On AAA framework in opportunistic ad hoc network: OLSR usecase

Willy Jimenez, Hakima Chaouchi

CNRS SAMOVAR Lab/UMR 5157
TELECOM & Management SudParis, 9 rue Charles Fourier, 91011 EVRY, FRANCE
{Willy.Jimenez, Hakima.chaouchi}@it-sudparis.eu}

Abstract- In order to simply identify participating nodes in an ad hoc network, authentication is one good option to bring some trust in such dynamic and infrastructure-less network. Ad hoc network technology offers cheap and flexible network coverage extension. However it needs AAA support in order to allow a service provider to offer services over these ad hoc networks. The AAA should be distributed so that ad hoc nodes can benefit from such service even if there is no constant connectivity to the access network. [1]. This paper deals precisely with the AAA issue in an ad hoc network, it presents our architecture to handle AAA services in an ad hoc network with opportunistic connectivity to the infrastructure. First we introduce the general idea of distributing the AAA service on what we call "*special ad hoc nodes"* which will be preconfigured by a network operator, then a description of a possible design of this special ad hoc device is provided. We also propose AOLSR, an extension to OLSR protocol to provide routing services only to authenticated nodes.

Keywords: ad hoc networks, AAA, OLSR, authentication, trust, virtualized OS.

1 Introduction

In the context of Always On era, ad hoc technologies integration with the infrastructure is without any doubt a clever way to extend at low cost the network access coverage. However, a real and business oriented service deployment over ad hoc network requires firstly identifying the communicating nodes, securing the communications, and resource accounting. We believe that the integration of ad hoc and infrastructure-based technologies coupled with efficient security and accounting techniques is the answer for the urgent demand of network operators for appropriate architectures to host secure and large scale ubiquitous services.

This work is supported by Telecom Sud Paris, and ANR SARAH project 2006-2010

There are several security threats in ad hoc networks. First, those related to wireless data transmission such as eavesdropping, denial of service in message replaying, message distortion and active impersonation. Second, those related to ad hoc construction of the network. This means that attacks can come also from inside the mobile ad hoc network (MANET). Therefore we cannot trust one centralized node to support for instance the AAA service, because if this node would be compromised the whole network would be useless. Another problem is scalability. Ad hoc networks can have hundreds or even thousands of mobile nodes. This introduces important challenges to security mechanisms [2].

As most of the security issues in ad hoc networks are caused by trust less nodes. Authentication process is a strong solution to eliminate those misbehaving nodes or at least identify them. Nevertheless, ensuring authentication service in a self organized network is not easy to realize. We propose in this work to build a distributed AAA service in ad hoc network where the AAA service which is classically centralized in the infrastructure network is somehow distributed or designed hierarchically. These services will be securely distributed in the servicing ad hoc nodes. We also propose an authentication based OLSR routing named AOLSR to ensure the forwarding service only to authenticated nodes.

The remainder of the paper is organized as follows. First the distributed AAA architecture is presented in section 2, followed by the description of AOLSR protocol. Finally we present our approach analysis and we conclude this paper.

2 The proposed distributed AAA architecture

A number of research work was conducted on the classically centralized AAA functions, but very few studied the possible interactions between AAA and ad hoc networks since AAA service is classically centralized and ad hoc network is distributed and uncontrolled by its nature. Thus, the introduction of AAA into ad hoc environment is not an easy task due to the self organising aspect of the ad hoc network. However, by ensuring; as it is proposed in our paper, a secure delegation of the AAA service over some special ad hoc node, it is possible to consider extending the AAA service in the ad hoc network. This architecture is depicted on the Figure 1 below. It targets deploying several mechanisms such as authentication, authorization, accounting, key management, and other network services such as Neighbour and Service discovery mechanisms that are also necessary to provide information for the ad hoc node in order to allow him to get the appropriate service.

