Protocols for Distributed AAA Framework in Mobile Ad-hoc Networks

Sondes LARAFA and Maryline LAURENT-MAKNAVICIUS

CNRS Samovar UMR 5157, TELECOM & Management SudParis, 9 rue Charles Fourier, 91011 EVRY, FRANCE
sondes.larafa@it-sudparis.eu Maryline.Maknavicius@it-sudparis.eu

Abstract. Ad-hoc networks are subject to malicious attacks given their wireless nature and high dynamicity. Various security issues have been raised so far, especially access control. In a previous work, we focused on a light and distributed AAA framework using a six message-AAA protocol, and an access token mechanism. In this article, we optimize this AAA protocol through the definition of two new AVPs. Besides to resist to spoofing and replay attacks, we design a two-way protocol that allows destination nodes to check the access authorization validity of their neighbors.

Keywords: Ad-hoc networks, distributed AAA infrastructure, access phase, access token, SEND

1 Introduction

Ad-hoc networks are wireless networks able to self-configure with no administrator's assistance. They are known as infrastructure-less networks, i.e. with no central network entity for supporting packet routing. Ad-hoc networks might be very dynamic. A mobile node joins an ad-hoc network simply by connecting to the nearest already connected nodes. Once a mobile node is connected, it has three functions: transmitting and receiving data, in addition to routing.

Ad-hoc networks are very useful for supporting military and rescue operations because they are simple to set up and remain operational as long as there are enough nodes to relay traffic. They are likely to play an important role in the future networks by extending the operators networks coverage. This would enable other users to get access even if they can not directly reach the networks. A crucial prerequisite for this, however, is the availability of suitable authentication, authorization and charging mechanisms to ensure revenues for operators [1].

AAA (Authentication, Authorization and Accounting) infrastructures provide these functions. They especially ensure security by applying access control. In the wireless, dynamic and infrastructure-less context of ad-hoc networks, we are concerned by a distributed AAA infrastructure. We gave a detailed description of its design in [2] where we proposed a six-way protocol for mutual

authentication, and an access token to be used during the access phase after authentication and authorization phase.

In the second section of this document we give an overview of our distributed AAA infrastructure compared to centralized AAA infrastructures. Then in the third section we review our six-way authentication protocol for which we give an optimization through the definition of two new AVPs. The fourth section deals with the access token usage to secure the access phase. We demonstrate that it is vulnerable to spoofing and replay attacks if we just rely on SEND protocol for its validation process. That is why we introduce a two-way protocol to be executed between each newly authenticated node and any other selected destination node during the access phase.

2 Distributed AAA infrastructure for ad-hoc networks

2.1 Existing Centralized AAA infrastructure

AAA infrastructures are classically used by network operators and service providers to control the access to their networks and services, and also to perform accounting operations for next charging their subscribers. These ones are first authenticated then granted access to certain resources.

Figure 1 shows a AAA infrastructure composed typically of a Client Node (CN - the subscriber), the AAA server performing the authentication, and an access point which relays the access requests from the CN to the AAA server. The access point is also an Enforcement Point (EP) [1] which filters the traffic of non authenticated CNs.

A classical AAA infrastructure refers to the two following protocols:

- An access protocol, like PANA, that supports the communications between the CN and the access point for the authentication and authorization. As such, the CN integrates a PANA client and the access point a PANA authenticator.
- A AAA protocol that supports AAA exchanges between the AAA client within the access point and the AAA server.

PANA and AAA messages carry the same EAP (Extensible Authentication Protocol) messages [3]. Depending on the authentication method used, a fixed number of EAP messages transport authentication and authorization material for session establishment between the CN and the access point, thus between the CN and the network.

Because lightness of ad-hoc networks solutions is very important, we conceived a distributed AAA infrastructure that employs only AAA protocols (cf. section 2.2).

AAA exchanges cover three phases:

[1] The EP is responsible for enforcing policies with respect to authentication of subscribers, authorization to access and services, accounting and mobility, etc

– Authentication and authorization phase: EAP messages carry the authentication and authorization elements. An EAP Success message is sent to the CN if the authentication was successful and a session is established between the CN and the access point.
– Access phase: CN sends traffic through the access point. The EP does not filter this traffic.
– Accounting phase: the access point sends CN's consumption information to the AAA server.

