Light and distributed AAA scheme for mobile ad-hoc networks

Claire Sondès Larafa, Maryline Laurent, Hakima Chaouchi

Light and Distributed AAA Scheme for Mobile Ad-hoc Networks

Sondes LARAFA (sondes.larafa@it-sudparis.eu)*
Maryline LAURENT-MAKNAVICIUS
(Maryline.Maknavicius@it-sudparis.eu)*
Hakima CHAOUCHI (hakima.chaouchi@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. Our work focuses on a light and distributed AAA infrastructure that supports authentication and authorization. We make use of a distributed protocol based on the threshold cryptography theory, and the ISO 9798-3 authentication protocol standard. Access to the network is also controlled by an ACCESS TOKEN generated by the AAA service and referring to some Cryptographically Generated Addresses.

Keywords: Ad-hoc networks, Distributed AAA infrastructure, Threshold cryptography, Cryptographically Generated Addresses

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 also likely to be snuffed by providers of content, multi-media, games, etc, since they are less expensive than infrastructure networks and do support better users' mobility.

The wireless nature of ad-hoc connections and their dynamicity make them vulnerable to attacks. Hence security is of large concern. There are two approaches for security: secure individual communications and secure the whole network. Securing individual communications of a node A with a node B means that A shares some security material with B or trust a third party to get security material and communicate with B. Securing the whole ad-hoc network means that every joining node must authenticate itself to a trust entity in the network in order to be able to send traffic via the network. The second approach allows better network security since the access to the network is controlled.

* TELECOM & Management SudParis, 9 rue Charles Fourier, 91011 EVRY, FRANCE

Network access control can be achieved with AAA infrastructures [ACG+00] [RWRS00] [CLG+03]. In this paper, we focus on a AAA solution for ad-hoc networks.

The simplest approach to provide AAA functionality in ad-hoc network is to assign a single node to be the AAA server. The success of this scheme depends on that single AAA server node. This approach is not fault tolerant and is highly vulnerable, since failure of one node or compromising it by an adversary breaks the system. In addition, availability of the AAA server is not always possible given the expected mobility and unpredictability of ad-hoc networks. Therefore, a single AAA server cannot effectively service a whole ad-hoc network.

Replicating a fully functional AAA server on several different nodes, say n nodes (n AAA servers) can ensure the missing robustness in the single AAA server scheme. With n replicas, the system can withstand $n-1$ failures because the AAA service is available as long as there is at least one operational AAA server. Availability has also been improved since a joining node has a better chance of reaching one of the n AAA servers to be authenticated. Unfortunately, the system has become more vulnerable. An adversary needs only compromising one of the n AAA servers to compromise the whole system, because each replica has full knowledge of the system's secret. Therefore using replicated AAA servers is not a viable solution in ad-hoc networks.

To insure robustness, it would be better distributing the AAA server on the n most powerful and trustful mobile nodes (future AAA servers) that can be selected by a third party (organization, operator, etc). There are different ways to distribute AAA servers. This mainly depends on the authentication method being used. Our solution is based on authentication with public certificates [HFPS99]: the AAA servers use the same one public certificate and each new arriving node must have already a public certificate being delivered by the same certification authority. The location of the certification authority might be in the infrastructure and accessible in an opportunistic way by the ad hoc nodes, or located somewhere in a specific trusted ad hoc node. This is not addressed in this paper.

A public certificate can be used by several nodes at the same time by sharing the same private key (also called secret key). Shamir proposed a way to share a secret key between n different parties so that only $t \leq n$ parties associated among them can compute the original secret key [Sha79]. Shoup then proposed computing some signature shares, by using these key shares, so that the combination of those signature shares gives only one signature which is equivalent to the signature computed with the private key [Sho00].

This document introduces a new AAA architecture in mobile ad-hoc network. It is a distributed and dynamic architecture composed of AAA servers, AAA clients and Enforcement Points (EP). AAA servers are selected by the operator or the organization and AAA clients/EP are the nodes that have been already authenticated by the AAA servers at the time of joining the ad-hoc network.

2 State of the art and our work contribution

Basically three research works investigated decentralized and not necessarily distributed AAA services.