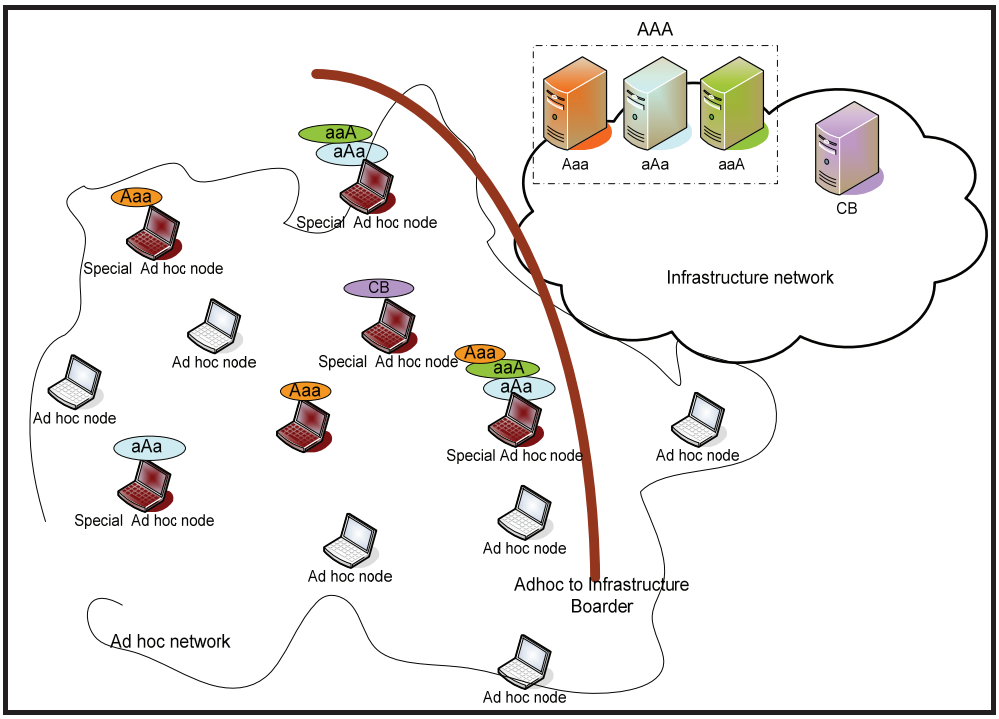

Figure 1: Delegated AAA service in a MANET [1]

To answer the question: How to distribute the AAA service over the ad hoc network? We propose two complementary design choices:

- the special ad hoc node design
- and the authentication based routing protocol design; the AOLSR.

2.1 Special Ad hoc node design

What we mean by special ad hoc node is a node capable to offer in a secured way the delegated network service from the operator in our case; the AAA service. The idea is to have either a dedicated ad hoc machine pre-configured by the operator to serve as a special ad hoc node for AAA service in the ad hoc network area; it might be a sort of robot-router moving in certain area. It can also be a machine belonging to trusted entity, or it can be a special running environment in a user's terminal. This special environment has to be isolated from the user's running environment. For this purpose, we decided to use virtualization techniques to create a special virtual machine in the ad hoc node that will run the delegated network services such as AAA service. We assume that in the future, virtualization will be less battery and resource consuming.

In this section we consider from a theoretical point of view, the use of the virtualization technologies in order to have a solution to deploy secure services in MANETs by using special ad hoc nodes. These nodes will run at least two environments, one for authentication controlled by the operator and one for the user. Virtualization with isolation concept will securely separate the two environments

embedded in the called trusted special ad hoc node. We studied different virtualization systems such as XEN, VMware and the security issues[3,4,5,6].

As stated earlier, the main idea is to have a distributed authentication service and bring it as close as possible to the users in a MANET. To do so, an operator should configure several nodes that will be part of the MANET; these nodes will have two different environments using virtualization software. The number of these special nodes will depend on the network size and the desired coverage service.

In the native environment of the node, the operator installs the AAA server; aka RADIUS server with a database of the customers that could use the services offered by the operator itself or a third party. The service offered to the customers is loaded in a virtual machine. Under these conditions, there should be a trust relation between the operator and the service provider if it is a third party, since sensitive data will be stored in the host operating system and the service provider will have physical access to the node.

Additionally it is important to consider the security issues regarding virtual machines implementation. However, the advantage of having virtual environment for the service offered is the possibility of migrating the service to another virtual machine in the case the service is down or is attacked by hackers reducing the down times and increasing service availability.

We designed our special ad hoc node by using open source software; implemented as follow:

- Prepare several laptops with Linux as host operating system.
- Install virtualization technology.
- Install “freeradius” server, and customers’ database. For security database content should be encrypted.
- Configure RADIUS server and policies.
- Create virtual machines for the service provider according to the required operating system.
- Prepare the booting sequence of the laptops to leave them ready to work with all the services activated.