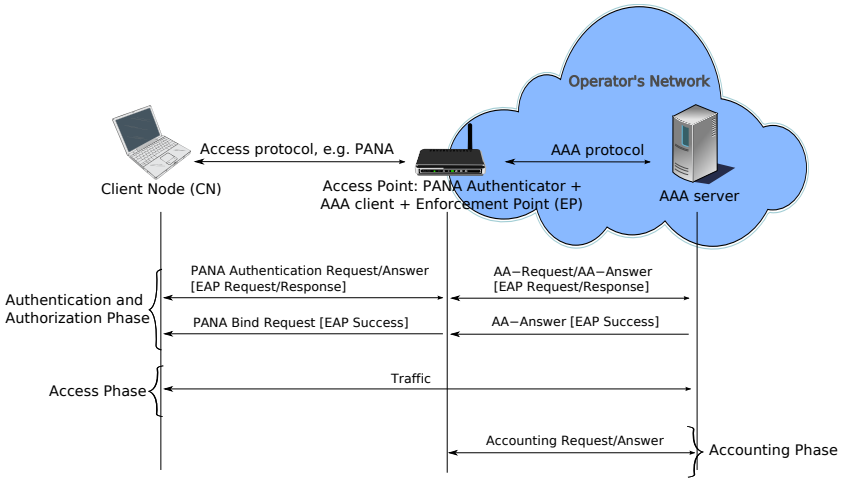

Fig. 1. Centralized AAA Infrastructure

2.2 Distributed AAA Infrastructure as defined in [2]

Introduction of AAA functionalities into ad-hoc networks is needed, as underlined in the introduction (cf. section 1). However it is inadequate to exploit centralized AAA frameworks given the decentralized nature of ad-hoc networks. That is why we designed a distributed AAA infrastructure [2] that has the following features.

n ($n \geq 1$) AAA servers (AAA_1, AAA_2,..., AAA_n), rather than one centralized AAA server, ensure the AAA service. Each CN is a AAA client so it does not need to contact an access point to join the AAA servers. This saves the exchanges between the access point and the CNs. Moreover when a CN (i.e. a AAA client) has been correctly authenticated, it acquire the functions of an Enforcement Point. This means that it becomes responsible for examining the arriving traffic and for filtering it if the generator node was not authorized.

During authentication and authorization phase, a CN contacts at least t ($1 \leq t \leq n$) AAA servers in order to be authenticated correctly. t is the threshold

number required by [2] based on the principle of Shamir Secret Sharing [4]. In [2], we adapted the ISO [9798-3] three-way authentication protocol [5] to our distributed AAA service thanks to the principle of Shoup's signature shares [6]. The resulted authentication protocol makes use of public key certificates that can be initialized into CNs by a third party (e.g. a Service Provider).

Mutual authentication occurs between a CN and the AAA servers. The CN authenticates itself first in order to avoid overloading the servers with heavy ciphering and deciphering operations during the first exchange. This improves resistance to denial of service attacks.

At the end of the authentication and authorization phase, AAA servers send an access token to the CN. This token is necessary during the access phase because it proves that the CN was successfully authenticated by the AAA service and authorized to use the ad-hoc network.

Since at least t AAA servers are required for CN's authentication, and because our authentication protocol consists of six exchanges, at least $6 \cdot t$ messages are necessary to achieve the authentication of one CN. It is obvious that the number of authentication messages increases faster than t given the coefficient 6. Optimization of the number of the protocol exchanges is then a necessity.

3 Distributed AAA Protocol Optimization

3.1 Distributed AAA Protocol as defined in [2]

Figure 2 depicts our six way-protocol as we defined it in [2]. The AAA service consists of n servers ($n \geq 3$), and the threshold number t is equal to 3. A joining node (JN - a CN) wishes to join the network. It obtains the list of the AAA servers available as explained in [2]. Then it initiates AAA exchanges with 3 of them e.g. AAA_1, AAA_2, and AAA_3. It sends its identity in the first exchange. The AAA servers reply in the second exchange with a random number R_{AAA} chosen jointly by them.

The third and fourth exchanges are inspired from the ISO three way-protocol since we distributed it using Shoup's principle. In the third exchange, JN signs with its private key the random R_{AAA} together with some data, namely a random number R_{JN} that it generates and the AAA service identity ID_{AAA}. AAA servers examine the validity of this signature. Then each server generates a piece of the AAA service signature and sends it in the fourth exchange. JN combines these pieces of signature according to Shoup's principle [6]. A signature computed by an entity A and sent to an entity B authenticates A towards B if B establishes this signature integrity. So if JN's signature and AAA service's signature are valid, the mutual authentication between JN and the AAA service is achieved successfully.

