Using New Tools for Certificate Repositories Generation in MANETs

Candelaria Hernández-Goya[1], Pino Caballero-Gil[1], Oscar Delgado-Mohatar[2], Jezabel Molina-Gil[1], and Cándido Caballero-Gil[1]

[1] Dpto. Estadística, I.O. y Computación. Universidad de La Laguna
38271 La Laguna, Tenerife, SPAIN
mchgoya@ull.es, pcaballe@ull.es, jezabelmiriam@gmail.com,
candido.caballero@gmail.com
[2] Instituto de Física Aplicada, Consejo Superior de Investigaciones Científicas
Serrano, 144, 28006, Madrid, Spain
oscar.delgado@iec.csic.es

Abstract. This paper includes a new proposal for the generation of certificates repositories in MANETs. The described process is based on the combination of the self-organized key management model together with the MultiPoint Relay (MPR) technique, generally used in the Optimized Link State Routing protocol. The main objective is to reduce the cost of generating and updating local certificate repositories by selecting those certificates that allow to reach the maximum number of nodes. This goal is just achieved by applying low-cost operations carried out locally by the nodes themselves.

Key words: Public-key management, MANETs

1 Introduction

Due to the lack of centralized infrastructure when dealing with Mobile Ad-hoc NETworks (MANETs) almost any task related to management or security services depend on network members themselves. It conveys that a significant amount of their restricted resources are spent on self-organization duties. Hence, special attention to nodes overload should be paid when designing new management mechanisms such as public-key management. In particular, the design of public-key management schemes may be catalogued as one of the hardest tasks when providing MANETs with security. The existent approaches to solve this problem are mainly based on one of two alternatives: distributed certification authorities or self-organized key management model.

In this paper the self-organized approach to public-key management is chosen as base in order to guarantee identical roles for all the network nodes. Therefore, nodes are in charge not only of creating, storing, distributing and revoking their public keys but also they should perform other classical management tasks such as packet routing.

The proposal here described tries to improve the performance of key management in the well-known web of trust model. In order to achieve this aim, we face the problem by combining typical authentication elements with common ideas used in routing protocols. In this way, we seek a reduction in resource consumption while undertaking the verification process associated to authentication.

The structure of this paper is as follows. Section 2 provides an insight into routing protocols used in MANETs, paying special attention to the Optimized Link State Routing (OLSR) protocol from which some ideas regarding the use of the MPR technique have been borrowed in order to improve key management tasks. Since our proposal is specifically designed to be deployed in the self-organized key management model, section 3 deals with the details of that approach. Both the Multipoint Relay technique described in Section 2 together with the graph-based public-key certification protocol described in Section 3 constitute the keystones of the proposal, which is described from an algorithmic point of view in this latter section. Finally, Section 4 closes the paper with the conclusions that may be extracted from this work and some questions that deserve further research.

2 Routing in MANETs: OLSR and MPR

In the last years one of the areas in MANETs where more research has been developed is that of routing algorithms [1]. Due to the lack of centralized infrastructure, routing in MANETs is one of the innumerable tasks that are in charge of the nodes themselves. This is just one of the reasons why cooperation among the members of the network is essential.

Some basic concepts referring to routing protocols used in MANETs and the information handled by them are here gathered in order to understand later how they are used to help in the authentication process.

There is a first basic classification used when talking of routing protocols that distinguish between proactive and reactive protocols. Protocols in the first category are characterized because each node should store a route for each reachable member of the network although such a path may not be required at that precise moment whilst regarding reactive protocols, it should be stressed that only when a request for communication between two nodes exists, a route discovery procedure is initiated. Due to this feature, these protocols are referred to as on-demand routing protocols.

Proactive algorithms are also known as table-driven routing protocols since local routing information defining the different paths is organized according a table stored by each node. The information contained in such a table defines an entry associated to each reachable node containing the next node in the path to the destination and a metric or distance, among another data. The metric can be defined in function of several criteria such as the hop distance, the total delay or the cost of sending messages. In networks with high mobility these routing protocols have a good behaviour since the paths are calculated in advance, and so the nodes do not have to wait until they are computed.

When comparing proactive and reactive protocols, it is important to point out that in the first set certain overload is originated in the network due to the continuous updates produced in routing information while those routing algorithms belonging to the second set have to face the delay due to the execution of routing discovery procedures produced any time a new path is defined.

In this work, we use certain elements of the proactive routing protocol known as Optimized Link State Routing (OLSR) protocol, which is one of the four basic protocols adopted for MANETs, in order to improve the construction of certificate repositories defined in the key management scheme when adopting the web of trust model.