WATCHMAN [KLMC08] proposes a solution with one Master AAA Server that elects slave AAA Servers among the most trustful and powerful ad-hoc nodes and according to

their location. It introduces a way to do the election and to have thus more than one AAA server in the network responsible for authentication. However this solution is not distributed enough. A joining node is authenticated by only one AAA server, and this makes the authentication vulnerable to corruption of one AAA server. WATCHMAN broadcasts a message each time a new node is authenticated and accepted to join the network. Thereafter all nodes add a new entry to their databases for the joining node.

SAACCESS [CLM07] proposes to delegate operator's AAA service to a decentralized and distributed ad-hoc AAA service when the ad-hoc network is disconnected from the operator's infrastructure network. However it is generic and does not explicit the distributed infrastructure and the protocols involved during authentication, authorization and accounting. It also separates these three functions so that a node can operate one or several of these three functions depending on there available resources such as battery, CPU, memory,etc.

LAMSAL's solution [Lam03] describes a decentralized and distributed model based on threshold cryptography where an AAA server is distributed among a group of ad-hoc nodes. LAMSAL gives an outline of the authentication protocol as well but with no details. It does not detail the authorization step neither.

Finally SKiMPy [MJTY05] analyzes a way for key management in a rescue scenario for secure exchanges after nodes authentication. It shows that node to node authentication with pre-established public certificates which mirror the third party hierarchy is the most efficient way for authentication. This is not suitable for us because our framework is not highly tightened to the third party. This one does not necessarily have a hierarchical model neither.

Our work proposes answers to the unsolved points of previews research works. Improvement of some other points is targeted too. First it designs a distributed AAA service model where the authentication is handled by several AAA servers and not just by a single entity. We also improve LAMSAL's authentication protocol and we avoid broadcasting new authenticated nodes' information. We rather propose a solution with ACCESS TOKEN that authenticated nodes must append to their messages so that messages are relayed by the other routing nodes.

3 Theory applied to our distributed AAA model

Shamir's threshold cryptography system [Sha79] is the basis of our AAA architecture model. This system allows the trusted third party to share the secret key (associated to the public key PK) of our AAA service between AAA servers. These servers can later encrypt in a distributed way any information using the key shares. They can especially sign (i.e. encrypt) an information in order to authenticate to another party as one AAA server. Thanks to threshold cryptography, attacks on the AAA service become more difficult as an adversary needs to corrupt at least t servers out of n for controlling that service.

It was Shoup who designed a way to use those key shares for generating signature pieces, which once put together, can be checked as being valid with the public key PK [Sho00]. These operations do not lead to the associated secret key disclosure and this is very important for the security of the system.

3.1 Secret key sharing among AAA servers

In asymmetric cryptography systems, we are given (M, e) and (M, d) such that: $M = pq$, $p = 2p' + 1$, $q = 2q' + 1$, $m = p'q'$ $1 \leq e \leq m$, $pgcd(e, m) = 1$ and $e \cdot d \equiv 1 \ mod \ m$.

The pair (M, e) is the public key (PK) and (M, d) is the secret key (SK). As M is the same in these two pairs, for convenience, we call in this article e the public key and S the secret key. In RSA system, it is known that if x is fixed, $x^S \ mod \ m$ is an encryption of x that can be decrypted by computing $(x^S)^e = x \ mod \ m$.

Let $AAA_1, AAA_2, ..., AAA_n$ be the servers in our framework. According to Shamir [Sha79], they can share the same secret key S while none of them knows it. This secret corresponds to the global AAA service private key. Each node only knows a key share s_i. Let $t \leq n$ be the threshold parameter. t out of n AAA servers associated $(AAA_{i_1}, AAA_{i_2}, ..., AAA_{i_t})$ are able to find S using the $\{s_{i_j}\}_{i_j \in \{1,..,n\}, j \in \{1,..,t\}}$, but they never compute it during the service. Here is a quick overview of how to compute s_i:

1. Choose $f(X) = \sum_{i=0}^{t-1} a_i \cdot X^i \in \mathbb{Z}[X]$, where $a_i \in \{0, .., m-1\}$ and $a_0 = S$
2. $\forall i \in \{1, .., n\}, s_i \equiv f(i) \ mod \ m$