AAA servers reply if they consider that the JN authentication was successful. Similarly JN triggers the fifth exchange if it succeeds to authenticate the AAA service. The sixth exchange carries the access token of the JN: AAA servers agree on a deadline T_{JN} after which JN will have no longer access to the network and

will have to re-authenticate itself. Each server concatenates T_{JN} with the IP address of the JN and signs the result following Shoup's principle (like it does in the fourth exchange). The resulted piece of signature is concatenated to the deadline T_{JN} (cf. Fig. 2). Please have a look on the section 4 for further details about the access token and its use during the access phase.

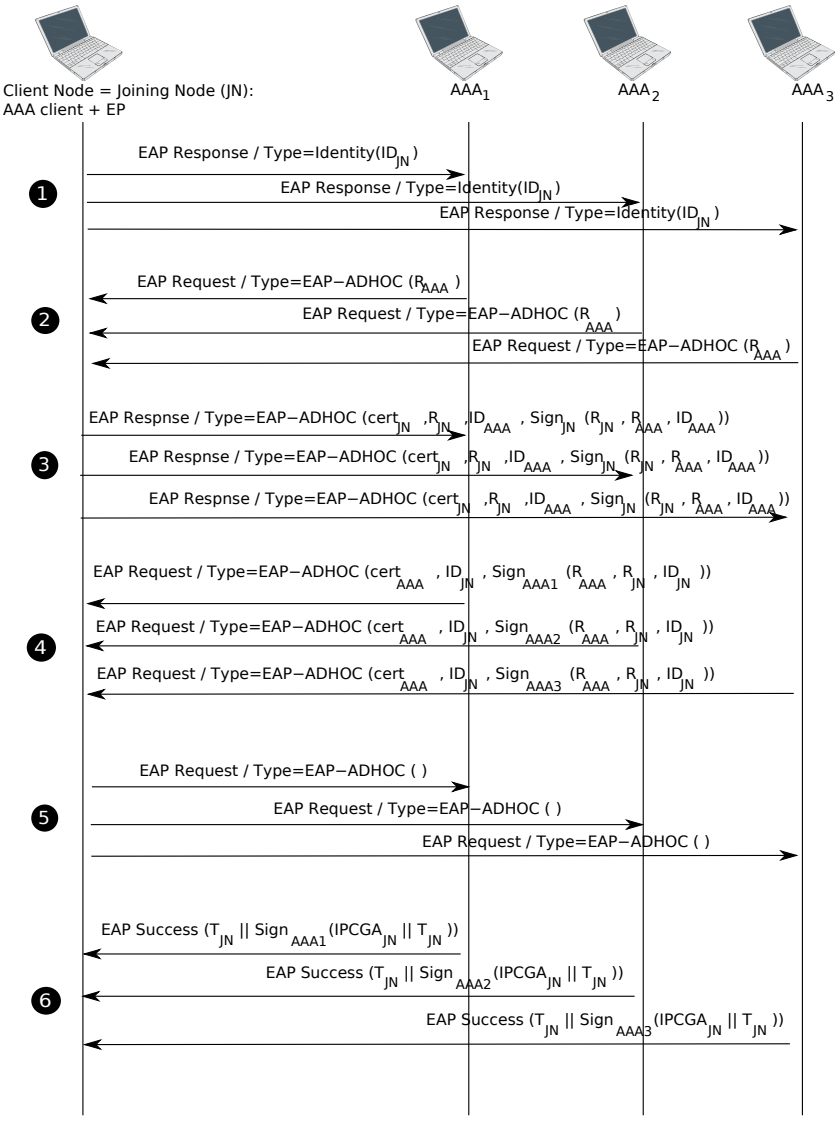

Fig. 2. AAA protocol in distributed AAA infrastructures [2]

We named EAP-ADHOC the distributed authentication method that we designed based on ISO [9798-3] norm.

3.2 Optimization

To optimize the protocol described in this section, the three last exchanges are replaced by one exchange (cf. Fig. 3). The fifth exchange is in fact an empty message where the JN informs the AAA service that the mutual authentication was successful. It induces the access token sending by the AAA service. Actually the success of JN's authentication is a sufficient reason for AAA servers to send the access token. Then it is up to the JN to trust or not the information sent by the AAA service.