In the OLSR proactive routing there are two stages clearly differentiated. Firstly, a reliable map of the network is built. In order to obtain such an accurate map, all the network nodes must exchange messages regarding the state of their connections links. In the second stage, and based on the map built, the optimum route among the nodes is generated. The main obstacle this protocol has to skip is the high number of messages to be exchanged among nodes. However, thanks to these messages the network configuration is known by all its members.

This routing algorithm has been extensively analyzed in the bibliography, and recently, some works devoted to improve it by integrating security tools [2] have been developed.

Thinking of reducing the overhead and message redundancy and trying to avoid the storm problem [3], a specific technique was defined in OLSR. In this technique each node selects a particular neighbour subset (nodes at one-hop distance) whose members will be in charge of broadcasting the information related to topology control. By doing so, the number of messages exchanged is considerably reduced, [4].

Roughly speaking, it can be said that this technique allows determining the minimum number of nodes needed for reaching the whole network when it is recursively applied. This procedure is named as the MultiPoint Relay (MPR) technique. The way we will utilise the basics of this technique in the key management proposal as well as its relationship with Graph Theory problems are included below.

The MPR technique was originally deployed for reducing the duplicity of messages at local level when broadcasting information in a proactive MANET [5]. In general, the number of redundant packets received by a node may be equal to the number of neighbours a node has. In the OLSR protocol only a subset of nodes will be in charge of retransmitting the received packets. In this way, every node u must define among its direct neighbours a set of transmitters (here denoted by $MPR(u)$) that will be the only ones in charge of retransmitting the messages emitted by the initial node.

According to this method, the choice of the set MPR should guarantee that all the nodes in a two-hop distance of the initial node receive the messages. In order to fulfil this requirement every node in a two-hop distance of u must have a neighbour belonging to $MPR(u)$.

In routing models the network is usually represented with a graph whose vertex set $V = \{u_1, u_2, \ldots, u_n\}$ symbolizes the set of nodes of the network. In this way, for any node u, $N^i(u)$ denotes the set of u's neighbours in a i-hop distance from u. Consequently, $N^1(u)$ stands for u's direct neighbours and $|N^1(u)|$ corresponds to vertex u degree. These sets are defined by using the shortest path and in such a way that $N^i(u)$ and $N^{i+1}(u)$ are disjoint sets.

Following the notation defined in [6] jointly with the one previously introduced in this paper it is feasible to formally define the MPR set for a vertex u as $MPR(u) \subseteq N^1(u) | \forall w \in N^2(u) \exists v \in MPR(u) | w \in N^1(v)$.

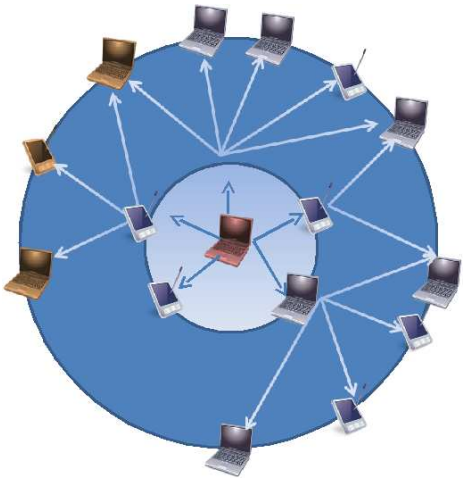
Through this definition, decision and optimization problems associated may be defined. According to the Computational Complexity hierarchy such problem belongs to the $NP - hard$ class. The heuristic defined in the OLSR routing procedure uses a greedy approach handling the vertex degree as parameter. The idea is to select the neighbours of the original vertex u which cover the highest number of vertices in u's two-hop vicinity that have not been previously covered.

In order to calculate $MPR(u)$ we need to define several vertex subsets that are specified below. First, for each node v in a one-hop distance from u it is required to consider a new vertex subset $W_u(v)$ formed by those vertices that simultaneously belong to the order 2 u's neighbourhood and are direct neighbours of v (see figure 1(a)). This set may be calculated by the following intersection $W_v(u) = N^2(u) \cap N^1(v)$. Vertices in this set have in common the fact that they are candidates to be covered by vertex v.

