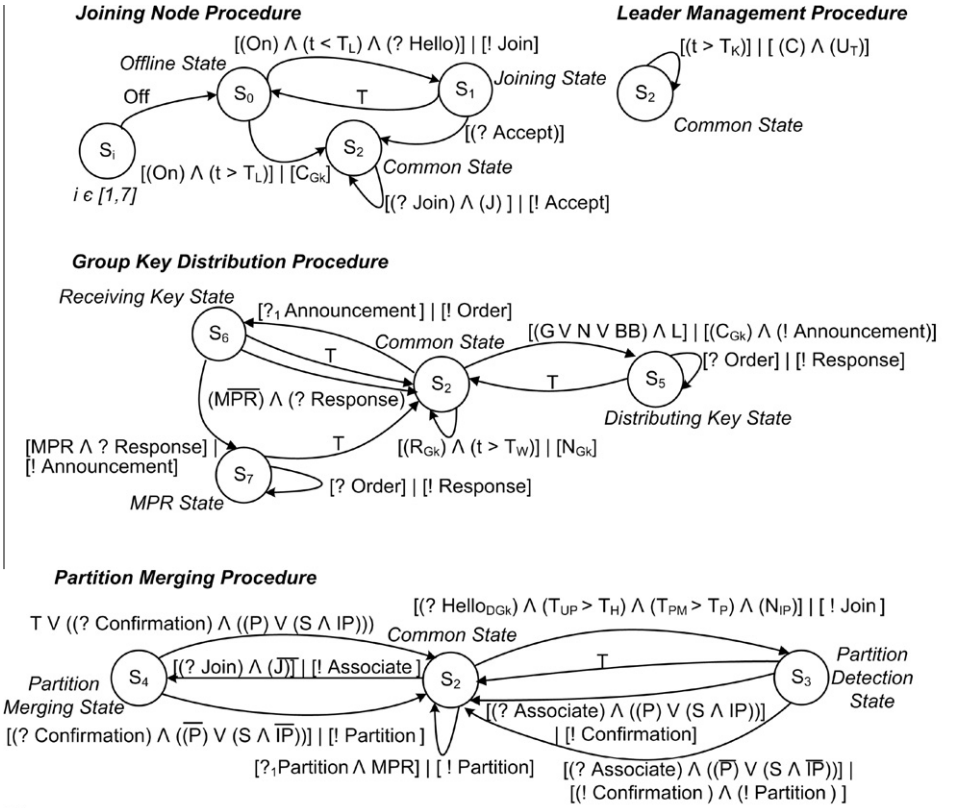


Fig. 9. State machines of EGSR procedures.

evaluate if the protocol fulfills the three classic properties: boundedness, liveness, and repetitiveness [21].

The results show that the protocol has the expected properties: boundedness, because protocol has a finite number of states; liveness, as there are no dead-locks, representing that all actions of the protocol are possible; and repetitiveness, because it is possible to return to the initial state from any state in the Petri net through at least one sequence of transitions. Then, the proposed protocol has neither loops nor states from which it is not possible to reach any other state.

5.2. Security analysis

In this section, we discuss potential security issues and how our protocol handles them.

5.2.1. Group-key disclosure

If a non-authorized node obtains the current group key Gk_n, it can sign routing control messages and damage the network. Nevertheless, this non-authorized node can only use the obtained group key by a restricted period of time, because the group key will be changed in the next group-key update. The non-authorized node cannot obtain the new key, because it does not have the private key and a valid certificate required by the Order message in the group-key distribution procedure. Assuming f_r is the average frequency of the automatic group-key distribution, which replaces the group key, then we can estimate that the malicious node will be excluded from the network in a period $p \leqslant 1/f_r$. Besides, if the non-authorized node does malicious actions, a BBDS created with any intrusion detection system and/or any trust system can detect the malicious action and send an alarm before the next automatic group-key distribution. This alarm triggers a new group-key distribution procedure, quickly excluding the non-authorized node.

5.2.2. Internal attacks against EGSR

We assume in EGSR that a non-authorized node can steal a group key. In these cases, the non-authorized node will participate in the network until the next group key distribution. Hence, we must guarantee that this non-authorized node cannot interfere on EGSR functions.

A malicious node that does not send or forward control messages may try to damage EGSR in four different ways: not starting a group-key distribution when it is the leader; not forwarding the key in the group-key distribution when it is an MPR; not forwarding the Partition message in the partition-merging procedure; and starting an EGSR procedure without finishing it. If a malicious node is the leader and does not start the group-key distribution, then it will be automatically replaced by all nodes, which will select a new leader in a distributed way. Hence, the attack will only cause an extra delay in the new group-key distribution. In the cases a malicious node is an MPR node that does not cooperate on the group-key distribution or does not forward Partition messages, this failure will probably be fixed by the other MPR redundant nodes and will only introduce a greater delay on the group-key distribution/partition-merging procedure. In the absence of redundancies on a flood with MPRs, EGSR also works, because the nodes that did not receive the new group key will create a small group with a different group key, which will trigger a partition-merging procedure with some non-malicious neighbor that obtained the group key. After the partition merging, all nodes will share the same group key and EGSR duties are accomplished. Finally, in the cases a malicious node starts an EGSR procedure without finishing it, the effect will only be a small delay. As we showed in Fig. 9, all the states of EGSR have a timeout to return to the Common State, avoiding dead-locks. Hence, malicious nodes can increase EGSR control overhead and delays to obtain a group key, but they cannot disrupt the protocol. It is worth mentioning that proposals based on contributory group key agreement have no tolerance to these kinds internal attacks, because the mechanisms always depends on the collaboration of the whole network. If a malicious node does not contribute generating its own public value when it is needed, the mechanism cannot be accomplished and the group key is not generated.

