Wireless Pers Commun (2011) 56:625–636
DOI 10.1007/s11277-010-9994-7

A Fair and Secure Cluster Formation Process for Ad Hoc Networks

Helena Rifà-Pous · Jordi Herrera-Joancomartí

Published online: 20 April 2010

Abstract An efficient approach for organizing large ad hoc networks is to divide the nodes into multiple clusters and designate, for each cluster, a clusterhead which is responsible for holding intercluster control information. The role of a clusterhead entails rights and duties. On the one hand, it has a dominant position in front of the others because it manages the connectivity and has access to other node's sensitive information. But on the other hand, the clusterhead role also has some associated costs. Hence, in order to prevent malicious nodes from taking control of the group in a fraudulent way and avoid selfish attacks from suitable nodes, the clusterhead needs to be elected in a secure way. In this paper we present a novel solution that guarantees the clusterhead is elected in a cheat-proof manner.

Keywords Clustering · Fairness · Leader election · Mobile ad hoc networks · Security

1 Introduction

The proliferation of wireless devices has motivated the deployment of big public ad hoc networks. Traditional flat topologies encounter significant scalability issues to manage these networks, so clustering protocols that organize mobile nodes into groups are preferred. In cluster-based architectures, each group or cluster has a clusterhead which is responsible for traffic management. The clusterhead possesses valuable information about nodes' location and their contacts. Thus, its election is critical. From the viewpoint of building a scalable and efficient network, clusterheads shall be the nodes that can guarantee a backbone at a minimum cost. From the point of view of establishing a reliable and robust network, they should belong to honest users.

H. Rifà-Pous (✉) · J. Herrera-Joancomartí
Internet Interdisciplinary Institute (IN3), Universitat Oberta de Catalunya, Barcelona, Spain
e-mail: hrifa@uoc.edu

J. Herrera-Joancomartí
Department of Information and Communications Engineering,
Universitat Autònoma de Barcelona, Bellaterra, Spain
e-mail: jherrera@deic.uab.cat

While there is a great deal of research activity in the line of finding the optimal clustering algorithm for an efficient and long-lived network (an overview is presented in [1]), few works have dealt with the security challenges of clusterhead election algorithms. This paper discusses the problem of secure clusterhead election in wireless ad hoc networks. The contribution is the identification and classification of security challenges of clusterhead election algorithms and settling the bases for building a secure and fair protocol. The paper proposes distributed algorithms for electing a leader in a cheat-proof manner.

The rest of the paper is organized as follows. In Sect. 2 we identify security challenges of clusterhead election algorithms and define some general rules to prevent them. An overview of the related work is given pointing out the open research issues. Section 3 describes the proposed cluster formation process. In Sect. 4 we evaluate the robustness of the protocol from the security point of view, and finally, in Sect. 5 the conclusions of the work are presented.

2 Overview

Cluster formation protocols determine how to group nodes and which ones are the best to be set up as clusterheads. Each protocol uses a specific utility function based on some metric (e.g. node degree, remaining battery life, node mobility). When a clustering process is started, nodes have to declare and disseminate the value of the parameters involved in the utility function. Then, the function is applied and the optimal clusterheads are elected.

The election of the appropriate node to be a clusterhead is threaten by two main vulnerabilities: (1) *Selfish attacks* in which nodes cheat in order to not be elected clusterheads and keep energy and bandwidth for themselves, and (2) *Greedy attacks* in which nodes cheat to be elected clusterheads and have access to valuable information of the network topology and control the traffic of the members of their cluster. To prevent them, the election of the clusterhead must be fair (i.e. the elected node must be the one that maximizes the utility) and the role must be rotative. Rotation avoids, on the one hand, that a node gets burdened with more duties than others, and on the other hand that it can freely dominate the cluster.

To the best of our knowledge, there is not any proposal of cluster formation for mobile ad hoc networks that protects the utility function values ensuring the correctness of the process to choose the optimal node for being the clusterhead. Prior work in securing cluster formation protocols for ad hoc networks is based on random algorithms [2–4] that assume an homogeneous network in which any node is equally valid to be the clusterhead, or on voting algorithms [5] that use and protect the utility-based election procedure, but not the data that is used for weighing the nodes.