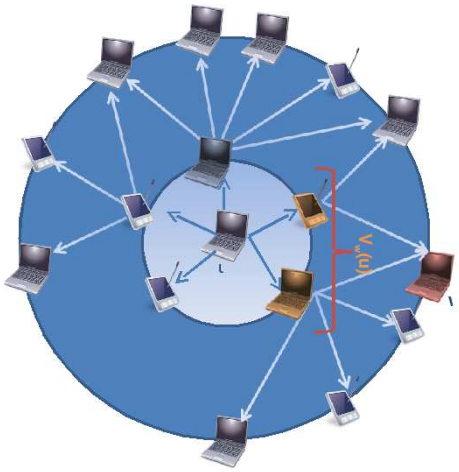
A second vertex subset is defined for each vertex w belonging to u's two-hop neighbourhood $V_w(u)$. In this case, such a subset may be obtained through the intersection $V_w(u) = N^1(w) \cap N^1(u)$ (figure 1(b)). This new set gathers those vertices in $N^1(u)$ that may cover vertex w. When transferring the MPR computation to the self-organized PKI model, $V_w(u)$ is calculated by using $N_1(w)$ this means the set of predecessors of vertex w.

Analyzing the description of the problem from the Graph Theory point of view, it can be concluded that the node being examined and its MPR set must form a dominant set in its level 2 vicinity. A dominant set in a graph is a vertex subset such that any node in the corresponding graph has and edge linking it to a vertex in the dominant set.

The greedy heuristic algorithm is composed by two main stages stages. In the first one those vertices w in $N^2(u)$ which have an only neighbour v in $N^1(u)$ are examined, in order two include in $MPR(u)$ the vertex v to which is connected. In case there are remaining nodes without covering in $N^(u)$, in the second stage, those vertices in $N^1(u)$ which cover more vertices in that situation are also included in $MPR(u)$. A graphic explanation of how the algorithm works is included in figure 2.



(a) $W_u(v)$.

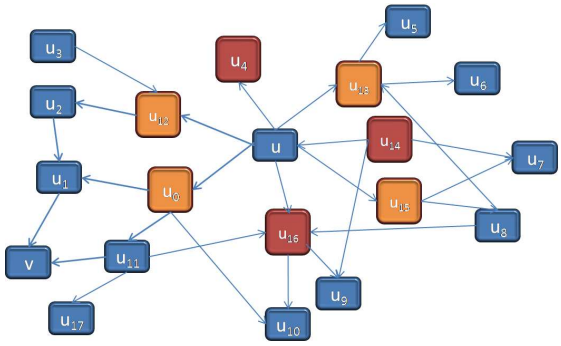


(b) $V_w(u)$.

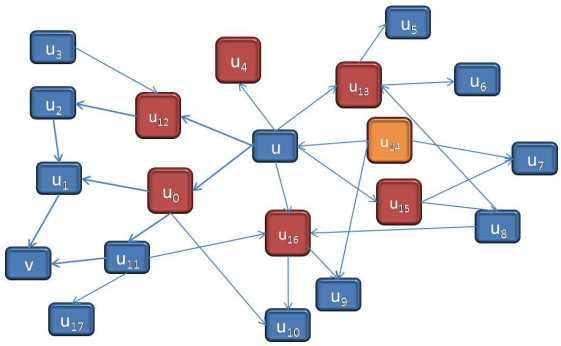
Fig. 1. Defining some vertex subsets

3 Key Management in MANETs: Self-Organized Model

In this section the main characteristics of the public-key management model that we use are described before introducing several new ideas that form our proposal.



(a) Stage 1: Isolated nodes in $N^2(u)$ are analyzed.



(b) Stage 2: Nodes of maximum degree are included in $MPR(u)$

Fig. 2. Stages in MPR-OLSR

In the bibliography we may find two main alternatives for the deployment of Public-Key Infrastructures (PKIs) in MANETs: distributed certification authorities, and self-organized public-key management model.

In the first case the certification procedure relays on distributed Certification Authorities (CAs) that thanks to a (t,n) threshold signature scheme issue and renew nodes's certificates [7], [8].

The main drawbacks of this model are the computational intensive operations required by the threshold application when signing a certificate, and the definition of additional procedures such as share refreshing [9]. Also, when dealing with certificate validation, nodes should locate a correct coalition but, depending on the actual network topology and conditions, it might result infeasible.

In this work we decided to follow the self-organized key management model based on the web of trust approach. Several are the reasons that justify the choice of this option. First, this model demands less maintenance overhead. Secondly, it is well worth remarking that on the one hand the self-organized approach eases the use of a simple bootstrap mechanism and on the other hand all the nodes perform equal roles.

The self-organized model in MANETs was initially described in [10]. Its authors put forward the substitution of the centralized certification authority by a self-organized scenario where certification is carried out through chains of certificates which are issued by the nodes themselves. Such a scheme is based on the information stored by each node and the trust relationship among neighbour nodes.