5.3. EGSR reliability in dynamic environments

Dynamic environments are common scenarios for ad hoc networks and are characterized by frequent network partitions and/or mobile nodes. To guarantee EGSR reliability in dynamic environments, we need small convergence times and also robustness to message losses and to node departures. Mobility can cause message losses if nodes that are neighbors loose their connectivity while they are in an EGSR procedure. Partitions can cause nodes to abruptly depart while an EGSR procedure is running.

The node departures caused by network partition events do not influence the joining node procedure or the group key distribution of EGSR. The group key distribution is not affected, because, if a group key does not reach all the nodes, a new leader will be automatically chosen to select and distribute a new key. Also, when a joining node tries to obtain the group key of a node and this node leaves the network, then the joining node must choose another node to obtain the group key. Indeed, the partition events only influence on the leader choice, because it changes the allocated IP list. The use of SOLSR guarantees that this list is always updated, but there is a convergence time to detect all the nodes that left the network in SOLSR. Hence, if a partition is formed and, just after that, a group key distribution is started, the new leader choice can have a greater delay.

The partition merging procedure robustness is based on the use of timers and on the control of the number of nodes in each partition. SOLSR guarantees the consistency on the number of nodes in each partition. Hence, every time a node tries to start a new partition merging procedure, only the group key of the greatest partition will be used in the network. Even if there are many simultaneous partition procedures, after all the partition merging events, all the nodes will share the group key of the greatest partition. If there are partitions with the same size, other parameters can be used to decide which partition has the greatest priority, such as the IP of the current leader. In addition, EGSR uses timers to avoid that a node starts many partition

procedures at the same time. Hence, if a node has recently participated of a partition merging, it will not start a new one for a short period, to increase the chances of network stabilization before the new partition merging procedure starts.

The mobility has no impact on EGSR if the nodes contact time is enough for the neighbor nodes exchanging at most four EGSR messages. If this assumption is not hold, the network routes are not available, and some EGSR procedure has begun, then this procedure will be restarted with another neighbor node. Hence, EGSR timeouts guarantee the absence of deadlocks. If routes are available and updated by SOLSR, then the EGSR procedures can continue using the routes to maintain the node communication, even though the nodes are not neighbors anymore. We do not consider that the flood is impacted by the mobility because of the redundancies of this process.

5.3.1. Analytical analysis of the convergence delay

In this section, we present an analytical evaluation of convergence delay of EGSR procedures, assuming no errors in the message transmission. The objective is to show that the protocol works even if the scenario is dynamic and the protocol procedures are frequently called.

In this analysis, we assume that the network topology is a grid for calculating the number of hops among nodes and the number of MPR nodes. For simplicity, we consider in this analysis that the average number of MPR nodes in a network is given by $\sqrt{N}$, where N is the number of nodes in the network. Moreover, the maximum number of hops will always be smaller than $\sqrt{N}$. For calculating the packet latency, we assume that all EGSR messages are smaller than 1500 bytes, to guarantee that messages are not fragmented by the transport protocol. This is an acceptable assumption if we consider the use of public keys of 1024 bits, a group key of 128 bits and a message authentication code with an output of 128 bits. These are common values when using AES [36], RSA [35], HMAC [19], and MD5 [34], which are known algorithms for symmetric cryptography, asymmetric cryptography, message authentication, and hash function, respectively. Hence, in the following analysis, we assume that the messages have size $M = 1500B$ for considering the worst case for EGSR.

First, we analyze the delay for distributing a new group key for the whole network. In this procedure, each node of the network, except for the round leader, receives an Announcement message, which is broadcasted by an MPR node, and then, the node exchanges two unicast messages with the MPR. We considered in this analysis that all neighbors verify the signature of the Announcement message simultaneously. Hence, we can estimate the delay for distributing the group key for the whole network, T_{dd}, by

$$T_{dd} = (M_p + 1)\cdot T_u + 2\cdot(N-1)\cdot T_u + (M_p+1) \cdot (T_{sg} + 2\cdot T_{ck}) + (2\cdot T_{sg} + 4\cdot T_{ck} + T_{ek} + T_{dk})\cdot(N-1), \quad (8)$$

where T_u is the delay for a node to send a message of size M to a neighbor, M_p is the number of MPR nodes, T_{sg} is the delay for signing the message with asymmetric cryptography, T_{ck} is the delay for checking this kind of signature, T_{ek} is the delay for encrypting the group key with asymmetric cryptography, and T_{dk} is the delay for decrypting the group key with asymmetric cryptography. Assuming the use of IEEE 802.11G, the messages can be sent in a rate of 54 Mbps and $T_u \approx (1500\cdot 8)/(54\cdot 10^6) = 0.22$ ms. We did not consider the propagation delay, because it is a negligible value of approximately 1 μs [15]. We also estimate the cryptographic delays using a portable computer and the OpenSSL benchmark [33], obtaining $T_{sg} = T_{dk} \approx 0.02$ s, and $T_{ck} = T_{ek} \approx 0.9$ ms. Hence, if we assume, for instance, a network with *100* nodes, then $T_{dd} \approx 6.6$ s.