In the context of wireless sensor networks, Sun et al. describe a proposal [6] that deals with the problem of performing an action based on some metric, which by default, is not verifiable. However, they assume a static network in which each node knows its 1-hop neighbors. Other solutions for sensor networks [7,8] involve the existence of a trusted base station that leads the process. None of the approaches is suitable for generic, mobile, and infrastructureless ad hoc networks.

3 Fair and Secure Cluster Formation

The proposed protocol provides fairness and security to the cluster formation process using a distributed algorithm that monitors the system and checks if the data claimed by a node

is coherent and reliable. We assume the majority of network users are honest. In order to authenticate the information provided by each user and be able to track its behavior, a mechanism based on symmetric ciphers, hash functions and digital signatures has been developed.

The cluster formation process is composed of three subprotocols as follows:

1. The **Cost Discovery Protocol** provides reliable information about the measure used to evaluate the suitability of a node to be a clusterhead. Our protocol uses a utility function based on the degree, i.e., the number of one hop neighbors. In this phase, nodes publish the number of neighbors they have and exchange their particular perspective of the network to create a consensual view.
2. The **Clusterhead Designation Protocol** selects the best suited node in a neighborhood to be the clusterhead based on some verifiable data. The information provided by each node in the Cost Discovery Protocol is correlated with data from other nodes.
3. The **Cluster Management Protocol** manages the cluster formation and actualization once the clusterhead is set up. Cluster nodes endorse the role of the elected clusterhead by signing a claim. Then, the clusterhead is set up as a trusted authority for that cluster. The clusterhead controls the authorized nodes that can work in its neighborhood through the issuance of public key certificates that attest it.

The proposed cluster formation process can be applied in flat networks that want to create its first topology, or in clustered networks that have to update some clusterhead roles. It is possible that clusterhead election leads to the abolition of a cluster, or on the contrary, results in the division of a cluster and the creation of new groups.

Following, we first depict the cryptographic framework of the designed process and describe the notation we use hereinafter (see Table 1). Then, we detail the steps of the Fair and Secure Cluster Formation Process.

3.1 Cryptographic Tools

The security mechanisms developed in the cluster formation process take a public key infrastructure approach to control the users' access to the network. We propose the use of two infrastructures, one external and another one local or internal of the cluster. The external one is supported by a global Internet Certification Authority (CA) and it is used to control the identity of the nodes and avoid spoofing attacks. The local authority issues certificates which are bound to local short keys easier to manage and operate with. These certificates control the access of the nodes to the resources of the network and are the means to expel misbehaving nodes off the cluster.

The management of the local CA is hold by clusterheads, which issue certificates using their local private key. The authorization certificate of the clusterhead is supported by the identity key signature of a group of nodes big enough to represent the support of the network with it. As clusterheads constitute the main elements of the network backbone, using identity keys to generate their certificates provides a means to validate the correctness of the assertion beyond the clusterhead coverage area.

The designed protocols use different cryptographic algorithms. Symmetric ciphers are used to construct commitments. Digital signatures are the basic tool for providing authenticity when two peers first interact. Finally, hash functions are employed to create hash chains that efficiently produce one time authenticity tokens.

Hash Chains, first proposed by Lamport [9], are composed of a sequence of values that can only be computed in one-way. A hash chain of length N is constructed by applying