Now in the optimization, the last message is an EAP Success which informs the JN that its authentication was successful. It carries the authentication information of the AAA service and the access token of the JN. Both are encapsulated into two new AVPs (Attribute Value Pair): AAA Authentication AVP encapsulates the authentication information and the Access Token AVP encapsulates the access token (cf. section 3.3.).

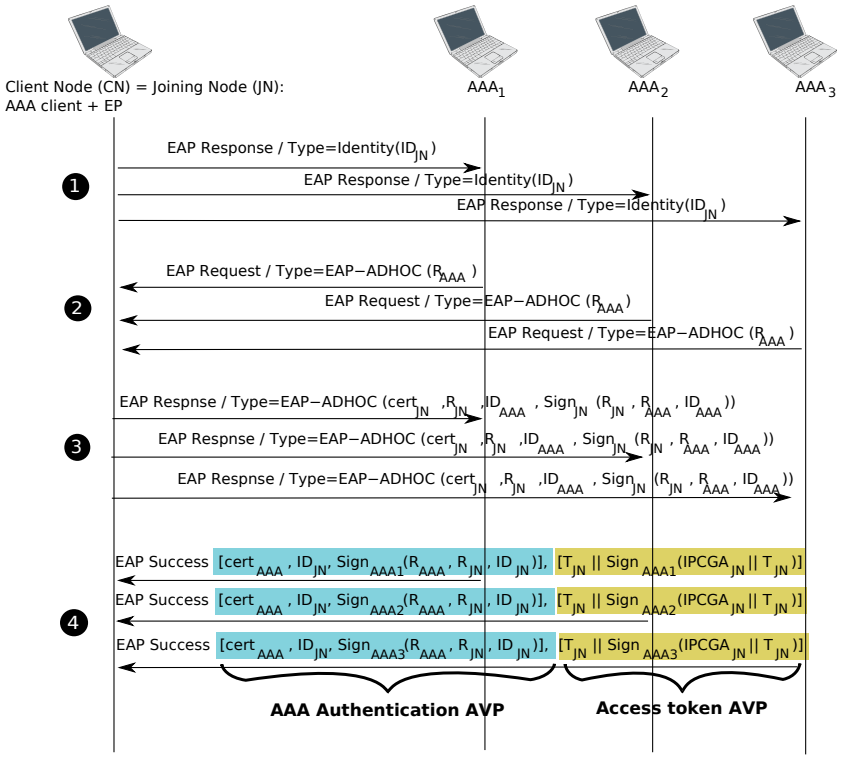

Fig. 3. Optimized AAA protocol in distributed AAA infrastructures

3.3 New AVPs

AVPs are tuples < *attribute name*, *value* > that carry specific authentication, accounting, authorization, routing and security information as well as configuration details for the AAA exchanges. For example all EAP messages for JN's authentication shown in the figure 3 are transported in EAP-Payload AVPs. However there is no AVP expected to transport AAA service authentication information at the same time as the EAP Success message in the EAP-Payload AVP (cf. the last exchange in the optimized protocol, section 3.2).

AAA protocols such as Radius [7] and Diameter [8] support creation of new attribute value pairs for the new applications needs. So we define the AAA authentication AVP to carry the AAA service authentication information. Similarly there is no special AVP for access token transportation. So we define an Access Token AVP to carry the access token of the JN. The content of these two AVPs is detailed in the next two paragraphs.

AAA Authentication AVP The data field of the AAA Authentication AVP contains:

- $cert_{AAA}$: the X.509v3 certificate of the AAA service
- ID_{JN}: the identity of the JN that it has sent in the first exchange
- $Sign_{AAA_i}(R_{AAA}, R_{JN}, ID_{JN})$: the piece of the AAA service signature computed by the server number i on the random numbers R_{AAA} and R_{JN} together with the identity of the JN. Its length is equal to the AAA service private key length. According to NIST recommendations 1024 bits RSA private key length is enough for most of the applications until 2010 [9]. But this maybe insufficient for communicating critical data.

These information actually form the authentication material of the AAA service that are sent in the fourth exchange of the non-optimized protocol (cf. section 3.1).