Without loss of general information, suppose that nodes $AAA_1, ..., AAA_t$ (i.e. $i_1 = 1$, $i_2 = 2$,...,i_t=t) gather to compute S, then according to Lagrange interpolation:

$$f(X) \equiv \sum_{j=1}^{t} s_j \cdot L_j(X) mod(m) \text{ where } L_j(X) = \frac{\prod_{i=1, i \neq j}^{t} (X - s_i)}{\prod_{i=1, i \neq j}^{t} (s_j - s_i)}$$

As $f(0) = a_0$ and $a_0 = S$, $f(0) = S$. Computing f using the $\{s_i\}_{i \in \{1,..,t\}}$ allows the servers to calculate finally S.

3.2 Authentication of a AAA service by an incoming node using Shoup's principle

During authentication exchanges, the joining node JN sends a random number R_{JN} to $\{AAA_i\}_{i \in \{1,..,n\}}$. R_{JN} is the information to sign by the AAA servers to authenticate themselves to the JN. Each AAA_i computes a signature share using the following formula [Sho00]:

$sign_i(R_{JN}) = {R_{JN}}^{2g \cdot s_i}$ where $g = n!$ and sends it to the JN.

For convenience, let $l_i = L_i(0)$. JN combines the signature shares as follows:

JN computes $w = sign_1(R_{JN})^{2l_1} \cdot sign_2(R_{JN})^{2l_2} \cdot ... \cdot sign_n(R_{JN})^{2l_n}$

w is such that: $w^e = R_{JN}^{e'}$ where $e' = 4g^2$

As $pgcd(e, e') = 1$, $\exists (a, b) \quad e'a + eb = 1$

Knowing (a, b), JN computes $sign_{AAA}(R_{JN}) = w^a R_{JN}^b$, which is the signature of the AAA service on R_{JN}.

If $sign_{AAA}(R_{JN})^e = R_{JN} mod(m)$, then the authentication of the AAA service by the JN succeeds.

If there is no equality, JN cannot authenticate the service. Either some servers are corrupted or some non-AAA server nodes are pretending to be AAA servers.

3.3 Mutual Authentication between an incoming node and a non distributed AAA server

Due to wireless connections and mobility, the environment of ad-hoc networks is a hostile environment and it is desirable to carry out mutual authentication during access to the network. On one hand a joining node JN must authenticate itself to the group of AAA servers. On the other hand the group of AAA servers must be authenticated by the JN (cf. section 3.2). This exchange is symmetric and can be done in any order.

ISO [9798-3] [iso] standard provides a three-way protocols for mutual authentication with public certificates. This protocol takes place between two entities A and B (cf. Fig 1). Since we need to use it in a distributed environment, we propose to extend it in section 5.2.

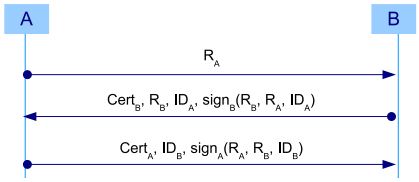

Figure 1: ISO three-way protocol

In figure 1, A generates a random number challenge R_A used to detect replay attacks and to prevent forgery then it sends it to B. B authenticates itself first. It generates another random number R_B and sends it to A with its certificate $Cert_B$, the A's identity ID_A, and its signature (generated with its secret key over R_B, R_A and ID_A). ID_A establishes that the signature actually was intended to be sent to A. A verifies the integrity of the signed information and establishes the identity of B. Then it authenticates itself likely.

4 Proposed Framework

To ensure AAA service, ad-hoc nodes are either AAA servers or AAA clients/Enforcement Points.

The service initiation depends on a higher institution (a trusted third party) like a Service Provider, a Military institution, a Rescue Organization, etc, which indicates the mobile nodes to be AAA servers. These servers offer AAA services.

As soon as a sufficient number n of AAA server devices (n fixed by the trusted third party) present in the ad-hoc network is reached, the service can start. n can be equal to 1. In this case there is one server in the network. This can be sufficient if the total number of nodes to be authenticated is small (not more than 10). However if this server breaks

down or disconnects, the service will be unavailable. For this reason it is better to choose n greater than 1.