Table 1 Protocols notation

Notation	Description
v_i	Node of a network. Nodes can be qualified as clusterheads h, clustergateways g.
NID	Network identifier. It remains unchanged during the whole life time of the ad hoc network.
UID_i	User identifier belonging to node v_i. The identifier is the hash of his identity certificate.
$NodeID_i$	Node identifier in the network: MAC and IP addresses
CID	Cluster identifier. It equals the present clusterhead UID_h.
CUN	Cluster update number. It is a sequence number from 1 to 10.
$ACert_i$	Authorization certificate of node v_i.
$IdCert_i$	Identity certificate of node v_i.
LID_i	Identifier of the local public key of node v_i. It is the hash of the public key.
$TChain_i$	Top value of a hash chain used to authenticate some messages of node v_i.
$Chain_i^n$	nth value of a hash chain of node v_i.
$ACID_i$	Authorization certificate identification number of node v_i. It is a unique identifier.
k_i	Temporal symmetric key of node v_i.
$H(.)$	Hash function
$MAC_{k_i}(.)$	MAC function using the key k_i.
$E_{k_i}(.)/D_{k_i}(.)$	Cipher/Decipher a message using the symmetric key k_i.
$S_{sk}(.)/V_{pk}(.)$	Digital sign/validate a signature using a secret/public key.
lsk_i/lpk_i	Local secret/public key of node v_i.
$idsk_i/idpk_i$	Identity secret/public key of node v_i.
$degree_i$	Number of neighbors, i.e. degree, of node v_i.

a one-way hash function $H(.)$ recursively to an initial seed value $Chain^N$: $Chain^{N-1} = H(Chain^N), Chain^{N-2} = H(Chain^{N-1}), \ldots, Chain^0 = H^N(Chain^N)$. In general, $Chain^n = H^{N-n}(Chain^N)$. The top element of the chain $Chain^0$ has to be bound to a user. To do so, in our scheme we include this element in the authorization certificate. Then, the rest of the chain elements are revealed in ascending order ($Chain^1, Chain^2, .. Chain^N$) when the user needs to claim his presence.

3.2 Cost Discovery Protocol

The Cost Discovery Protocol is the first phase of a clustering process. Its aim is that nodes interchange information about their utility to be clusterheads. We present the protocol in Protocol 1 assuming the prior existence of a cluster-based architecture in the network. The update is of cluster cl_a which clusterhead is node h_a.

We will now review the protocol focusing on challenges and implemented prevention mechanisms to avoid attacks. The process is activated with a signed $RecAd$ (Step 1) that carries cluster unique identifiers that will be used in the whole process. The signature protects such identifiers from replaying attacks.

Protocol 1

1. **Init** $[h_a \rightarrow all]$
   $h_a \in cl_a$ triggers the Cost Discovery Process. h_a does:

(a) Compose a Reclustering Message RM: $RM = \{NID, CID, CUN\}$.
(b) Sign RM, get $RecAd = \{RM, S_{lsk_a}(RM)\}$ and send it. Trigger timers $T_1 = (1/v) \cdot d^{\lambda}$ and $T_2 = 3 \cdot T_1$, with v: bandwidth speed, d: cluster density, and $\lambda > 1$ which depends of the exponential backoff.

2. **Presence Announcement** $[v_j \in cl_a \rightarrow 1hop]$
After receiving a $RecAd$ message, a node $v_j \in cl_a$ verifies its signature ($V_{lpbk_a}(RecAd)$) and if correct announces its presence with a $Hello$ message:
$Hello_j = \{NID, CID, CUN, UID_j, ACID_j, Chain_j^{CUN}\}$.
3. **Degree Commitment** $[v_i \in cl_a \rightarrow cl_a]$
Nodes $v_i \in cl_a$ gather the information in $Hello$ messages to construct a $List$ file with all their neighbors identifiers, and a $ListChain$ that demonstrates the veracity of the $List$.

(a) For each received $Hello$ from v_j, check the Chain authenticity: $H^{CUN}(Chain_j^{CUN}) \stackrel{?}{=} TChain_j$ (value of $ACert_j$), and the validity of the authorization certificate: $V_{lpk_a}(ACert_j)$.
(b) If checks are correct, accept node v_j as a neighbor.
(c) After timeout T_1, compose $List_i$ with received UIDs of nodes in cl_a and generate a commitment
   i. Compose a list $ListChain_i$ with the received $Chain$ values, and compute $H(ListChain_i)$.
   ii. Generate a random symmetric key k_i and compute $c_i = E_{k_i}(H(ListChain_i))$.
(d) Compose $Degree_i = \{NID, CID, CUN, List_i, c_i\}$, sign it, and send $DegAd_i = \{Degree_i, S_{lsk_i}(Degree_i)\}$.