We can also estimate the delay of the partition merging mechanism, T_{dp}. In this case, nodes exchange three unicast messages, and then, the smallest partition is flooded. We consider that the messages sent by the MPR nodes during the flood are processed by the neighbors simultaneously, because the message is sent in broadcast. Hence,

$$T_{dp} = (3 + M_{p_p})\cdot T_u + 3\cdot T_{sg} + 5\cdot T_{ck} + 2\cdot T_{ek} + 2\cdot T_{dk} + M_{p_p}\cdot(T_{es} + T_{ss} + T_{ds} + T_{cs}), \quad (9)$$

where M_{p_p} is the number of MPR nodes in the smallest partition, T_{es} and T_{ds} are the delays for encrypting and decrypting, respectively, the new group key with the current group key, T_{ss} is the delay for calculating the signature of the message using a message authentication code such as HMAC, and T_{cs} is the time for checking this signature. Using OpenSSL with a portable computer, we estimate $T_{es} = T_{ds} \approx 0.04$ ms, $T_{ss} = T_{cs} \approx 0.01$ ms. Assuming $T_u \approx 0.22$ ms and $P = 50$ nodes, than $T_{dp} \approx 0.1$ s.

Hence, even in a network with high mobility, we can guarantee that neighbor nodes usually have enough contact time to accomplish their EGSR procedures. We also observed that the partition merging procedure is very fast, because it mainly uses symmetric cryptography, while the group key distribution is slower due to use of asymmetric cryptography. The delay in the partition merging must be small, because nodes can only send routing messages between the partitions after the procedure is accomplished. The group key distribution delay, however, does not impact the routing protocol, because nodes control the time the group key will be used after the group key distribution began. Moreover, assuming the previous parameters, if the network partition event frequency is smaller than 1/6.6 = 0.15 events per second, we can also guarantee that frequent partitions will not decrease EGSR reliability.

5.4. Performance analysis

In the previous sections, we showed the security robustness and the reliability in dynamic environments of EGSR. Now, we analyze the energy consumption of EGSR and compare our proposal to protocols based on contributory group-key agreement. We show that EGSR has the smallest energy consumption and that it is efficient even on non-favorable scenarios.

In order to show that our protocol is suitable for energy constrained devices, we analyze the energetic performance with Matlab 7. We used a simple model to evaluate our protocol, in which we estimate the number of message transmissions and the number of messages receptions of

each node in the network. We considered a network free of errors or collisions, because we wanted to evaluate the impact of the message exchange of the protocols without the interference of a saturated network or other effects that could hide the main differences in the functionalities of the protocols. We also estimate the average number of cryptographic operations carried out. Unless we state differently, our scenario comprises 256 nodes, using the IEEE 802.11 standard, with an average node density of 0.0121 nodes/m^2, which corresponds to a dense community network [5]. We use these parameters to guarantee a non-favorable scenario in all evaluations, because both the density and the number of nodes influence protocols based on network flooding events. We consider a scenario free of errors, because we wanted to evaluate the protocols functions without the interference of external parameters. We consider that the average number of neighbors of each node is approximately constant even with the mobility. We use the default values for SOLSR message rate, which are one Hello per two seconds and one Topology Control (TC) per five seconds, as suggested in the RFC 3626 [8]. The other parameters used in this analysis are on Table 3. Because the frequency of nodes leaving/joining the network as well as the number of partition splitting/merging events depends on the scenario, we assume, in this first analysis, the same frequency for all these parameters. Consequently, the network begin with 256 nodes and ends with the same amount of nodes, because the number of nodes joining is the same of the number of nodes leaving the network. Also, the energy impact of each procedure over the whole system is proportional to the individual energy consumption of each procedure. In the second analysis, we compare the energy consumption of each procedure.

The amount of traffic exchanged among nodes depends on the size of each message. We specify the size of messages in EGSR, but this data is not specified in the other mechanisms. As a consequence, we only analyze the number of message transmitted and the amount of energy expended with cryptographic operations.

The energy consumption with cryptographic operations considered in this analysis are relative to "StrongARM" microprocessor, designed for embedded low-power environments. These microprocessors are suitable for cellular phones, PDAs and sensor nodes. Energy consumption is on Table 4 [6]. We choose RSA with 1024-bit key, Advanced Encryption Standard (AES) with 128-bit key length, and keyed-Hash Message Authentication Code (HMAC) with 128-bit key length as cryptography functions, because they are well-known and largely used.

Table 3
Parameters of the proposed EGSR protocol.

Variable	Value
Number of nodes	256
Average number of neighbors per node	8
Average number of MPRs among neighbors per node	4
Total time	1 h
Group-key distribution frequency	10 dist/h
Partition-splitting/merging frequency	10 part/h
Node-joining frequency	10 join./h
Node-exclusion frequency	10 exc./h
Average number of nodes on the revoked certificate list	30

Table 4
Cryptographic consumption for small devices based on the "StrongARM" microprocessor.