Every user of the network must be authenticated once the service is available. Authentication is mutual and it is carried out with public certificates [HFPS99]. AAA servers get only one certificate, and use it as a group. Threshold cryptography requires that at least any t AAA servers out of n $(=t+k)$ participate to authentication.

We designed a distributed authentication protocol based on Shoup's principal (cf. section 3.2) and ISO three-way protocol (cf. section 3.3). It is executed between a joining node and AAA servers.

After a successful authentication, the servers send an ACCESS TOKEN to the joining node which then becomes an authenticated node and a trusted node as such. The ACCESS TOKEN is to some extent like a valid passport for the joining node.

In our framework authenticated nodes are not only AAA clients but also Enforcement Points. They watch out the traffic received from their neighbors and examine especially ACCESS TOKEN. When this one does not exist or is false they do not relay the traffic (cf. section 5.4).

The proposed formula of the ACCESS TOKEN and its validity check procedure suppose that ad hoc nodes are IPv6 nodes and use the SEcure Neighbor Discovery protocol (SEND) [AKZN05]. SEND makes use of Cryptographically Generated Addresses (CGA) [Aur05] to secure the Neighbor Discovery protocol [NNS98]. CGA is a method for binding a public key to an IP address so that the sender of a given traffic is highly likely to be the owner of the claimed address. This method targets mitigating most of the IP addresses spoofing.

Nodes using SEND protocol are registering the neighbors' IP address into their neighbor cache. In our framework, the cache is helpful for checking the ACCESS TOKEN's validity.

Next sections give details on the points raised in this part. They give a more precise description of authentication and authorization exchanges.

5 Authentication and Authorization

Joining clients have to authenticate themselves to the AAA service i.e. to at least t AAA servers before accessing to the ad-hoc network. During authentication, AAA messages are exchanged between the clients and the servers (cf. section 5.1). AAA messages contain authentication information (credentials, signatures, etc.) provided by the two parties. This information can be carried by a AAA application that extends the base AAA protocol (cf. Fig 2) to EAP messages support [EHZ05]. In section 5.2, we define an EAP method called EAP-ADHOC. EAP-ADHOC is mainly based on the ISO three-way protocol that we adapted to fit our distributed context (cf. section 3.3).

If the authentication is successful, AAA servers send an AA-Answer message (AA stands for Authentication and Authorization) to the joining client containing an EAP-Success message and an ACCESS TOKEN. The EAP-Success message informs the client that the authentication succeeded. If the authentication fails, AA-Answer message contains an EAP-Failure [EHZ05].

The ACCESS TOKEN is the authorization information necessary to the client during his access to the network i.e. when sending traffic through the network. Henceforward

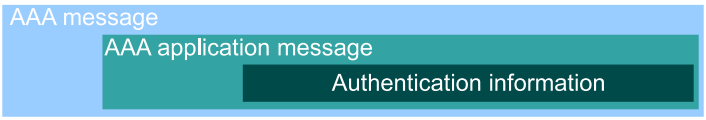

Figure 2: Encapsulation of authentication information in AAA messages

this information must be included in all the client's traffic in order to be relayed by the other nodes. Relay nodes are Enforcement Points (EP) that examine ACCESS TOKENS and decide to forward the traffic if the TOKEN is correct or to drop it if the TOKEN is false or absent.