Next step should be signing a confidentiality agreement with the service provider and deliver the special ad hoc nodes to users who would like to provide the connectivity via the ad hoc service to other customers. The incoming customer will buy the service and use their accounts with the telecom operator to access it.

In order to have an idea about the authentication time, we built a scenario with three nodes, one of them with a virtualization application, as shown in the following picture:

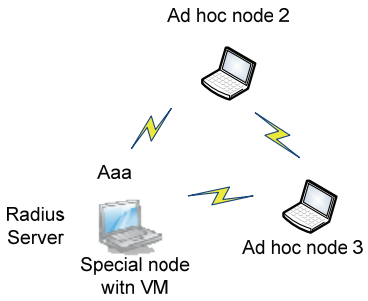

Figure 2: MANET with special ad hoc node for Measurements

All the nodes are using Linux as operating system; the virtual machine is Windows XP and was created using Xen and VirtualBox. The equipment used for the test is listed in the following Table 1:

Special node with VM	Ad hoc node 2	Ad doc node 3
HP xw4600 Core2 E6850 @ 3GHz 4GB RAM WPN111 Wireless USB Adapter 250GB HDD Fedora Core 8 + ndiswrapper + freeradius	Dell Latitude D410 Pentium M @ 2GHz 2GB RAM Intel PRO Wireless 2915 802.11 a/b/g Wi-Fi 40GB HDD Fedora Core 8 + radtest	Dell Inspiron 6400 Core2 T7200 @ 2GHz 2GB RAM Intel PRO Wireless 3945 802.11 a/b/g Wi-Fi 160GB HDD Fedora Core 8 + radtest

Table 1: Equipment list

The special node in figure 2 has freeradius server and the virtual environment; the nodes are using the radtest command to perform the authentication and the time is measured with Wireshark tool (traffic capture tool).

Two scenarios are considered per virtualization application, in the first one, the VM is ON but with no activity, in the second the VM is busy with a file transfer with the opposite node, it means, if we measure the time for node 2, the window VM is transferring a file with node 3 and vice versa. The transfer was done using Filezilla application.

VM type in Special Node	Node 2	Node 3
VirtualBox VM	2,881	1,150
VirtualBox VM busy	10,312	10,742
Xen VM	3,245	1,880
Xen VM busy	10,656	27,810

Table 2: Authentication Times expressed in mili seconds

From Table 2, we can observe how the authentication time increases when the VM is busy, which is reasonable since the kernel of the special node has more work to do, forward packets to/from Windows VM to one of the nodes; additionally the occupation of the air interface also increases. Then, it is important to evaluate all these factors in order to dimension the topology of the network and guarantee a target performance. Also it is important to study the power consumption when using virtual systems on an ad hoc node.

2.2 Authentication based ad hoc routing design- AOLSR

Based on the fact that special ad hoc nodes are available in the ad hoc network, we just need to ensure secure authentication to maintain access control in the ad hoc network. For this purpose, we propose to extend secure OLSR (sOLSR) [8] to include authentication.

sOLSR is a solution which adds a security mechanism to the OLSR protocol behavior [9]. sOLSR uses a cryptographic shared key to sign all the packets in order to ensure the integrity of OLSR control traffic data; only the nodes that share the key can participate in the routing domain; meaning that messages without verifiable signatures are discarded.

Additionally, to prevent replay attacks, secure OLSR uses timestamps, so the nodes exchange their timestamps before allowing the flow of any traffic between them. In this exchange process, three new messages types are introduced in OLSR:

- Challenge message
- Challenge-response message
- Response-response message

However, the exchange occurs between neighbors that have not registered timestamps of each other and where the traffic cannot be validated by the signature check. It means these messages are signed internally, and they carry their own digest/signature and they are never stacked with other OLSR-messages but rather sent in OLSR-packets of their own.

sOLSR description

The Timestamp Exchange Process

When A receives a signed message from a neighbor B, for which A has no registered time value, A initiates the timestamp exchange process. It means A sends a challenge message. The message is broadcasted since A might not have a route to B. The challenge message contains a 32-bit nonce. A then signs this message with a digest of the entire message and the shared key.