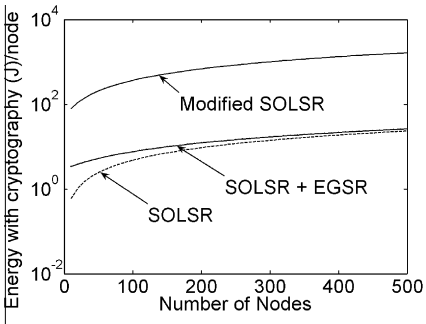
Algorithm	Action	Cost
RSA	Encrypt/Verify	0.74 mJ/1024-bit message
RSA	Decrypt/Sign	15 mJ/1024-bit message
AES	Encrypt/Decrypt	0.00217 mJ/128-bit block
HMAC	Sign/Verify	0.0108 mJ/1024-bit message
DH operation	Modular exponentiation	14.6 mJ/1024-bit message

5.4.1. Performance impact of proposed EGSR protocol

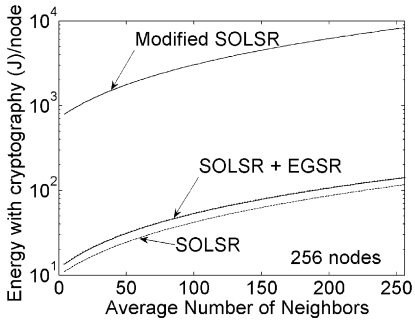
We compared the energy consumption with cryptographic operations of our proposal, the EGRS protocol, with SOLSR protocol to evaluate our proposal overhead over the routing protocol. Also, we compared the system composed of SOLSR and EGSR with a modified version of SOLSR using asymmetric cryptography, called Modified-SOLSR. The use of a private key to sign all messages in secure routing protocols based on asymmetric cryptography simplifies the identification of malicious nodes. Asymmetric cryptography, however, consumes much more energy than symmetric cryptography, as we show when we compare the modified version of SOLSR, which is based on asymmetric cryptography, with the traditional SOLSR and SOLSR + EGSR. By comparing the traditional SOLSR and SOLSR + EGSR, we can measure the impact of the proposed group-key management over the whole system. Indeed, SOLSR does not provide any group-key management, although the security of the routing protocol depends on it.

We consider the worst case performance conditions for our protocol, which means that the analyzed node always consumes the maximum energy per procedure. Therefore, we assume that the analyzed node is always an MPR in the group-key distribution procedure and its partition always changes the group key in the partition-merging procedures. Fig. 10(a), (b), and (c) show the energy consumed by one node during one hour.

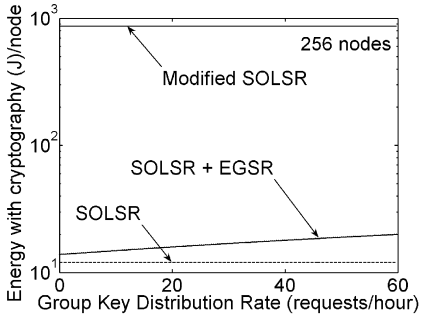
Fig. 10 shows the performance of SOLSR, Modified SOLSR, and SOLSR plus our proposal EGRS, denoted by SOLSR + EGRS, which makes the secure group-key management. Fig. 10(a) depicts the impact of the number of nodes over EGRS and SOLSR. Network size has a greater impact over the Modified-SOLSR and SOLSR than over EGSR, because both routing protocols often use many flooding events to maintain the link states. EGSR flood events are less common, because they occur only in a restricted area of the network during network partition merging events. EGSR increases SOLSR security without adding great energy consumption. Indeed, Modified-SOLSR consumes up to 62 times more energy than SOLSR + EGRS. Hence, the use of EGSR increases SOLSR security with a low energy consumption, which is adequate for networks composed of constrained devices. Fig. 10(b) shows the network density impact over EGSR, SOLSR, and Modified SOLSR. We considered that the network is composed of 256 nodes



(a) Per node energy consumption due to cryptography operations during one hour according to the network size.



(b) Per node energy consumption due to cryptography operations during one hour according to network density.



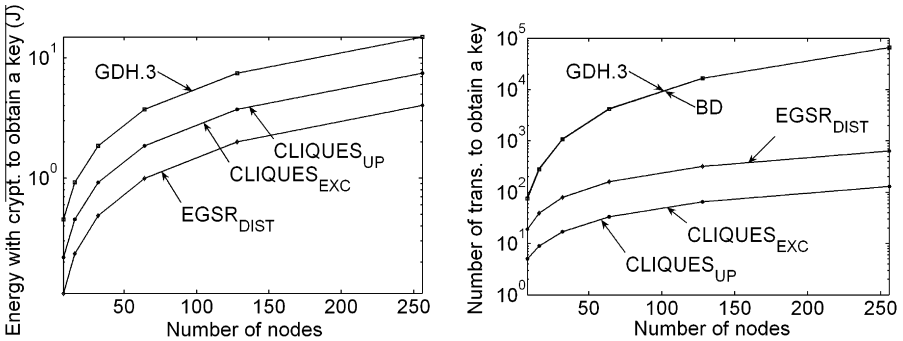
(c) Per node energy consumption due to cryptography operations during one hour according to key update frequency.

Fig. 10. The proposed EGSR worst case performance analysis. Per node energy consumption due to cryptographic operations during one hour.

and that the MPRs are calculated as if the nodes were disposed in a grid. In this configuration, Modified-SOLSR consumes up to 68 times more energy than SOLSR + EGSR. Besides, EGSR only consumes less than 21% of the energy of the system formed by SOLSR and EGSR. Finally, in Fig. 10(c) we observe the impact of EGSR when we increase the group-key distribution rate. The group key distribution consumes more energy than the partition merging and the joining node procedures, which means that it is the most impacting procedure of EGSR and gives an upper bound of EGSR energy consumption. Hence, we increase the group key distribution rate up to one distribution per minute, which is a high rate that could help in scenarios where authorized nodes send the group key to friends which are not authorized to access network resources. Even for a frequent key distribution rate of one key distribution per minute, our proposal EGSR consumes less than 39% of the total energy of the complete system composed by SOLSR plus EGRS, proving that EGRS has a small influence in the system performance.