4. **Degree Recount** $[v_j \in cl_a]$
A node v_j receives $DegAd_i$ from nodes in cl_a, and from clusterheads v_b of neighboring nodes.

(a) v_j conducts a signature validation of $DegAd_i$ messages $V_{lpk_i}(DegAd_i)$.
(b) If $V_{lpk_i}(DegAd_i) = false$:
   i. Send an Error message in multicast for cl_a nodes. Request a resend.
   ii. If resends cannot be validated, discard the message and put node v_i in the reputation list.

Nodes in cl_a wait T_2 time to receive $DegAd_i$ messages. After T_2, incoming $DegAd_i$ are discarded.
5. **Commitment Opening** $[v_i \in cl_a \rightarrow cl_a]$
Node v_i opens the commitment delivered in $DegAd_i$ messages by sending $KeyAd_i = \{k_i, Chain_i^{CUN}\}$.
6. **Commitment Validation** $[v_j \in cl_a]$
Nodes $v_j \in cl_a$ receive $KeyAd_i$ messages from cluster members v_i and do:

(a) Decipher c_i field of each $DegAd_i$ message: $H'(ListChain_i) := D_{k_i}(c_i)$.
   i. If v_i has not delivered the right key, it is considered malicious and put in the repudiation list.
(b) Check the correctness of $H'(ListChain_i)$ using the $Chain$ values received in $KeyAd$ messages and computing the hash of them: $H'(ListChain_i) \stackrel{?}{=} H(ListChain_i)$. If it does not validate:

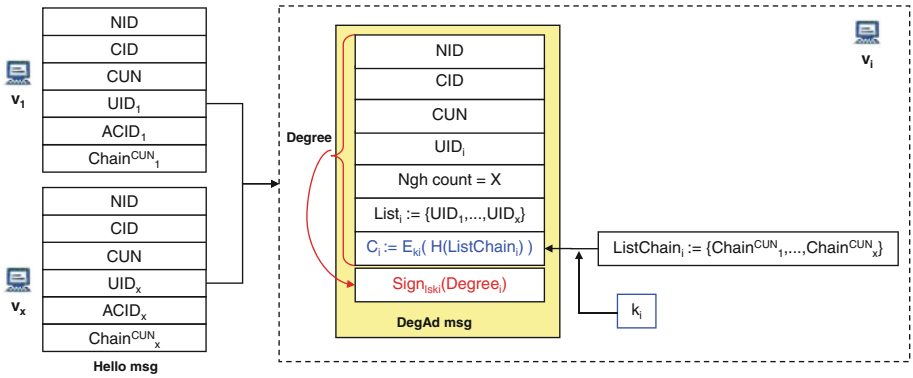

Fig. 1 Degree message composition

i. Check weather $\forall Chain_j \in ListChain_i$ is correct: $H^{CUN}(Chain_j^{CUN}) \stackrel{?}{=} TChain_j$.
ii. If $Chain_j$ is correct, then c_i is incorrect and node v_i is put in the repudiation list.
iii. If $Chain_j$ is incorrect and v_j cannot deliver the right value, v_j is put in the repudiation list.

Members of the cluster under clusterhead election send $Hello$ messages (Step 2) to announce their presence. $Hello$ messages contain information to identify and address the originating node UID_i, a proof of node's membership in the cluster $ACID_i$, and a message authenticity attribute $Chain_i^{CUN}$ which is used to demonstrate that the message has been generated by the declared user v_i in that precise reclustering update. On the other hand, the $ACID_i$ is a link to the authorization certificate $ACert_i$ that proves the bound between node's identifiers (UID_i) and network/link layer addresses, and assures the honesty of the individual. However, the cited three elements do not keep off resending attacks. These can be detected monitoring low-level traffic, or in the final phase of the protocol, when the received information is correlated.