In this model public keys and certificates are represented as a directed graph $G = (V, A)$, known as *certificate graph*. Each vertex u in this graph defines a public key linked to a node, and each arc (u, v) symbolizes a certificate associated to v's public key, signed by using u's private key. Each node u has a public key, a private key, and two certificate repositories, the updated and the non-updated repositories. Initially the updated certificate repository will contain the list of certificates on which each node trust (out-bound list) and the list of certificates of all the nodes that trust on u (in-bound list).

A sequence $P_{uv} = \{(u, u_0), (u_0, u_1), ..., (u_m, v)\}$ of certificates where the vertices are all different is called a chain of certificates from u to v.

The tasks that a member of the network has to develop in this public-key management scheme are the following [10]:

1. Certificate Management:
   (a) Key generation: The node generates its keys by itself.
   (b) Certificate issue: The node issues certificates that bind public keys of other nodes to their identities.
   (c) Certificate exchange: The node exchange certificates with other nodes and builds its non-updated repository.
   (d) Updated certificate repository construction: The node builds its updated repository.
2. Public-Key Verification:
   (a) Finding a certificate chain.
   (b) Verifying the certificates in the chain.

Next we describe how certificate management and public-key verification are carried out in the self-organized model.

Each node u generates by its own the pair formed by its public key and its secret key. Then a request for signing the generated public key is sent to u's neighbours. Since these nodes are in a one-hop distance from u, they can use any trusted mechanisms such as side channels in order to assure the binding established between the corresponding public key and the node's identity.

Apart from that, in order to ease certificate revocation, each certificate issued will be valid for a certain period of time. This parameter may be chosen depending on the mobility characteristics of the underlying MANET.

Since the certificates issued by a node are stored in its local repository, one of the tasks that a node may perform during idle periods is the renewal of certificates issued by it to those nodes that might still be considered as trusted. Otherwise, certificate renewal may be developed on demand. It means that when an expired certificate is included in the non-updated repository of a node, such a node should request a renewal for that certificate.

When a certificate for a node u is issued by a node v the edge (v, u) is added to the certificate graph and each node u and v stores it in its in-bound and out-bound list, respectively.

Note that the speed in the creation of the certificate graph and its density depend on the motivation of the users for distributing certificates, and on nodes' mobility. In particular, the more mobility the nodes have, the more complete the repositories will be. The same happens with other aspects related to MANETs cooperation.

As in any PKI-based system, certificate revocation should also be taken into account. When revocation is initiated due to key compromise or misbehaviour of the corresponding node, the certificate issuer sends a message to all nodes stating that such a certificate has been revoked. This can be accomplished thanks to that each node maintains a list containing the members of the network that have contacted him to request updates of the certificates he has issued. Hence, in fact it is not necessary to send the revocation message to all the members of the network. The last proposals related to revocation policies in MANETs defend the creation of schemes based in reputation systems [11], [12]. When revocation is due to the fact that the expiration time has been reached, such a revocation can be deduced directly by all nodes since the expiration date is contained in the certificate.

Certificate exchange is a low-cost procedure because it only involves one-hope distance nodes. It allows to share and to distribute the issued and stored certificates. A description of this procedure is as follows:

1. Every node u retransmits the hash values of the certificates stored in the repositories G_u and G_u^N to its neighbours. The recipient nodes answer with the hash values of the certificates contained in its repositories.
2. Every node contrasts the received value with the one he already has and requests to its neighbours only the certificates that are new.
3. If the local memory of a node is not enough, the expired certificates are deleted from the non-updated repository, starting by the oldest ones.
4. In this way, after a short period of time the non-updated repository G_u^N contains almost all the certificate graph G. Afterwards, the only task to be carried out by the nodes is to exchange the new certificates.

In the original proposal two ways of building the updated certificate repository G_u of a node u were described:

1. Node u communicates with its neighbours in the certificate graph.
2. Node u applies over G_u^N an appropriate algorithm in order to generate G_u after checking the validity of every single certificate.

The selection of the certificates stored by each node in its repository should be done carefully in order to satisfy at the same time two requirements: limitation in storing requirements, and usefulness of the repository in terms of ability to find chains for the largest possible number of nodes. This problem, known as optimal dispersal of certificate chains, has lately received particular attention in the bibliography [13].

The algorithm used in the construction of the updated repositories will influence in the efficiency of the scheme, so it should be carefully designed. The simplest algorithm for that construction is the so-called Maximum Degree Algorithm (MDA) (see algorithm 3.1), where the criterion followed in the selection of certificates is the degree of the vertices in the certificate graph.