We do not provide the analysis of the number of transmitted messages in the comparison of SOLSR + EGSR and Modified-SOLSR, because the number of messages exchanged by both SOLSR and Modified-SOLSR are just the same. Hence, the only interesting parameter is the number of messages exchanged by EGSR. Hence, in the following analysis we show the number of transmitted messages and energy consumption with cryptographic operations of each procedure of EGSR. Indeed, in the next analysis, we compare our proposal, EGSR, with contributory key agreement mechanisms: Group Diffie–Hellman (GDH.3) [37], Burmester–Desmedt (BD) [4], and the CLIQUES using GDH.3 [38]. In BD, all nodes spend the same amount of energy and, in GDH.3, there is a special node responsible for executing more cryptographic operations, assuming that at least one node has more CPU and energy power. The BD and GDH.3 protocols only generate a new group key, and consequently, the same algorithm is executed for the network initialization, network partition splitting/merging, and node joining/leaving. The CLIQUES using GDH.3 improves GDH.3 performance according to the type of the dynamic group event. These three mechanisms assume there is an auxiliary procedure to detect dynamic group events, to organize nodes, and to authenticate nodes in the group-key distribution. CLIQUES using GDH.3 also assumes there is a mechanism to elect a controller node and to maintain the data required by the controller node to perform dynamic group events. Our proposal not only detects all these dynamic group events, but also organizes and authenticates nodes.

In the next analysis, we compared our proposal, EGSR, with contributory key agreement mechanisms: Group Diffie–Hellman (GDH.3) [37], Burmester-Desmedt (BD) [4], and the CLIQUES using GDH.3 [38]. In BD, all nodes spend the same amount of energy and, in GDH.3, there is a special node responsible for executing more cryptographic



(a) Cryptographic operation energy consumption to obtain a group key.

(b) Number of message transmitted to obtain a group key.

Fig. 11. Group-key distribution due to key update and node exclusion in EGSR and in the main contributory group-key agreement proposals.

operations, assuming that at least one node has more CPU and energy power. The BD and GDH.3 protocols only generate a new group key, and consequently, the same algorithm is executed for the network initialization, network partition splitting/merging, and node joining/leaving. The CLIQUES using GDH.3 improves GDH.3 performance according to the type of the dynamic group event. These three mechanisms assume there is an auxiliary procedure to detect dynamic group events, to organize nodes, and to authenticate nodes in the group-key distribution. CLIQUES using GDH.3 also assumes there is a mechanism to elect a controller node and to maintain the data required by the controller node to perform dynamic group events. Our proposal not only detects all these dynamic group events, but also organizes and authenticates nodes.

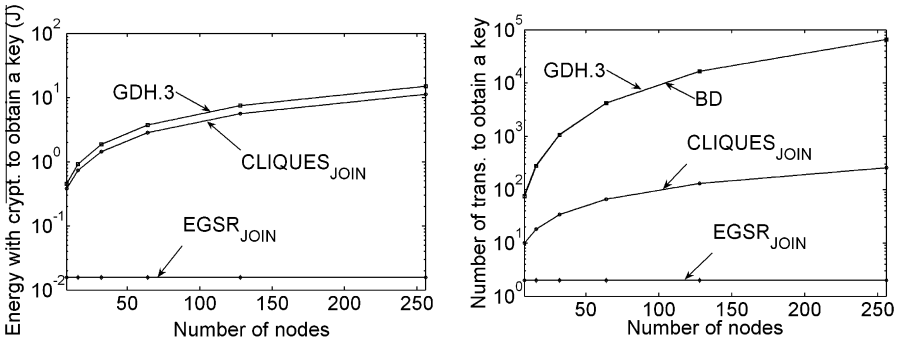
We present the sum of cryptographic operation energy consumption of all nodes and the number of transmitted messages during the group-key distribution. The analysis of cryptographic operation energy consumption does not consider energy consumed with the authentication in EGSR, because GDH.3 and CLIQUES only deal with the cryptographic operations to obtain a new group key, but they do not specify an authentication procedure. Therefore, we just compare the energy on the key distribution/agreement. Besides, BD uses many exponentiations with small exponents, while GDH.3 uses a few exponentiations with large exponents, which is much more energy consuming. Since our energy data refers to exponentiations with large exponents, we do not analyze the cryptographic operation energy consumption of BD.[1] In addition, the evaluated group-key agreement mechanisms assume that all group members can hear all messages and that nodes know the routes to each other in the network. Network routes, however, are not always available when distributing a group key for routing, and then, all messages of these protocols are flooded to guarantee that the message will always achieve the destination node.

In Fig. 11(a), we show the energy consumed with cryptographic operations to distribute a group key due to periodical replacements of the group key and to node exclusions. Here, we consider a network composed of 256 nodes with approximately eight neighbors per node and the number of MPRs is calculated as if the nodes were placed in a grid. We denote '$EGSR_{DIST}$' as the EGSR group-key distribution procedure, which is evoked in node exclusions, key updates, and the bad behavior detection system notifications. Also, we call '$CLIQUES_{UP}$' the group-key-update mechanism in CLIQUES using GDH.3 and '$CLIQUES_{EXC}$' the node exclusion in CLIQUES using GDH.3. We observe that our proposal outperforms the three protocols regardless of the number of nodes. GDH.3 consumes up to 3.72 times more energy than EGSR, while both '$CLIQUES_{EXC}$' and '$CLIQUES_{UP}$' consumes up to 1.8 more energy than EGSR. On the other hand, '$EGSR_{DIST}$' expends more messages than '$CLIQUES_{EXC}$' and '$CLIQUES_{UP}$', as we see in Fig. 11(b). The main reasons for that are the EGSR detection/warning of the need for a new group-key distribution and the messages exchanged to guarantee an authenticated communication. The control messages for these tasks are not taken into account in the analysis for CLIQUES, BD, and GDH.3, because these mechanisms do not specify how to accomplish these required tasks.