After the node's announcement phase, nodes calculate their degree and send a $DegAd$ message (Step 3) to state it. In order to proof its degree, a node must be able to demonstrate it knows the $Chain^{CUN}$ of its neighbors. As $Chain^{CUN}$ are sensitive and their revelation could led to construct fake $DegAd$ messages (reusing chain elements), they must be protected. Node v_i includes $Chain_j^{CUN}$ of all received $Hello_j$ messages in a list named $ListChain_i$, which is hashed and ciphered using a symmetric key k_i. The result is the degree commitment c_i, which is inserted in $DegAd_i$ and constitutes a proof of a v_i's knowledge of its neighborhood in a particular moment. Figure 1 depicts the contents of $DegAd$ and how they are generated. Besides, $DegAd$ contains NID, CID and CUN attributes to prevent replay attacks. All $DegAd$ contents are signed with the node's local private key to provide authenticity and integrity to the message.

Once $DegAd$ messages have been delivered to all nodes in the cluster, at time T_2, commitments are opened with a $KeyAd$. Owners of $DegAd_i$ send $KeyAd_i$ messages (Step 5) in which they reveal their $Chain_i^{CUN}$ and the symmetric key k_i used in the commitment computation. k_i is used to decipher the commitment c_i. The result has to be a message with an $H(ListChain)$ parameter. Since this message follows a particular template, when a node deciphers a commitment c (Step 6) it recognizes if the

key it is using is correct or not. This prevents forwarding nodes changing the keys of some $KeyAd$ messages in order to falsely accuse their owners of having generated $DegAd$'s with fake lists of neighbors ($List$ and $ListChain$ fields). Originators of $DegAd$ messages are only put in the black list when its commitment is successfully opened, but its contents do not match the reported list of neighbors stated in $DegAd$.

3.3 Clusterhead Designation Protocol

The Clusterhead Designation Protocol is initiated once the metric evidence about the utility of each node to be a clusterhead has been distributed. The protocol objective is to use the evidence to select the most appropriate node as the clusterhead. In Protocol 2 we detail the steps of the protocol.

Protocol 2

1. **Clusterhead Selection** $[v_i \in cl_a]$
   Each node v_i does:
   (a) Contrast the information received in the $List$ attribute of $DegAd$ messages, and create a $List'$.
      i. Put users with more than a 20% of intra-cluster unidirectional evidence in RepList.
      ii. In each $List_j$, with $v_j \in cl_a$, remove all the evidence generated or reported by malicious users.
      iii. In each $List_j$, with $v_j \in cl_a$, discard intra-cluster degree evidence that is not bidirectional.
   (b) Select the clusterhead candidate as the node with the contrasted highest degree in $List'$.
      i. Prioritize nodes that have not been clusterheads before; modify the degree of old clusterheads with a factor of $\frac{x}{x+1}$, with x the distance to the cluster instance in which they were clusterheads.
      ii. If the selected node is in the list of some neighboring clusterhead v_b (i.e. is a clustergateway), validate $DegAd_b$ and $ACert_b$. If v_b evidence is verified, select another candidate.
      iii. If no node remains in the list of eligible clusterheads, the cluster is abolished.
2. **Clusterhead Candidacy Announcement** $[h_i \rightarrow 1hop]$
   The candidate clusterhead h_i does:
   (a) Generate a random number $seed_i^*$ and compute a chain of 10 hashes to find $TChain_i^*$.
   (b) Compose a message with the mandatory fields of a clusterhead certificate, which include the CID^* of the new cluster, the computed $TChain_i^*$, the list $OldH^*$ with the $UIDs$ of the last three clusterheads, and the list $List_i'$ of $UIDs$ of clusterhead's verified neighbors. The message is: $ChCand_i = \{NID, CID^*, Validity^*, NodeID_i, LID_i, TChain_i^*, OldH^*, List_i'\}$.
   (c) Complete the $ChCand_i$ message with CID and CUN of the present reclustering update, and its degree (i.e. the order of $List_i'$): $ChCandExt_i = \{ChCand_i, CID, CUN, \mid List_i' \mid\}$.

(d) Sign $ChCandExt_i$: $IamCh_i = \{ChCandExt_i,\ S_{ipvk_i}(ChCandExt_i)\}$. Send $IamCh_i$.