B has to respond to this message with a challenge-response message. B first generates the digest of its IP address, the received nonce and the shared key. B then generates a 32-bit nonce and transmits the nonce, the timestamp of B, the digest and a digest of the entire message and the shared key.

When A receives the challenge-response message from B, it first tries to validate the data. If the digest can be validated, then the timestamp of B is used to create the difference of time between A and B. A then generates a response-response message and broadcasts it to B. The message contains the timestamp of A, a digest of A's address, the nonce received from B and the shared key and a digest of the entire message and the key.

Finally, when B receives the response-response message from A, it tries to verify the digests. If they can be verified, B uses the received timestamp to register its time difference to A and the process is now completed.

Secure OLSR Implementation

The secure version of OLSR is implemented as a plugin of the olsrd daemon. It captures all the incoming traffic, verifies the packets and removes the signature message and updates the size field of the OLSR packet header. The plugin also intercepts all outgoing OLSR traffic to add the signature messages and updates the packets size as shown in the Figure 3 below [7]:

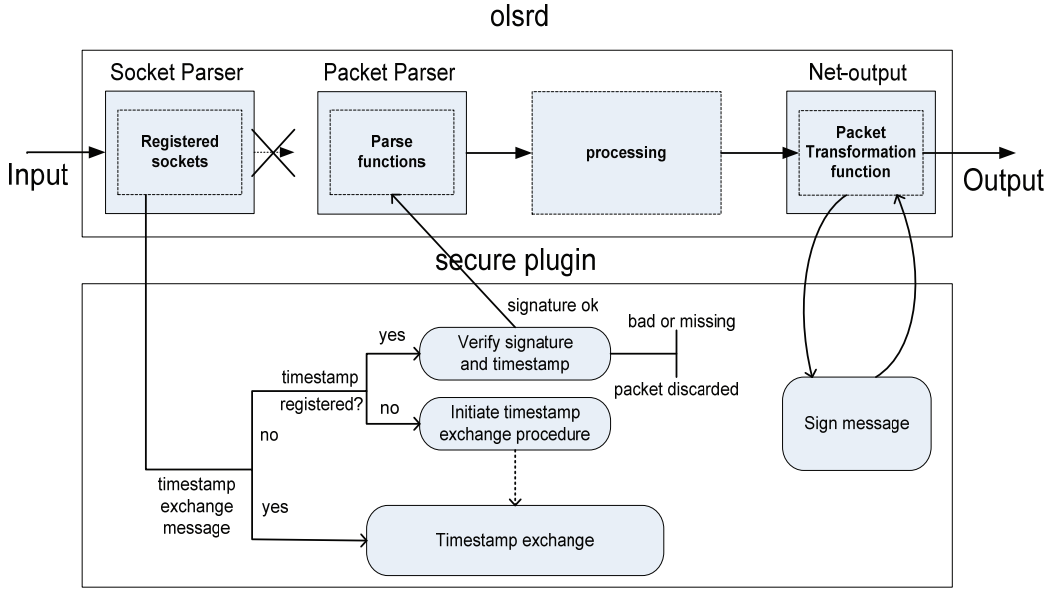

Figure 3: sOLSR design [7]

2.3 Proposed AOLSR description

Continuing with the purpose of having distributed authentication in MANETs now we propose the protocol part. In this case we want to authenticate all the nodes that will participate in the MANET routing domain, if authentication fails the node is not added and the ad hoc routing is blocked; it's the access control.

It means, there will be one special node inside the network which also runs a RADIUS server; the special ad hoc node, which will restrict the access to the routing domain through nodes authentication. The solution is implemented as a plugin of the OLSR daemon, and it is done modifying the existing secure plugin of olsrd as explained later.

Authenticated Optimized Link State Routing protocol or AOLSR is a new implementation of OLSR protocol where nodes are required to be authenticated by an AAA server as a previous requirement to participate in the routing domain. It is noted that if AOLSR is activated, OLSR and sOLSR packets are dropped by AOLSR nodes.

This new plugin has been developed by us based on secure OLSR plugin and then it supposes there is an existing key shared among all nodes. The idea of this new version is to add an authentication step as a mechanism to avoid that an attacker that compromised the key could join the routing domain. It means, in the case a node gets the shared key; it won't join the network unless it is authenticated by the radius server.