Algorithm 3.1: MDA-$G_{out}(G, u, l_{out}, c)$

Step 0: $V_{out} = \emptyset, A_{out} = \emptyset, Dout = \emptyset, i \leftarrow 1$
Step 1: $e_{out} = min\{deg_{out}(u), c\}$
Step 2: $l \leftarrow deg_{out}(u)$
Step 3: $N^1(u) = S_{out}(N^1(u)) = \{v_1, v_2, \ldots, v_l\}$
Step 4: $D_{out} = \{v_1, v_2, \ldots, v_{e_{out}}\}$
Step 5: $V_{out} = V_{out} \cup \{u\} \cup D_{out}, A_{out} = A_{out} \cup \{(u, v_i)\}, \ i = 1, 2, \ldots, e_{out}$
Step 6: $i \leftarrow 1, l_i \leftarrow 1$
Step 7: **while** $i < e_{out}$ **and** D_{out} **not** $= \emptyset$
do
- **Step 7.1:** **if** $l_i = l_{out}$
  - **then Step 7.1.1:** $i \leftarrow i + 1$
  - **else**
    - **Step 7.2.1:** $v_i = get(D_{out})$
    - **Step 7.2.2:** $N^1(v_i) = S_{out}(N^1(v_i))$
    - **Step 7.2.3:** $w_i = get(N^1(v_i))$
    - **Step 7.2.4:** **while** $w_i \in D_{out}$ **and** $N^1(v_i)$ **not** $= \emptyset$
      - **do Step 7.2.4.1:** $w_i = get(N^1(v_i))$
    - **Step 7.2.5:** **if** $N^1(v_i) = \emptyset$
      - **then Step 7.2.5.1:** $i \leftarrow i + 1$
      - **else**
        - **Step 7.2.6.1:** **if** $w_i \notin D_{out}$
          - **then**
            - **Step 7.2.6.1.1:** $put(w_i, D_{out})$
            - **Step 7.2.6.1.2:** $A_{out} = A_{out} \cup \{(v_i, w_i)\}$
            - **Step 7.2.6.1.3:** $V_{out} = V_{out} \cup \{w_i\}$
            - **Step 7.2.6.1.4:** $l_i = l_i + 1$
            - **Step 7.2.6.1.5:** $i \leftarrow i + 1$
- **Step 7.3:** **if** $i \ mod \ e_{out} = 0$
  - **then Step 7.3.1:** $i \leftarrow 0$

There is a more sophisticated algorithm, called Shortcut Hunter Algorithm, in which certificates are chosen taking into account that when they are deleted, the length of the minimum path between the nodes connected through that certificate is increased in more than two units.

When using the MDA, every node u builds two subgraphs, the out-bound subgraph and the in-bound subgraph, which when joined generate the updated certificate repository G_u. The out-bound subgraph is formed by several disjoint paths with the same origin vertex u while in the in-bound subgraph u is the final vertex. In the description of the algorithm that follows, the starting node is u and $deg_{out}(u)$, $deg_{in}(u)$ stands for the in-degree and the out-degree respectively of node u. The number of paths to be found is represented by c.

A bound on the number of disjoint paths starting at u, as well as a bound on the number of disjoint paths to be built with u as final node are given by e_{out} and e_{in}, respectively.

Another important input parameter is s, which represents the maximum number of vertices to be included in the subgraph generated when the in-bound and the out-bound subgraphs are combined. This parameter may be also controlled by defining as $l_{out} = \lceil s/(2e_{out}) \rceil$ the length of the chains generated when building the out-bound subgraph and $l_{in} = \lceil s/(2e_{in}) \rceil$ for the in-bound one.

In order to apply the greedy criterion, $S_{out}(N)$ and $S_{in}(N)$, where N consists of a set of vertices, include the sorted vertices of N into descending order according to $deg_{out}(u)$ and $deg_{in}(u)$, respectively.

Note that the process to build the in-bound subgraph is equivalent to it except for the fact that in this case the edges to be chosen are always incoming edges.

In the first stage of the MDA, $deg_{out}(u)$ outgoing arcs from u are included. The final vertices of these arcs are then included in D_{out}. This set is implemented as a typical queue where the insertion (put) and the extraction (get)operations are used. Henceforth, e_{out} arcs are chosen in such a way that the formed paths are disjoint. This is accomplished by selecting their origin belonging to D_{out} and checking that neither the origin nor the final vertices were previously used in another path.

The main contribution introduced in this paper consists in substituting the MDA proposed for the updated repository construction by a new algorithm that uses the MPR technique described in 2. In this way, for each vertex in the certificate graph we have to define a re-transmitter set. Hence, the smallest number of vertices required for reaching the whole certificate graph will be obtained.