Access Token AVP The data field of the Access Token AVP contains:

- T_{JN}: the access token deadline, it specifies the expiration time of the access token as the number of seconds since midnight on 1st January 1970
- $Sign_{AAA_i}(IP_{CGA_{JN}}||T_{JN})$: the piece of the AAA service signature computed by the server number i on the concatenation of the IP address and the access token deadline of the JN. The IP address is a Cryptographically Generated Address (cf. section 4.1)

When the CN receives at least t Access Token AVPs from t AAA servers, it computes $Sign_{AAA}(IP_{CGA_{JN}}||T_{JN})$ [6] then obtains its access token: $AT_{JN} = T_{JN}||Sign_{AAA}(IP_{CGA_{JN}}||T_{JN})$ (cf. Fig. 4). This access token is like a passport for JN. Henceforward it is added to JN's traffic during the access phase, and JN's neighbors check its validity to establish if the JN was authorized or not by the AAA service. Figure 5 illustrates a JN sending a message M to a destination

Dest. This message includes AT_{JN} and has $IP_{CGA_{JN}}$ as a source IP address. IN, R_1 and R_2 are relay nodes and operate as EPs. They examine AT_{JN} before relaying M. In the reality the only node that can verify AT_{JN} is the immediate neighbor IN (cf. section 4).

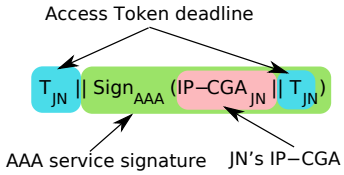

Fig. 4. Access Token content ($IP\text{-}CGA_{JN} = IP_{CGA_{JN}}$)

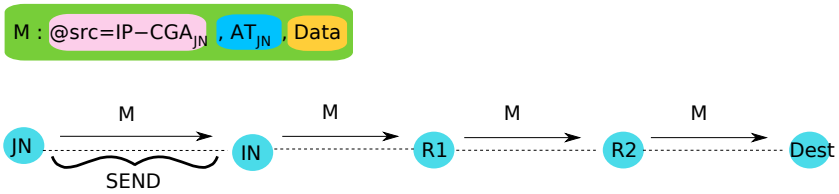

Fig. 5. Access Token Adding to JN's Messages

4 The Access Token Usage to secure the Access Phase

Unlike the solution proposed in [10], we make use of an access token to control and secure the access to the ad-hoc network. Ad-hoc nodes need not keep a list of authorized nodes in their caches. AAA servers need not broadcast a message each time a new node is authorized to access the network, either.

Authenticated and authorized nodes must rather add their access tokens to their messages in the beginning of the access phase to prove their legitimacy. You can refer to section 3 for the access token content and the way it is obtained by the JNs.

In this section we give an overview of CGA and SEND for better understanding of their role in the validity check of the access token during the access phase.

4.1 Brief Introduction to CGA and SEND

Cryptographically Generated Address (CGA) [11] strongly links the public key of a CN to its IP address. It is computed using CGA parameters that include:

– the node's public key
– a modifier: an integer that avoids collision by introducing randomness
– collision count: an integer equal to 0, 1 or 2

Once a CGA is computed by a JN, it becomes its IP address or IP-CGA. CGA parameters are necessary for the JN's neighbors to verify the CGA address. For this reason SEND transports the CGA in the IP source address and the CGA parameters in the CGA option of the same packet (cf. the paragraph below).

SEND [12] is a secure version of the Neighbor Discovery Protocol (NDP). NDP enables a JN to discover its neighboring nodes or determine if they are still reachable by mainly soliciting them to advertise their link-layer addresses. So JN sends Neighbor Solicitation messages to its immediate neighbor IN from which it receives Neighbor Advertisement messages as a reply (cf. Fig. 6).

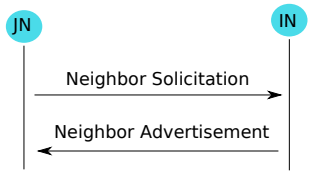

Fig. 6. Neighbor Discovery Protocol

SEND uses CGAs (cf. the previous paragraph) and includes the CGA option, the RSA option and the Timestamp and Nonce options in NDP packets to secure NDP. The CGA option contains the CGA parameters. The RSA option contains an RSA signature generated on all the other options, the packet source and destination IP-CGAs, and the packet data.

After SEND exchanges, the JN and its neighbors record the CGAs of the neighboring entities and the necessary parameters for verification in their SEND caches.

4.2 Access Token Validity Check by the Immediate Neighbors

Immediate neighbors (INs) of the JN receive JN's access token AT_{JN} (cf. Fig. 4) in the traffic of JN (cf. Fig. 5). They check the validity of AT_{JN} relying on the information recorded in their caches filled in during SEND exchanges (cf. section 4.1). The first step is to validate JN's IP-CGA after SEND exchanges by means of the CGA parameters.