AOLSR uses the same process of secure OLSR, and it just adds some extra conditions in the behavior, as can be seen in the next Figure 4.

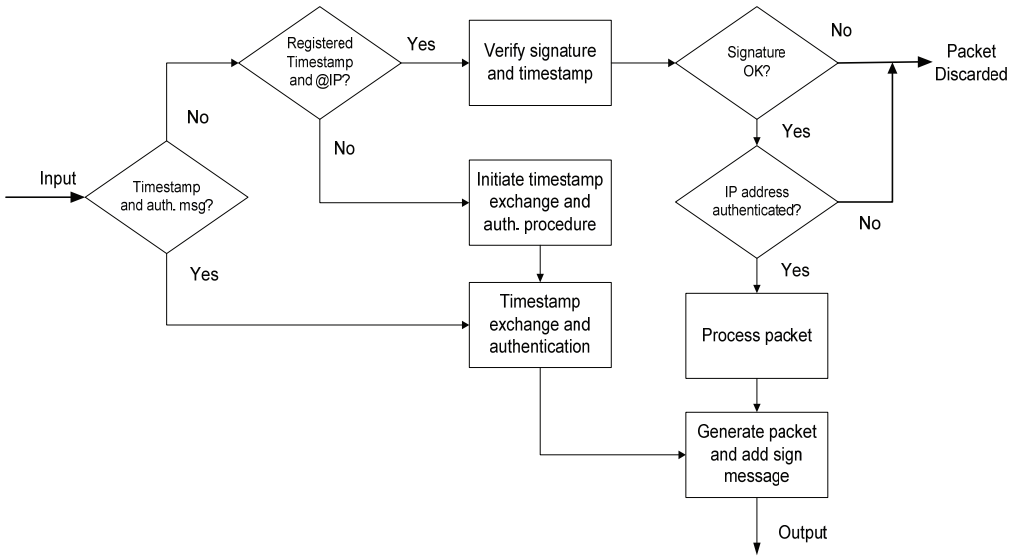

Figure 4: AOLSR - Flow Diagram

In our implementation we consider:

- Three nodes: A, B and C are considered in a two hop topology.
- A is a special node that has a radius server for authentication purposes.
- The IP address of the node is used as user name for authentication purposes.

- The radius server is implemented using freeradius server.
- The radius client request is done using the command “radtest”.

Now, we will describe how the AOLSR protocol functions considering two steps. At the beginning node A is available and node B is the first node coming to join the network. Once A and B are in the same MANET then C will join them and will be required also authentication.

First Node Authentication Scenario

The situation for this case is shown in the Figure 4. A is the main node of the network and is running AOLSR and the radius server.

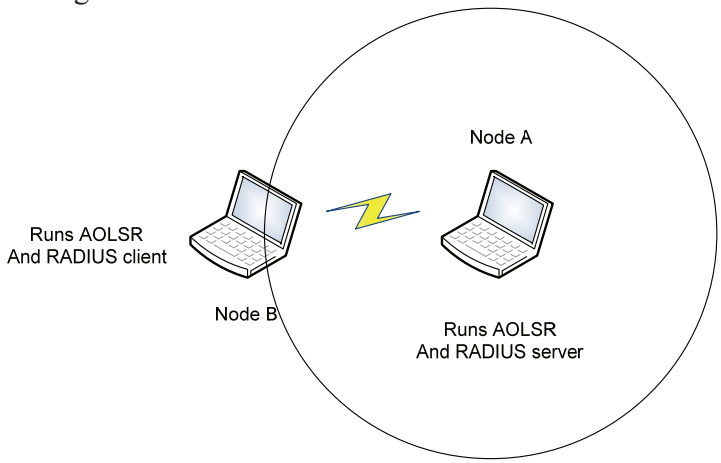

Figure 5: AOLSR - First node authentication scenario

Then B comes running AOLSR and wants to join A. When A receives any signed message from the new neighbor B, for which A has no registered time value, A initiates the timestamp exchange process. A sends the challenge message. The message is broadcasted since A might not have a route to B.

B has to respond to this message with a challenge-response message, but before that B authenticates with the radius server in A.