Fig. 12(a) and (b) show the energy consumption and the number of transmitted messages when a node joins the network for EGSR, denoted as '$EGSR_{JOIN}$', CLIQUES using GDH.3, denoted as '$CLIQUES_{JOIN}$', and GDH.3. We observe that EGSR is much less energy consuming than these mechanisms. Indeed, even CLIQUES using GDH.3 consumes up to 714 times more energy with cryptographic operations to distribute a key and 43 times more messages than EGSR. GHD.3 consumes up to 948 times more energy with cryptography and send about $7.34 \cdot 10^3$ times more messages than EGSR. EGSR outperforms these protocols because the new node event in EGSR is performed locally, while in the other mechanisms it demands message floods.

We also compared the energy consumption with cryptographic operations and the number of transmitted messages in network-partition-merging procedures and in network-initialization procedures assuming that all nodes join the network at the same time. We considered the worst case of partition merging for EGSR, which we call '$EGSR_{PART}$' in the graphs. Thus, the network is partitioned

[1] A previous work on measurements of energy costs using BD and GDH.3 empirically shows that BD consumes more energy with cryptography and message transmission than GDH.3 [6].

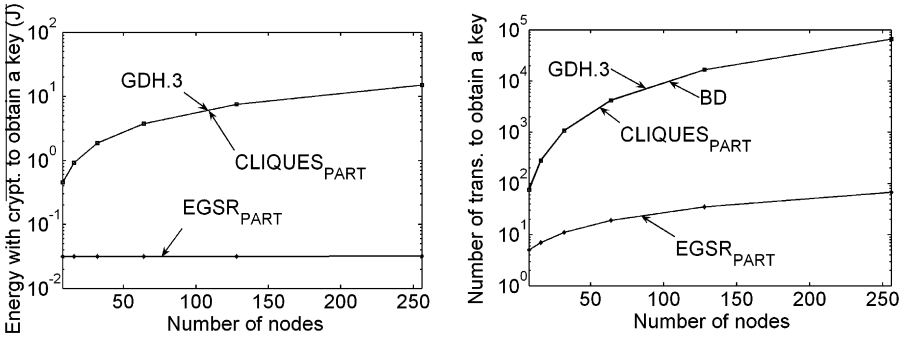


(a) Cryptographic operation energy consumption to obtain a group key.

(b) Number of message transmitted to obtain a group key.

Fig. 12. Group-key distribution due to a node joining event in EGSR and in the main contributory group-key agreement proposals.

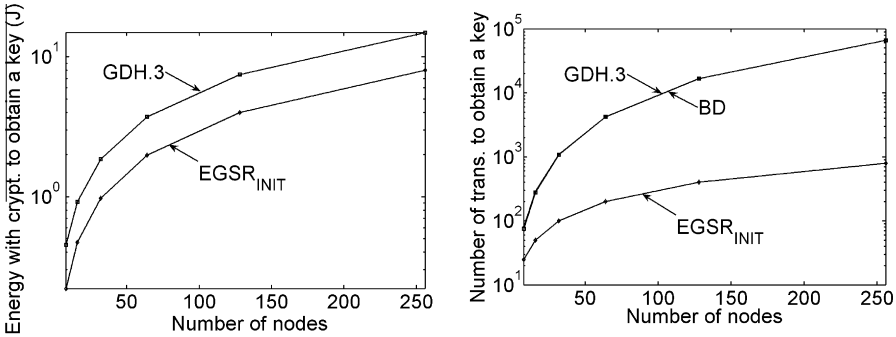
into two equal-sized groups to maximize the number of retransmissions of the Partition message. According to Fig. 13(a), CLIQUES using GDH.3 partition-merging procedure, denoted as '*CLIQUES*$_{PART}$', presents the same results than GDH.3. Indeed, CLIQUES using GDH.3 assumes that the partition merging should be a re-execution of GDH.3. Both Fig. 13(a) and (b) show that our protocol outperforms GDH.3, BD, and CLIQUES using GDH.3. EGSR consumption with cryptographic operations is up to 155 times smaller than the GDH.3 and the number of transmitted messages is up to 811 times smaller than the number of messages in GDH.3. In the initialization, depicted in Fig. 14, we observe our protocol initialization procedure performance, which we call '*EGSR*$_{INIT}$'. We do not compare EGSR with CLIQUES in the initialization, because this protocol suite is based on the assumption that the network initialization



(a) Cryptographic operation energy consumption to obtain a group key.

(b) Number of message transmitted to obtain a group key.

Fig. 13. Group-key distribution due to a partition-merging event in EGSR and in the main contributory group-key agreement proposals.



(a) Cryptographic operation energy consumption to obtain a group key.

(b) Number of transmitted messages to obtain a group key.