3. **Clusterhead Candidacy Support** $[v_j \in cl_a \rightarrow h_i]$
Nodes that receive the $IamCh_i$ message and are in the $List'_i$ field of it verify its contents and signature.

   (a) If they agree on the candidature, they countersign and send it: $S_{ipvk_j}(ChCand_i)$.
   (b) If they do not agree, they send a signed $Error$ message declaring the problem.

4. **Clusterhead Setting** $[h_i \rightarrow 1hop]$
The clusterhead candidate v_i receives signed certificate proposals and verifies the signature.
If 50% of nodes in $List'_i$ sign $ChCand_i$, node v_i gets its role. This signed message $ChCand$ is the proof of its role in front of cluster internal members, and other clusterheads of the network.

The Clusterhead Designation Protocol is cheat-proof since the naming of the clusterhead must be approved by the majority of neighbors on that cluster using verifiable evidence. The evidence is protected against reply attacks with the inclusion of NID, CID, and CUN attributes that specifically point to a reclustering process. Faking nodes are detected and expelled.

Occasionally, a malicious node that wants to control a cluster can legitimately become a clusterhead. Then, a new thread appears due to the authority and influence of this node in the cluster: the perpetuation of itself or some colluding nodes in this dominant position. To prevent that a clusterhead can create bogus nodes to falsify the topology of the network and be reelected clusterhead, the protocol requires that 50% of the stated neighbors of the clusterhead candidate participate in the issuance of its certificate using their identity private key. The use of a key certified by an external CA assures nodes have a real identity and avoids spoofing attacks. If the clusterhead candidate has a lot of neighbors that are not shared by other members of the cluster and, moreover, they do not participate in the clusterhead certificate issuance, the clusterhead is suspicious of manipulation. In these cases, plain cluster nodes can request identity evidence of clusterhead's neighboring nodes to authenticate its existence. If neighbors cannot prove the possession of an identity certificate issued by an external CA, the clusterhead candidate is put in the reputation list, its candidature is not supported, and a new clusterhead is elected.

3.4 Cluster Management Protocol

The Cluster Management Protocol is started once the clusterhead is set up. This protocol is used to manage the cluster formation and actualization. The clusterhead is responsible for issuing authorization certificates for good-behaving neighbors whose identity can be correctly verified. In Protocol 3 we summarize its phases.

Protocol 3

1. **Authorization Certificate Request** $[v_j \in cl_a \rightarrow h_i]$
Nodes request a certificate with a $CertReq = \{UID, ACID, wish\_NewKey\}$ ($ACID$ is attached in case the user has a former authorization certificate).
2. **Authentication Proof Request** $[h_i \rightarrow v_j \in cl_a]$
The clusterhead replays with an $AuthReq$ message that contains an $AuthMode$ parameter which indicates the type of authentication proof that is required (depending of the

confidence that the clusterhead has with the requesting node, $AuthMode$ would be one or another).

3. **Authentication Proof** $[v_j \in cl_a \rightarrow h_i]$
   The node sends an authentication proof, which is a digital signature or a Message Authentication Code (MAC) depending on the authentication mode.
4. **Authorization Certificate Issuance** $[h_i \rightarrow v_j \in cl_a]$
   The clusterhead validates received authentication proofs. If correct, it issues the certificates.

Assuming the clusterhead is honest, the issuance of authorization certificates is robust against malicious attacks. If the clusterhead is malicious, it could issue or deny authorization certificates uniquely based on its own criteria to maximize its profit. However, the clusterhead role only makes sense if there are a group of nodes that supports it and so, it is indirectly forced to follow the rules.

4 Adversarial Models

The clusterhead election process provides a fair, non-manipulable leader election. The accomplishment of the process is granted if it can be assured that data used in the utility function is certain and that any node cannot set up itself as a clusterhead nor avoid the role without being punished. We analyze the process using two adversarial models: a node that wants to get the leadership of the cluster and a selfish node that does not want to become the leader.