When A receives the challenge-response message from B, it first tries to validate the data. If the digests can be validated, then the timestamp of B is used to create the difference of time between A and B. Additionally, A checks the radius log file to verify the successful authentication of B. If it is good, A adds B to its authorized IP address list. Later A generates a response-response message and broadcasts it to B. The message contains the timestamp of A.

Finally, when B receives the response-response message from A, B uses the received timestamp to register its time difference to A and also B adds A in the trusted IP address list.

When A received another message from B, A already has B in the timestamp database, and then A will check also the authenticated IP address list, where B will be if the authentication process was successful. If not, then any packet coming from B is going to be rejected.

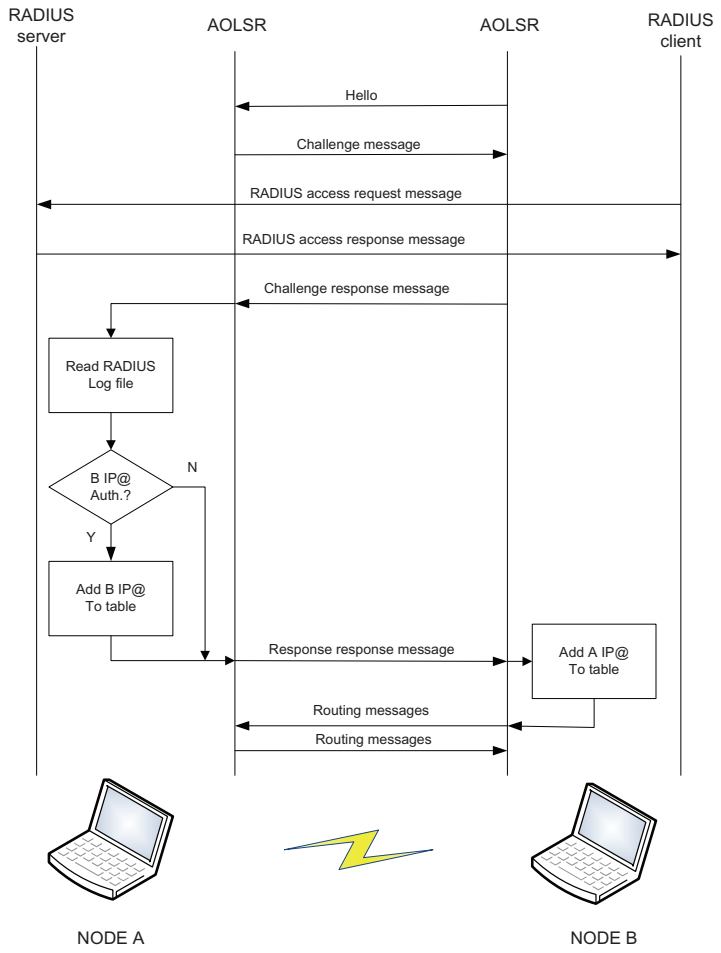

Figure 6: First node authentication - Logical Operation

Second Node Authentication Scenario

In this scenario a node C is the new node that wants to join the network formed by nodes A and B. The location of node C is next to B but cannot communicate directly to node A as shown in the next figure.

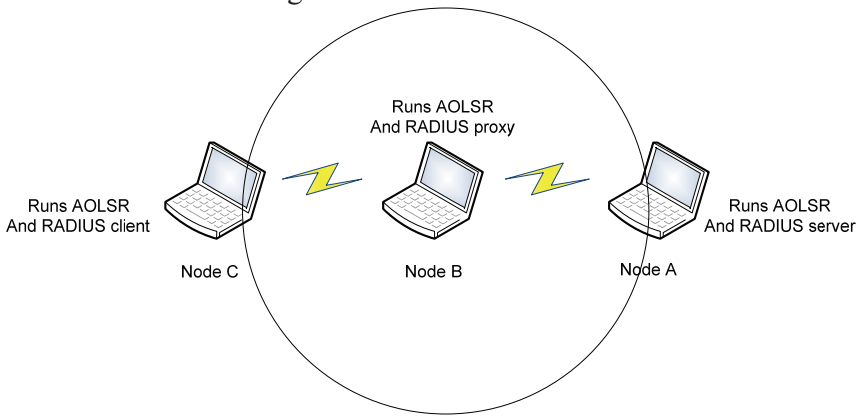

Figure 7: AOLSR - Second node authentication scenario

Since C cannot communicate with A directly, we assume that B is running a radius proxy also which will be used for C authentication.