The MPR heuristic adapted to the certificate graph is described below (algorithm 3.2). First, node u starts by calculating $MPR(u) = \{v_1, v_2, \ldots, k\}$. Then, these vertices are included in G_{out} together with the edges $(u, v_i), i = 1, 2, \ldots, k$. Henceforth, nodes v_i in $MPR(u)$ apply recursively the same procedure of re-transmitting backwards the result $MPR(v_i)$.

In order to extend the notation used in the introduction of the MPR greedy heuristic described in section 2, which is required to be used in the certificate

graph, we denote by $N_i(u)$ the set of predecessors of node u that may be found in an i-hop distance.

Algorithm 3.2: MPR-$G_{out}(G, u)$

Step 0: $Initialization : MPR(u) = \emptyset$
Step 1: $N^1(u) = \{v_1, v_2, \ldots, v_l\}$
Step 2: $N^2(u)$
comment: First stage
Step 3: for $i \leftarrow 1$ **to** l
do
- **Step 3.1:** $N^1(v_i)$
- **Step 3.2:** $W_{v_i}(u) = N^1(v_i) \cap N^2(u) = \{w_1, w_2, \ldots, w_k\}$
- **Step 3.3: if** $k \neq 0$
  - **then**
    - **Step 3.3.1: for** $j = 1$ **to** k
      - **do**
        - **Step 3.3.1.1:** $N_1(w_j)$
        - **Step 3.3.1.2:** $V_{w_j}(u) = N^1(u) \cap N_1(w_j)$
        - **Step 3.3.1.3: if** $|V_{w_j}(u)| = 1$
          - **then**
            - **Step 3.3.1.3.1:** $MPR(u) = MPR(u) \cup \{v_i\}$
            - **Step 3.3.1.3.2:** $N^2(u) = N^2(u) \setminus W_v(u)$

Step 4: $N(u) = N(u) \setminus MPR(u)$
Step 5: $l = l - |MPR(u)|$
comment: Second stage
Step 6: while $N^2(u) \neq \emptyset$
do
- **Step 6.1: for** $i = 1$ **to** l
  - **do**
    - **Step 6.1.1:** $N(v_i)$
    - **Step 6.1.2:** $W_{v_i}(u) = N^1(v_i) \cap N^2(u)$
    - **Step 6.1.3:** $d_u^+(v_i) = |W_{v_i}(u)|$
- **Step 6.2:** $d_{max}(u) = \max d_u^+(v_i), i = 1, 2, \ldots, l$
- **Step 6.3: for** $i = 1$ **to** l
  - **do Step 6.3.1: if** $d_u^+(v_i) = d_{max}(u)$
    - **then**
      - **Step 6.3.1.1:** $MPR(u) = MPR(u) \cup \{v_i\}$
      - **Step 6.3.1.2:** $N^2(u) = N^2(u) \setminus W_{v_i}(u)$
- **Step 6.4:** $N^1(u) = N^1(u) \setminus MPR(u)$

Step 7: $MPR(u) = \{v_1, v_2, \ldots, v_k\}$
Step 8: for $i = 1$ **to** k
do
- **Step 8.1: if** $v_i \notin V_{out}$
  - **Step 8.1.1:** $V_{out} = V_{out} \cup \{v_i\}$
  - **Step 8.1.2:** $A_{out} = A_{out} \cup \{(u, v_i)\}$
  - $MPR - G_{out}\{G, v_i\}$

This means that the smallest number of certificate chains required in order to reach the remaining nodes will be obtained as well. The algorithm proposed is an iterative scheme that may be described in the following way:

1. Every vertex $u \in G$ locally determines its re-transmitter set ($MPR(u)$), which include the certificates associated to the corresponding edges.
2. This vertex contacts all the nodes in $MPR(u)$. At this stage, every node $v \in MPR(u)$ has previously obtained its re-transmitters set $MPR(u)$, and consequently it may send to node u the certificates associated to such a set.

Since each node knows from whom is a re-transmitter, the G_{in} subgraph is generated by applying first the reverse process and then adding in-going arcs.

The certificate chains required in the authentication are built by using the arcs $(u, MPR(u))$. After that, $\forall v \in MPR(u)$ and $\forall w \in MPR(v)$ the arcs (v, w) are also added after having checked that they have not been added in previous updates.

Note that the procedure every node $u \in G$ has to develop in order to build $MPR(u)$ takes $1 + ln(N^2(u))$ steps when no bound is defined on the length of the chains to be built. Otherwise, the number of iterations to be carried out is given by the number of hops to explore in the certificate graph. As for the definition of the aforementioned bound, it has to be remarked that such a parameter may be dynamically adjusted in function of the changes experienced by the certificate graph. This may be justified by the fact that as the network evolves, the information contained in each node's repository is more complete.