Fig. 14. Group-key distribution due to network initialization in EGSR and in the main contributory group-key agreement proposals.

is always performed as the original contributory key agreement protocol, which corresponds to GDH.3 in our analysis. Indeed, even though EGSR initialization mechanism is done through $n-1$ partition merging procedures, where n is the number of nodes, our protocol outperforms GDH.3 in up to 2 times, when comparing the energy with cryptographic operations and up to 70 times, when comparing the number of transmitted messages.

6. Conclusions

In this paper, we presented and evaluated the efficient group-key management for secure routing protocol (EGSR). Our protocol restricts non-authorized access to the network through periodic and triggered group-key replacement. EGSR with SOLSR makes ad hoc routing more secure against non-authorized nodes with small energy consumption. Moreover, the proposed protocol synchronizes the new group-key use and is robust against node failures and network partitions.

The analysis of our protocol indicated that it correctly works and is implementable. Besides, it is adequate to energy constrained devices. The analysis showed that EGSR consumes less energy and transmits fewer messages than BD, GDH.3, and CLIQUES using GDH.3, which are known protocols of group-key agreement. Moreover, the joining-node procedure and the partition-merging procedure of EGSR are energy efficient. This is an important characteristic, because these events are common in ad hoc networks and should be executed without large energy consumption. Therefore, the use of EGSR increases routing security in ad hoc networks without a great impact over network performance.

Acknowledgment

The authors thank CAPES, CNPq, FAPERJ, and FINEP.

References

[1] C. Adjih, T.H. Clausen, P. Jacquet, A. Laouiti, P. Muhltahler, D. Raffo, Securing the OLSR protocol, in: IFIP Med-Hoc-Net, June 2003, pp. 1–10.
[2] Y. Amir, Y. Kim, C. Nita-Rotaru, J.L. Schultz, J. Stanton, G. Tsudik, Secure group communication using robust contributory key agreement, IEEE Transactions on Parallel and Distributed Systems 15 (5) (2004) 468–480.
[3] M.S. Bouassida, I. Chrisment, O. Festor, Efficient group key management protocol in MANETs using the multipoint relaying technique, in: International Conference on Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies (ICN/ICONS/MCL 2006), April 2006, pp. 64–71.
[4] M. Burmester, Y. Desmedt, A secure and efficient conference key distribution system, Lecture Notes in Computer Science 950 (1998) 275–286.
[5] M.E.M. Campista, I.M. Moraes, P. Esposito, A. Amodei Jr., L.H.M.K. Costa, O.C.M.B. Duarte, The ad hoc return channel: a low-cost solution for brazilian interactive digital TV, IEEE Communications Magazine 45 (1) (2007) 136–143.
[6] D.W. Carman, P.S. Kruus, B.J. Matt, Constraints and Approaches for Distributed Sensor Network Security (final). Tech Report 00-010, NAI Labs, September 2000.
[7] H. Chan, A. Perrig, D. Song, Random key predistribution schemes for sensor networks, in: IEEE Symposium on Security and Privacy, May 2003, pp. 197–213.
[8] T. Clausen, P. Jacquet, Optimized Link State Routing Protocol (OLSR), RFC 3626, October 2003.
[9] D.O. Cunha, O.C.M.B. Duarte, G. Pujolle, A cooperation-aware routing scheme for fast varying fading wireless channels, IEEE Communications Letters 12 (10) (2008) 794–796.
[10] W. Diffie, M.E. Hellman, New directions in cryptography, IEEE Transactions on Information Theory IT-22 (6) (1976) 644–654.
[11] N. Fernandes, O.C.M.B. Duarte, An efficient group key management for secure routing in ad hoc networks, in: IEEE Globecom 2008 Computer and Communications Network Security Symposium (GC'08 CCNS), December 2008, pp. 1–5.
[12] N.C. Fernandes, M.D.D. Moreira, O.C.M.B. Duarte, A self-organized mechanism for thwarting malicious access in ad hoc networks, in: 2010 Proceedings IEEE INFOCOM, San Diego, CA, USA, March 2010, p. 5.
[13] R.C. Gangwar, A.K. Sarje, Secure and efficient dynamic group key agreement protocol for an ad hoc network, in: International Symposium on Ad Hoc and Ubiquitous Computing (ISAUHC '06), December 2006, pp. 56–61.
[14] A. Hafslund, A. Tonnesen, R.B. Rotvik, J. Andersson, O. Kure, Secure extension to the OLSR protocol, in: OLSR Interop and Workshop, San Diego, California, August 2004, pp. 1–4.
[15] IEEE 802.11 Working Group, IEEE standard 802.11-2007, Standard 802.11, IEEE Computer Society, November 2007.
[16] Y. Kim, A. Perrig, G. Tsudik, Simple and fault-tolerant key agreement for dynamic collaborative groups, in: Proceedings of the Seventh ACM conference on Computer and Communications Security (CCS '00), ACM, New York, NY, USA, 2000, pp. 235–244.
[17] J. Kong, P. Zerfos, H. Luo, S. Lu, L. Zhang, Providing robust and ubiquitous security support for mobile ad-hoc networks, in: Ninth International Conference on Network Protocols (ICNP'01), November 2001, pp. 251–260.
[18] E. Konstantinou, Cluster-based group key agreement for wireless ad hoc networks, in: Third International Conference on Availability, Reliability and Security (ARES 08), March 2008, pp. 550–557.
[19] H. Krawczyk, M. Bellare, R. Canetti, HMAC: Keyed-Hashing for Message Authentication, RFC 2104, February 1997.
[20] K. Kumar, V. Sumathy, J.N. Begum, Efficient region-based group key agreement protocol for ad hoc networks using elliptic curve cryptography, in: IEEE International Advance Computing Conference (IACC 2009), March 2009, pp. 1052–1060.
[21] D. Lamch, Verification and analysis of properties of dynamic systems based on petri nets, in: International Conference on Parallel Computing in Electrical Engineering (PARELEC'02), 2002, pp. 92–94.
[22] L. Lazos, R. Poovendran, Power proximity based key management for secure multicast in ad hoc networks, Wireless Network 13 (1) (2007) 127–148.
[23] D. Li, S. Sampalli, An efficient group key establishment in location-aided mobile ad hoc networks, in: Second ACM International Workshop on Performance Evaluation of Wireless ad hoc, Sensor, and Ubiquitous Networks (PE-WASUN'05), 2005, pp. 57–64.
[24] J.H. Li, R. Levy, M. Yu, B. Bhattacharjee, A scalable key management and clustering scheme for ad hoc networks, in: International Conference on Scalable Information Systems (INFOSCALE'06), vol. 28, 2006, pp. 1–10.
[25] Z. Li-Ping, C. Guo-Hua, Y. Zhi-Gang, An efficient group key agreement protocol for ad hoc networks, in: Fourth International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM '08), October 2008, pp. 1–5.
[26] J. Liu, D. Sacchetti, F. Sailhan, V. Issarny, Group management for mobile ad hoc networks: design, implementation and experiment, in: Sixth International Conference on Mobile Data Management (MDM'05), ACM Press, 2005, pp. 192–199.
[27] H. Luo, J. Kong, P. Zerfos, S. Lu, L. Zhang, URSA: ubiquitous and robust access control for mobile ad hoc networks, IEEE/ACM Transactions on Networking 12 (6) (2004) 1049–1063.
[28] L. Luo, R. Safavi-Naini, J. Baek, W. Susilo, Self-organised group key management for ad hoc networks, in: ACM Symposium on Information, Computer and Communications Security (ASIACCS'06), March 2006, pp. 138–147.
[29] S. Marti, T.J. Giuli, K. Lai, M. Baker, Mitigating routing misbehavior in mobile ad hoc networks, in: Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking (MobiCom), New York, NY, USA, 2000, pp. 255–265.
[30] C.A. Maziero, ARP: Petri Net Analyzer, 1990.
[31] J.V.D. Merwe, D. Dawoud, S. McDonald, A survey on peer-to-peer key management for mobile ad hoc networks, ACM Computing Surveys 39 (1) (2007).
[32] Q. Niu, Study and implementation of a improved group key protocol for mobile ad hoc networks, in: Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and