When it arrives, node C sends a signed message to B, then a process similar to the one described above is initiated.

Node B has no registered time value for node C, thus B initiates the timestamp exchange process sending the challenge message.

When C receives the challenge message, it performs the authentication with the radius proxy running in node B, which forwards the request to the RADIUS server running in node A. After that C sends B a challenge-response message.

When B receives the challenge-response message from C, the timestamp of C is used to create the difference of time between them. Additionally, B checks the radius log file to verify the successful authentication of C. If it is good, B adds C to its authorized IP address list. After, node C generates a response-response message for B which contains the timestamp of C.

Finally, when C receives the response-response message from B, C uses the received timestamp to register its time difference with B and also registers B in the trusted IP address list.

When B received another message from C, B already has A in the timestamp database, and then B will check also the authenticated IP address list, where C will be if the authentication process was successful. If not, then any packet coming from C is going to be rejected.

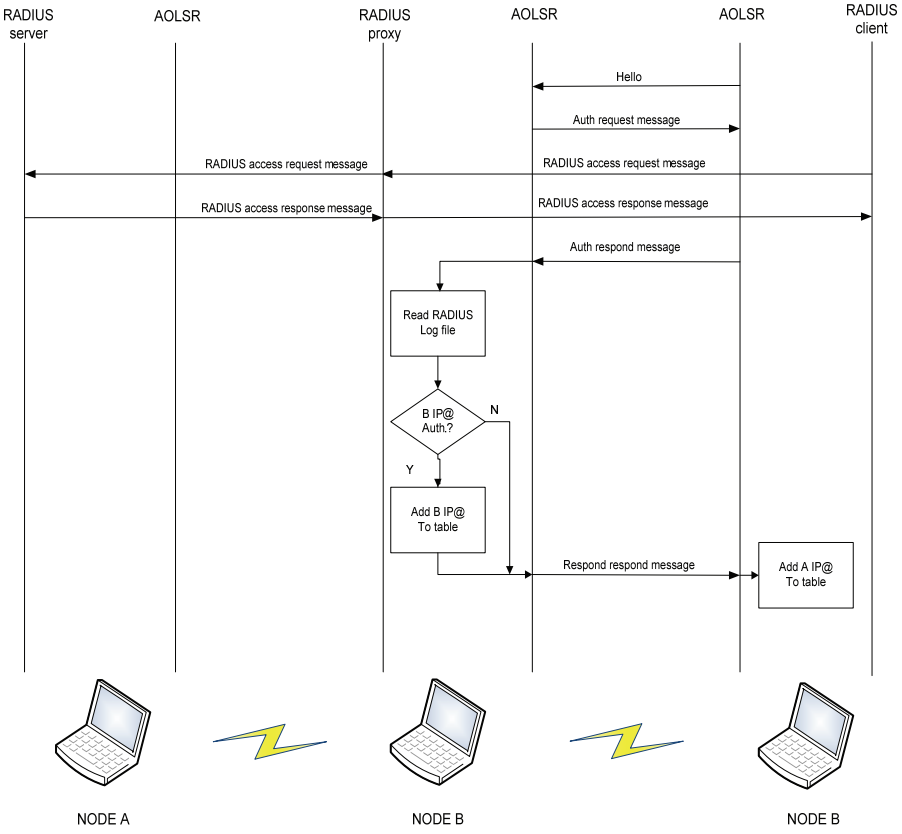

Figure 8: Second node authentication - Logical Operation

AOLSR Practical Implementation

For our practical implementation we add some new functions to the secure plugin code, especially to module olsrd_secure.c. We kept the main functions of the plugin because as explained before it intercepts all the incoming traffic and checks if they are timestamp exchange messages, if yes then the plugin processes the packet according to the packet type, if not it just removes the signature and sends it to the general packet parser function; and this behavior is helpful for our authentication process.

Then, new functions were added to create, add and consult a list of known IP address, and it is done after a successful login of the incoming new node. The authentication is verified reading the RADIUS log file and verifying the authentication of the new node's IP address. Additionally new lines were added to force the look up in the registered IP address list before routing.