The second step is to check if the IP-CGA received in the traffic is equal to that validated using above. The third step consists in concatenating the IP-CGA of the received traffic with the expiration time of validity T_{JN} in AT_{JN} and in verifying the consistency of this concatenation with the signature $Sign_{AAA}(IP_{CGA_{JN}}||T_{JN})$ in AT_{JN}. This verification can be processed thanks

to the public key of the AAA service. This public key was previously registered by the INs after their authentication. In case of successful verification, INs add AT_{JN} and T_{JN} in their caches entry for JN. Henceforward INs no longer proceed with the previous check until the expiration of T_{JN}. They simply check if the access token received is equal to that in their caches and that the deadline T_{JN} has not expired yet.

4.3 Access Token Validity Check by the Destination

In the subsection 4.2 we demonstrated that the validity check of the access token is based on SEND and CGA. Consequently only INs can do the check. It is important that remote destination nodes can do the check, too, in order to detect malicious nodes attacks like spoofing and replaying attacks. So we need a protocol like SEND that carries CGA parameters beyond the immediate neighborhood of JN. That protocol should never carry the access token without securing it by a signature, a hash or a ciphering. The two-way protocol depicted in figure 7 meets these two requirements.

In the first exchange with the destination node Dest, the JN sends its CGA parameters namely the modifier, the public key (PK) and the collision count for its IP-CGA verification. In addition to these parameters, it sends a nonce that identifies this message and the corresponding response sent later by Dest. It sends also a sequence number in order to avoid message replaying attacks.

The sequence number is equivalent to the timestamp sent in the Timestamp option of SEND packets. If a mechanism exists to correctly synchronize ad-hoc nodes, JN can send a timestamp rather than a sequence number.

g, p, and A_{JN} are Diffie Hellman parameters [13] that JN sends as well, thus asking the destination to compute a Diffie Hellman shared key K. Finally AT_{JN} is also included in the first exchange in order to prove that the JN was authenticated and authorized by the AAA service.

The RSA signature of the JN is computed on all these elements plus the source and destination IP addresses of the message, respectively $IP_{CGA_{JN}}$ and $IP_{CGA_{Dest}}$. Thanks to this signature the access token is protected from spoofing attacks. No other node can indeed send the signed CGA parameters for $IP_{CGA_{JN}}$ and AT_{JN} validation (cf. section 4.1).

Now Dest can validate AT_{JN} following the same procedure described in section 4.2. It first authenticates the JN thanks to the RSA signature and establishes, so, that it is effectively the owner of the public key received. Then the validity of $IP_{CGA_{JN}}$ and AT_{JN} can be checked.

Dest moreover computes the Diffie Hellman shared key K. Let's d (resp. j) be the secret Diffie Hellman number of Dest (resp. JN), then $A_{Dest} = g^d mod\ p$ (cf. Fig. 7) and $K = A_{JN}^d mod\ p = A_{Dest}^j mod\ p$.

If all these operations were successful, Dest responds with a similar message containing the same nonce of JN's message. JN achieves the same verifications on Dest's elements. If these operations were successful, JN and Dest have been successfully and mutually authenticated. Both have also proved that they were successfully authenticated by the AAA service and so authorized to access the

network. In addition they have established a Diffie Hellman shared key K that they can use in their later communications. Hence they no longer need to send their access tokens until one of the deadlines T_{JN} T_{Dest} expires.

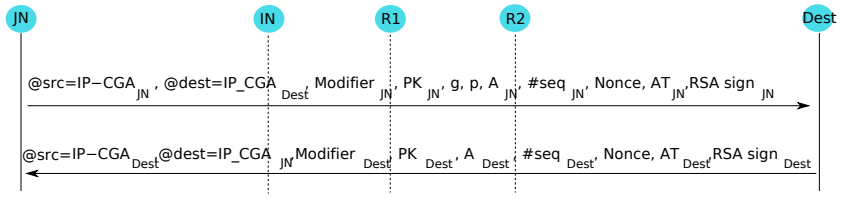

Fig. 7. Protocol for source to destination access token checking

4.4 Robustness to Classical attacks

Signing the concatenation of $IP_{CGA_{JN}}$ and T_{JN} with the AAA service's private key (cf. section 3.3) preserves the IP-CGA from spoofing. It preserves at the same time the access token from spoofing. If we place ourselves in the case where just the access token is attached to JN's traffic during the access phase (cf. Fig. 5), this preservation can unfortunately be guaranteed only in the immediate neighborhood of JN since only immediate neighbors can execute SEND (cf. section 4.1).