Thanks to this substitution the generated procedure is easier and more efficient, guaranteeing in this way that each node has a set of neighbours that allows it to reach the biggest number of public keys.

Although this work is in its inital stage, a first implementation has been developed. It has been carried out using Java and the open source library JUNG 2.0 [14] (Java Universal Network/Graph Framework) which provides the basic tools for representing and dealing with graphs. Apart from this, JUNG allows generating random graphs with the small-world property, which is fulfilled by certificate graphs [15]. When a graph holds this characteristic, most nodes may be reached by a small number of hops from any source node. This kind of graphs has received special attention in several scientific disciplines including MANETs. The particular small-world model used in the simulation developed was proposed by Kleingberg [16]. When generating a graph with n^2 vertices according to this model, the first step is to create an nxn toroidal lattice. Then each node u is conected to four local neighbours, and in addition one long range connection to some node v, where v is chosen randomly, according to a probability proportional to $d^{-\alpha}$. d denotes the lattice distance between u and v and α stands for the clustering coefficient. This coefficient α is defined as the average over all the nodes in the network of the relationship between the maximum number of edges that may be deifned and the number of edges that actually exist. Generating the graphs following this model guarantees that the shortest paths may be determined using local information, what makes them particularly interesting for the networks we are dealing with.

Some of the data gathered from the computational experience are shown below (table 1). The number of nodes in the graph (n), the rate of certificates contained in the repository (R_c), the clustering coefficient (α), the maximum

length in the chains generated (C_l) and the time consumption while the execution (t) expressed in seconds are the marameters that have been measured. From this experience, it may be remarked that the certificate rate finally contained in the local repository increases as the size of the graph increases as well as the clustering coefficient increases. This phenomenon may be better apreciated in figure 3. Additionally, the maximum length in the chains obtained are kept at reasonable values, what makes the chain verification process lighter. Finally, the rate of certificates stored in the repository surpasses 95% in more than 75% of the executinos while time consumption corresponds to sensible values. These first experiments shown promising results.

	$\alpha = 0.1$			$\alpha = 0.4$			$\alpha = 0.8$		
n	R_c	C_l	t	R_c	C_l	t	R_c	C_l	t
9	42.93	4	0.24	37.03	3	0.27	37.78	3	0.18
16	82.08	3	0.49	86.67	3	0.41	84.17	3	0.46
25	93.13	3	0.64	96.00	3	0.59	96.00	3	0.69
36	98.70	3	0.81	99.63	3	0.83	99.44	3	0.8
49	99.73	4	1.24	99.18	4	1.2	99.59	4	0.92
64	99.59	3	0.68	100.00	4	0.68	99.48	3	0.64
81	99.92	4	0.77	99.92	4	0.81	99.82	4	0.84
100	99.93	4	0.91	99.93	4	0.97	99.80	4	0.96

Table 1. Computational Experience

One of the main advantages of the proposal is that all the information gathered for the construction of the chains is locally obtained by each node.

After obtaining the in-bound and out-bound subgraphs, both subgraphs are merged and the initial repository is generated so that the authentication process may start.

When a node u needs to check the validity of the public key of another node v, it has to find a certificate chain P_{uv} from itself to v in the graph that results from combining its own repository with v's repository.

If this chain is not found there, the search is extended to $G_u \cup G_u^N$, what implies the inclusion of u's non-updated repository in the search. If this second exploration is successful, u should request the update of those certificates that belong exclusively to G_u^N. When no path is found, the authentication fails.

Once the path P_{uv} is determined, u should validate every certificate included in it. This is done as follows:

1. The first certificate in the chain (u, u_0) is directly checked by u since it was signed by u himself.
2. Each one of the remaining certificates (u_i, u_{i+1}) in the chain may be checked using the public key of the previous node u_{i-1}.
3. The last arc (u_m, v) corresponds to the certificate issued by u_m that binds v with its public key.