The authentication process was embedded in the timestamp exchange process, it means when a new node arrives, it will initiate the timestamp exchange process and will be authenticated at the same time. Now, it was necessary to add a flag to differentiate the node with the RADIUS server and node with the proxy RADIUS in order to avoid that an incoming node tries to authenticate the server. For the RADIUS server and proxy we use freeradius server, and for the client the tool radtest.

3 Conclusions

We proposed two possible complementary solutions to implement the distributed authentication service in a MANET: using virtualization to build special ad hoc node that securely offers delegated network services such as distributed AAA service in MANET, and securing the routing protocol with our proposed AOLSR. We proposed to use virtualization to build special nodes in the network that provides the Authentication service on an ad hoc node in a secured manner by isolating the user environment from the AAA service embedded in the user's ad hoc node. While in the second solution; AOLSR, the routing protocol is modified to perform authentication as previous step of joining the MANET routing domain. This is to allow only authenticated nodes to benefit from the routing protocol.

From a performance point of view of the special ad hoc node, we got some authentication time measurements that show better response in the case of VirtualBox, however to conclude that VirtualBox is really better it is necessary to perform a more extensive test, we just wanted to have a reference for the authentication times. Before using virtualization it is important also to know the possible vulnerabilities of the package to use in order to implement basic security mechanism that can ensure the isolation and protection of the host environment and guest machines.

In the case of securing the routing protocol; AOLSR, we worked with an active routing protocol called OLSR. We used an open source daemon implementation (www.olsr.org). This daemon already has a secure plugin which uses a shared key to sign the packets and uses timestamps to avoid replay attacks and establish neighbor trust relationships. Regarding the key, it assumes that it is already available for the nodes that join the network. In our solution we modified the secure plugin of OLSR

daemon to add an authentication phase and we called it Authenticated OLSR (AOLSR), thus we assume also the availability of a shared key while the authentication process could protect the network from attacker that were able to get the shared key. It means, there is an initial node that acts as Authentication server and runs AOLSR which will enable the formation of a MANET, then any node that wants to join the network has to run AOLSR and has to be authenticated by the server, if succeeds it will be part of the network. For practical implementation we added new functions in the secure plugin to have authentication, in fact we keep the timestamp procedure and we added access control conditions for authentication and a new table to keep the known and authenticated IP address of the neighbors. We tested the implementation among three nodes and it worked, however a deep evaluation is required to test the performance of this solution. Of course, a larger ad hoc network is better to experiment our proposal.

4 References

[1] Chaouchi H. and Laurent-Maknavicius M. Annex 1 of French patent: Intégration de la technologie ad hoc dans la chaine de valeur des télécommunications, *INPI n° 0756559, 2007*.

[2] H Yang, H Y. Luo, F Ye, S W. Lu, and L Zhang, "Security in mobile ad hoc networks: Challenges and solutions" (2004). IEEE Wireless Communications. 11 (1), pp. 38-47. Postprint available free at: http://repositories.cdlib.org/postprints/618

[3] Antonopoulos A. Securing Virtualized Infrastructure: From Static Security to Virtual Shields. Senior Vice-President & Founding Partner, Nemertes Research. http://hackreport.net/wp-content/uploads/2007/03/nemertes-issue-paper-securing-virtualized-infrastructure.pdf

[4] Ormandy T. An Empirical Study into the Security Exposure to Host of Hostile Virtualized Environments. http://taviso.decsystem.org/virtsec.pdf

[5] Ferrie P. Attacks on Virtual Machine Emulators. Symantech Advanced Threat Research
http://www.symantec.com/avcenter/reference/Virtual_Machine_Threats.pdf

[6] King S., Chen P., Wan Y., Verbowski C., Wang H. and Lorch J. SubVirt: Implementing malware with virtual machines. In /Proceedings of the 2006 IEEE Symposium on Security and Privacy/ (May 21 - 24, 2006). IEEE Computer Society, Washington, DC, 314-327.

[7] Tonnesen A. Master Thesis: Implementing and extending the Optimized Link State Routing Protocol. University of Oslo. 2004. http://www.olsr.org/docs/report.pdf

[8] T. Clausen, “Securing OLSR problem statement”, Internet draft, 2005

[9] http://www.olsr.org/