Parallel/Distributed Computing (SNPD 2007), Vol. 1, July 2007, pp. 304–308.
[33] OpenSSL Core and Development Team, OpenSSL - Cryptography and SSL/TLS Toolkit, April 2010, <http://www.openssl.org>.
[34] R. Rivest, The MD5 Message-Digest Algorithm, RFC 1321, April 1992.
[35] R.L. Rivest, A. Shamir, L. Adleman, A method for obtaining digital signatures and public-key cryptosystems, Communications on ACM 26 (1) (1983) 96–99.
[36] J. Schaad, R. Housley, Advanced Encryption Standard (AES) Key Wrap Algorithm, RFC 3394, September 2002.
[37] M. Steiner, G. Tsudik, Waidner, Diffie–Hellman key distribution extended to group communication, in: CCS '96: Proceedings of the Third ACM Conference on Computer and Communications security, 1996, pp. 31–37.
[38] M. Steiner, G. Tsudik, M. Waidner, Key agreement in dynamic peer groups, IEEE Transactions on Parallel and Distributed Systems 11 (8) (2000) 769–780.
[39] J.C.M. Teo, C.H. Tan, Energy-efficient and scalable group key agreement for large ad hoc networks, in: Second ACM International Workshop on Performance Evaluation of Wireless ad hoc, Sensor, and Ubiquitous Networks (PE-WASUN'05), 2005, pp. 114–121.
[40] A. Tonnesen, Implementing and Extending the Optimized Link State Routing Protocol, Master's Thesis, University of Oslo, August 2004.
[41] P.B. Velloso, R.P. Laufer, O.C.M.B. Duarte, G. Pujolle, A trust model robust to slander attacks in ad hoc networks, in: Worshop in Advanced Networking and Communications (ANC) jointly with ICCCN'2008, August 2008, pp. 1–6.
[42] B. Wang, S. Soltani, J.K. Shapiro, P.-N. Tan, Local detection of selfish routing behavior in ad hoc networks, in: International Symposium on Parallel Architectures, Algorithms, and Networks, IEEE Computer Society, Los Alamitos, CA, USA, 2005, pp. 392–399.
[43] L. Zhou, Z.J. Haas, Securing ad hoc networks, IEEE Network 13 (6) (1999) 24–30.

Natalia Castro Fernandes received an electrical engineering degree in 2006 and a M.Sc. degree in electrical engineering in 2008 from UFRJ, Brazil. Currently, she is a D.Sc. student with UFRJ. Her major research interests are in ad hoc networks and security.

Otto Carlos M. B. Duarte received the Electronic Engineer degree and the M.Sc. degree in electrical engineering from UFRJ, Brazil, in 1976 and 1981, respectively, and the Dr. Ing. degree from ENST/Paris, France, in 1985. Since 1978, he has been a Professor with UFRJ. His major research interests are in multicast, QoS guarantees, security, and mobile communications.