4.1 Greedy Adversarial Model

During the Cost Discovery Protocol, a malicious user v_i can try to alter the process to falsely report a greater number of neighbors than he really has, or on the contrary, fake the degree of his neighbors so that it is smaller. In the first case, v_i should include more user identifiers in the neighbors $List_i$ of his $DegAd$. However, this list has to be contrasted with other nodes' lists, and if reported connections are unidirectional they are discarded. Thus, nodes not only have to introduce more information to their own $DegAd$ messages, but they also manipulate the others. The ways to do it are:

- *Creating DegAd messages of inexistent nodes and inserting them in the network: DegAd* messages are signed. So, the only way to perform this attack is becoming a clusterhead and issue certificates for inexistent users. In any case, the identity of the nodes is frequently rechecked by succeeding clusterheads and cluster members.
- *Reusing DegAd messages of former neighbors:* The insertion of old $DegAd$ messages in the network to falsely prove that some node has bidirectional connections with other nodes is useless because $DegAd$s are protected against reply attacks.
- *Wormhole attack:* A colluding group of nodes can act together to perform this attack: one node pretends it has some one hop neighbors that it really does not have and there is a friend node that resends the packets in both senses.
  This attack can be detected by setting the network interface in promiscuous mode and monitoring the node's physical connections.

The second way to perform greedy attacks in the Cost Discovery Protocol is by decrementing the degree of cluster members except oneself:

- *Deleting users from others' DegAd messages*: It is not possible since $DegAd$ messages are signed. Modifications invalidate messages.
- *Dropping nodes out of the network*: The attacker can flood the network with fake $Hello$ messages in order to make nodes compose $DegAd$s with erroneous unidirectional entries and in the end, these nodes are considered malicious to others and isolated. However, the authenticity of $Hello$ can be checked afterwards in the protocol and links to invalid entries are discarded without being penalized.
- *Putting less nodes in one's degree list $List_i$ so that the neighboring information of these nodes cannot be checked up*: This method decreases both the degree of target nodes and oneself by the same amount; thus, its purpose is not accomplished.
- *Filtering DegAd messages*: A greedy node can avoid resending $DegAd$s to lower their chances of becoming clusterheads. However, other good-behaving neighbors would resend $DegAd$s that finally arrive to everyone.

Another type of greedy attacks is faking the Clusterhead Designation Protocol:

- *Sending fake IamCh messages with fictitious neighbors*: In order to be successful, more than 50% of stated clusterhead's neighbors should be corrupted.
- *Sending $Error$ messages against other clusterhead candidates*: It is not viable since error messages have to include the evidence that demonstrates that the clusterhead candidate does not accomplish the requirements.

4.2 Selfish Adversarial Model

The actions that selfish nodes could take to fake the Cost Discovery Protocol are:

- *Incrementing the list of neighbors of other nodes in DegAd messages*: It is not possible since $DegAd$ messages are signed and so, protected against modifications.
- *Composing DegAd messages with less neighbors than one really has*: When contrasting information from $DegAd$ messages, unidirectional links are discarded. However, if a node has more than 20% of these kind of links, this node is expelled from the network and cannot use its services any more. Thus, this option is unviable.
- *Presenting itself as a clustergateway*: A group of colluding nodes could create a fake subnetwork and a clustergateway, since clustergateways are not eligible to be clusterheads. This attack is very expensive and it is not worthy for a selfish node.

Another way to perform selfish attacks is in the Clusterhead Designation:

- *Not sending $IamCh$ message*: If a node is elected to be the clusterhead and it does not send the $IamCh$ message, it is expelled from the network.
- *Sending an $IamCh$ message with a list of neighbors bigger than it really is*: The objective of this attack is to impede 50% of the clusterhead neighbors support its candidacy. However, it is fruitless since all nodes know the degree of each of them.

5 Conclusions

Current cluster formation protocols have a vulnerability that has not been treated nor resolved: the information used to design the best suited topology, i.e. the utility function, is unverifiable. If we create clusters based on false information, neither the architecture would provide an optimal backbone nor the clusterhead be reliable to perform this role.