INs can check the validity of the access token and so detect an access token spoofing attack. However destination nodes can not detect a spoofing attack without SEND. Figure 8 illustrates an example of spoofing. The node R_1 pretends that it is relaying one of JN's messages but in fact it has generated this message itself. It has placed into it the $IP_{CGA_{JN}}$ as a source IP address and AT_{JN} as the attached access token. Dest can not receive SEND messages from JN, so there are no CGA parameters for JN in Dest's cache. Dest can't verify the validity of AT_{JN} sent by R_1 and accepts the message as if it was sent by JN. Consequently it does not detect R_1 as a malicious node.

The two-way protocol described in section 4.3 solves this problem. It transfers CGA parameters for the AT_{JN} validation and protects these elements with a RSA signature. Replay attacks are also avoided by means of this protocol because it employs sequence numbers: Dest detects that a message was replayed by comparing its sequence number to those received in the previous messages.

5 Conclusions and perspectives

In this paper we investigated an optimized version of the AAA protocol described in [2]. Furthermore we demonstrated how to secure the access phase by means of a special form of the access token associated to the use of SEND, CGA, and a two-way protocol that we conceived in the section 4.3.

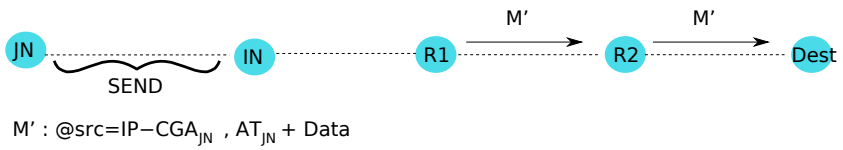

Fig. 8. Access Token Spoofing

The number of AAA exchanges and their length may be a difficult point in the implementations. We are now evaluating the overhead induced to make sure this authentication and access control protocol is usable for network access control.

Acknowledgment

We are thankful to ANR (Agence Nationale de la Recherche) for financially supporting the project TLCOM MobiSEND.

References

1. Nikolov, M.: Exploiting social and mobile ad hoc networking to achieve ubiquitous connectivity (2008) `http://developer.symbian.com/main/documentation/technologies/future_technology_ideas/milen_nikolov.jsp`.
2. Larafa, S., Maknavicius, M., Chaouchi, H.: Light and Distributed AAA Scheme for Mobile Ad-hoc Networks. First Workshop on Security of Autonomous and Spontaneous Networks, SETOP 2008, Loctudy, France (october 2008)
3. Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., Levkowetz, H.: Extensible Authentication Protocol (EAP). RFC 3748 (June 2004)
4. Shamir, A.: How to Share a Secret. Communications of the ACM (1979)
5. : ISO [9798-3] `http://www.iso.org/iso/fr/search.htm?qt=9798-3&published=on&active_tab=standards`.
6. Shoup, V.: Practical Threshold Signatures. Theory and Application of Cryptographic Techniques (2000)
7. Rigney, C., Willens, S., Rubens, A., Rubens, A.: Remote authentication dial in user service (radius). RFC 2865 (June 2000)
8. Calhoun, P., Loughney, J., Guttman, E., Zorn, G., Arkko, J.: Diameter Base Protocol. RFC 3588 (September 2003)
9. Keylength: `http://www.keylength.com/en/4`.
10. Khakpour, A., Laurent-Maknavicius, M., Chaouchi, H.: WATCHMAN: An Overlay Distributed AAA Architecture for Mobile Ad hoc Networks. The Third International Conference on Availability, Reliability and Security (ARES 2008), IEEE Computer Society, Barcelona, Spain (March 2008)
11. Aura, T.: Cryptographically Generated Addresses (CGA). RFC 3972 (March 2005)
12. Arkko, J., Kempf, J., Zill, B., Nikander, P.: SEcure Neighbor Discovery (SEND). RFC 3971 (March 2005)
13. Diffie, W., Hellman, M.: New directions in cryptography. IEEE Transactions on Information Theory 22, pages 644 to 654 (1976) `http://www.rsa.com/rsalabs/node.asp?id=2248`.