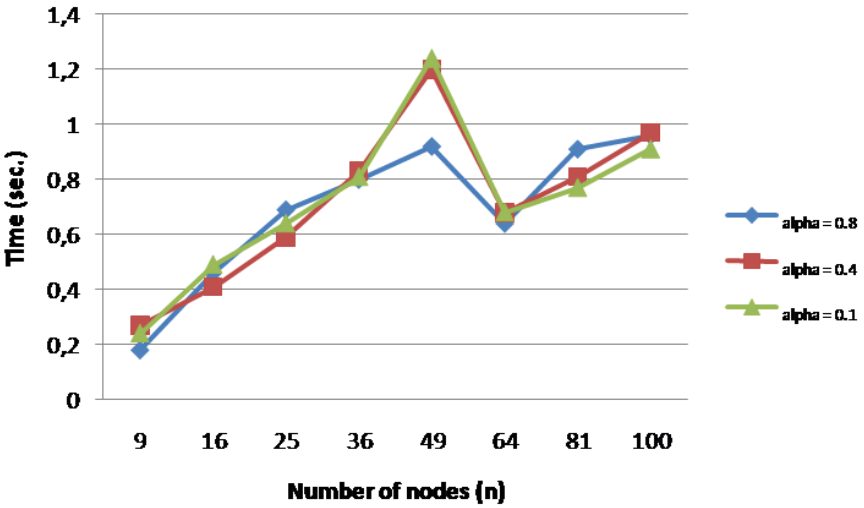


Fig. 3. Time consumption.

4 Conclusions and Further Research

This work proposes the application of the Multipoint Relay Technique in the computation of certificate repositories included in the self-organized public-key management model proposed by [10]. Our proposal is supported by the good results obtained when using the MPR procedure in the $OLSR$ routing algorithm in MANETs and a prelimiar computational experience. Through this improvement we provide the public-key management scheme with simplicity and efficiency.

The computational implementation of the proposal is part of a work in progress. Consequently, a future version of this paper will include an extended simulation experiment.

References

1. Royer, E., Toh, C.K.: A review of current routing protocols for ad-hoc mobile wireless networks. IEEE Personal Communications Magazine **6**(2) (1999) 46–55
2. Vilela, J.P., Barros, J.: A feedback reputation mechanism to secure the optimized link state routing protocol. In: IEEE Communications Society/CreateNet International Conference on Security and Privacy for Emerging Areas in Communication Networks (Securecomm'07), IEEE Computer Society (2007)
3. Ni, S.Y., Tseng, Y.C., Chen, Y.S., Sheu, J.P.: The broadcast storm problem in a mobile ad hoc network. In: MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking, New York, NY, USA, ACM (1999) 151–162

4. Laouti, A., Mhlethaler, P., Najid, A.., Plakoo, E.: Simulation results of the OLSR routing protocol for wireless network. In: 1st Mediterranean Ad-Hoc Networks workshop (Med-Hoc-Net). Sardegna, Italy 2002. (2002)
5. Clausen, T., Jacquet, P.: RFC 3626: Optimized Link State Routing Protocol (OLSR) (2003)
6. Mans, B., Shrestha, N.: Performance evaluation of approximation algorithms for multipoint relay selection. In: The Third Annual Mediterranean Ad Hoc Networking Workshop. (2004)
7. Wu, B., Wu, J., Fernandez, E.B., Ilyas, M., Magliveras, S.: Secure and efficient key management in mobile ad hoc networks. J. Netw. Comput. Appl. **30**(3) (2007) 937–954
8. Saxena, N., Tsudik, G., Yi, J.H.: Threshold cryptography in P2P and MANETs: The case of access control. Comput. Networks **51**(12) (2007) 3632–3649
9. Narasimha, M.., Tsudik, G., Yi, J.: On the utility of distributed cryptography in P2P and MANETs: The case of membership control. In: Proceedings of the 11th IEEE International Conference on Network Protocols (ICNP'03), IEEE (2003) 336–345
10. Capkun, S., Buttyan, L., Hubaux, J.P.: Self-organized public key management for mobile ad hoc networks. Mobile Computting and Communication Review **6**(4) (2002)
11. Arboit, G., Crepeau, C., Davis, C.R., Maheswaran, M.: A localized certificate revocation scheme for mobile ad hoc networks. Ad Hoc Networks doi:10.1016/j.adhoc.2006.07.003 (2006)
12. Moore, T., Clulow, J., Nagaraja, S.., Anderson, R.: New strategies for revocation in ad-hoc networks. In: 4th European Workshop on Security and Privacy in Ad-hoc and Sensor Networks, ESAS 2007
13. Jung, E., Elmallah, E.S., Gouda, M.G.: Optimal dispersal of certificate chains. IEEE Transactions on Parallel and Distributed Systems **18**(4) (2007) 474–484
14. : JUNG 2.0, (Java Universal Network/Graph framework)
15. Capkun, S., Buttyan, L., Hubaux, J.P.: Small worlds in security systems: an analysis of the PGP certificate graph. In: Proceedings of The ACM New Security Paradigms Workshop 2002, Norfolk, Virginia Beach, USA (September 2002) 8
16. Kleinberg, J.: The small-world phenomenon: An algorithmic perspective. In: Proceedings of the 32nd ACM Symposium on Theory of Computing. (2000)