We have developed a secure and fair process of clusterhead election and cluster formation based on a specific utility function. We have divided the process in three phases and have fit cryptographic operations to each part in order to create an efficient and practical scheme. The security anchor is a public key infrastructure. However, most of the cryptographic operations involved in a cluster formation process are very efficient, based on symmetric cryptography and hash chains [10].

The proposed protocols assure that the selected clusterhead is the node with the highest degree in a neighborhood. This role is temporal to prevent nodes gathering too much sensitive information on the cluster members they control, and moreover, to avoid the exhaustion of these selected nodes. The process is fair in the sense that the elected node is the best one to be the clusterhead and no fakes can be made to become the clusterhead or avoid the role. Nodes that become clusterheads are securely appointed so that they can be recognized by any node of the network.

Acknowledgements This work is partially supported by the Spanish Ministry grants TSI-020100-2009-374 SAT2, TSI2007-65406-C03-03 E-AEGIS and CONSOLIDER CSD2007-00004 ARES.

References

1. Yu, J. Y., & Chong, P. H. J. (2005). A survey of clustering schemes for mobile ad hoc networks. *IEEE Communications Surveys and Tutorials, 7*(1), 32–48.
2. Huang, Y., & Lee, W. (2003). A cooperative intrusion detection system for ad hoc networks. In: *ACM Security of ad hoc sensor net. (SANS)* (pp. 135–147).
3. Wang, X., Guo, X., & Yin, X. (2006). Nrce: A new random cluster election algorithm. *Journal of Communication and Computer, 3*, 40–44.
4. Sirivianos, M., Westhoff, D., Armknecht, F., & Girao, J. (2007). Non-manipulable aggregator node election protocols for wireless sensor networks. In: *Modeling and optimization in mobile ad hoc wireless net.* (pp 1–10). WiOpt
5. Vasudevan, S., DeCleene, B., Immerman, N., Kurose, J., & Towsley, D. (2003). Secure leader election algorithms for wireless ad hoc networks. In: *IEEE DARPA Information Survivability Conf. and Exposition (DISCEX)*, (pp. 261–272).
6. Sun, K., Peng, P., Ning, P., & Wang, C. (2006). Secure distributed cluster formation in wireless sensor networks. In: *Annual computer security app. (ACSAC)* (pp. 131–140). Washington, DC, USA: IEEE Computer Society.
7. Xiao-yun, W., Li-zhen, Y., & Ke-fei, C. (2005). Sleach: Secure low-energy adaptive clustering hierarchy protocol for wireless sensor networks. *Wuhan University Journal of Natural Sciences, 10*(1), 127–131.
8. Lee, P., Kim, J., Han, I., Ryou, H., & Ahn, W. H. (2007). Secure cluster header election techniques in sensor network. In: *ISPA Workshops, LNCS* (Vol. 4743, pp. 20–31). Springer.
9. Lamport, L. (1981). Password authentication with insecure communication. *Communications of the ACM, 24*(11), 770–772.
10. Rifà-Pous, H., & Herrera-Joancomartí, J. (2009). Cryptographic energy costs are assumable in ad hoc networks. *IEICE Transactions on Information and Systems, E92.D*(5), 1194–1196.

Author Biographies

Helena Rifà-Pous is an associate professor in the Department of Computer Science at the Universitat Oberta de Catalunya from 2007. She received a graduate degree and a Ph.D. degree in Telecommunications Engineering from the Universitat Politècnica de Catalunya in 2001 and 2008, respectively. From 2000 to 2007, she was with Safelayer Secure Communications as a research project manager, focused on PKI projects mainly for the public administration. Her research interests include information hiding, network security, key management and mobile networks.

Jordi Herrera-Joancomartí is an associate professor in the Departament d'Enginyeria de la Informació i les Comunicacions at the Universitat Autònoma de Barcelona. He graduated in Mathematics from the Universitat Autònoma de Barcelona in 1994 and he received his Ph.D. degree in 2000 from the Universitat Politècnica de Catalunya. His research interests include topics in the field of computer security and more precisely in copyright protection techniques and security in ad-hoc networks. He has published more than 50 papers in national and international conferences and journals. He is also a reviewer of various national and international conferences and journals. He has been a main researcher in several national research projects.