5.1 AAA Exchanges

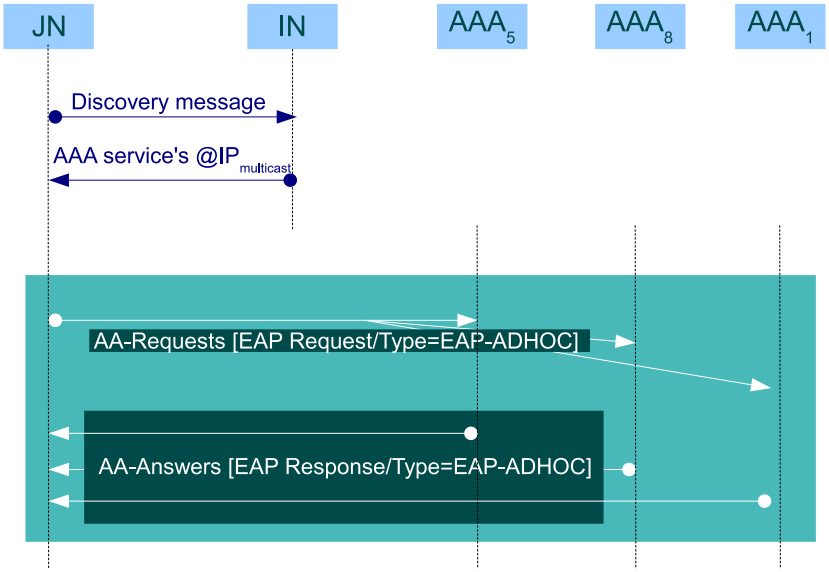

Figure 3: Sequence diagram of AAA authentication messages ($t = 3$)

Figure 3 illustrates the authentication and authorization sequence diagram where $t = 3$ and $n \geq 8$. A joining node JN wants to access to the network. First it connects to it. Then it uses SEND to discover its neighbors. We call this message *Discoverymessage*. IN is one of the immediate neighbors of JN. JN was already authenticated by the AAA service and functions as an EP. When it receives JN's *Discoverymessage*, it looks for ACCESS TOKEN. If the ACCESS TOKEN is present, it processes it. Otherwise it responds with

the list of the IP addresses of the AAA servers. An extension to ICMPv6 message might be possible to support these AAA servers' addresses.

Then JN initiates AAA exchanges with AAA servers. It sends to each one of them an AA-Request. As depicted on figure 3, the servers AAA_1, AAA_5 and AAA_8 were the first to respond with AA-Answer messages. JN does not wait for the other servers' answers and continues its exchanges with only the servers AAA_1, AAA_5 and AAA_8.

Figure 3 shows that AAA_5 responds first, then AAA_8, then AAA_1. Actually there is no order because of the vagaries of the ad-hoc network. EAP Requests and Responses are encapsulated in AA-Requests and AA-Answers. Next section details the content of these messages.

5.2 EAP-ADHOC Exchanges

EAP-ADHOC is an EAP method [ABV+04] that can run over the Diameter EAP application [EHZ05]. As depicted in figure 4, EAP-ADHOC is strongly inspired from the ISO three-way protocol. To avoid Denial of Service (DoS) attacks, AAA servers must not do heavy operations like encryption or decryption as soon as they are solicited by the JN. This constraint is easier to meet with ISO [9798-3] protocol than with UIT-T X.509/ISO protocol. For the same reason, the JN must also authenticate itself before authenticating the AAA service.

Our EAP method design takes into consideration these points and includes the following steps (cf. Fig 4):

1. AA-Request of the JN contains its identity ID_{JN} encapsulated in an EAP-Request

2. AA-Answers of the servers contain a random number R_{AAA} to be encrypted by the JN. All the servers send the same random number that they find in their data bases of random numbers. These data bases are created when the AAA server node is created. They are also periodically updated. How to maintain and secure these data bases is not addressed in this paper.

3. JN generates another random number R_{JN} and sends it with its certificate $cert_{JN}$, the identity of the AAA service ID_{AAA}, and its signature on $(R_{JN}, R_{AAA}, ID_{AAA})$ to each one of the t-first responding servers.

4. Each server AAA_i verifies the integrity of JN's information and responds if JN's authentication is successful. Its answer contains the certificate of the AAA service $cert_{AAA}$ and its partial signature $Sign_{AAA_i}(R_{AAA}, R_{JN}, ID_{JN})$ computed with its key share s_i (cf. sections 3.1 and 3.2).

5. When JN receives t answers from the servers, it combines their partial signatures and obtains the signature of the AAA service. Thanks to the public key in $cert_{AAA}$, it checks the integrity of the information received. If the service authentication is successful, JN sends an AA-Request with an empty EAP-Request.

6. Each server AAA_i sends an EAP-Success in its AA-Answer to inform that the mutual authentication was successful. It includes a partial signature that JN combines with the other servers' partial signatures (using Shoup's principle) to obtain its ACCESS TOKEN.

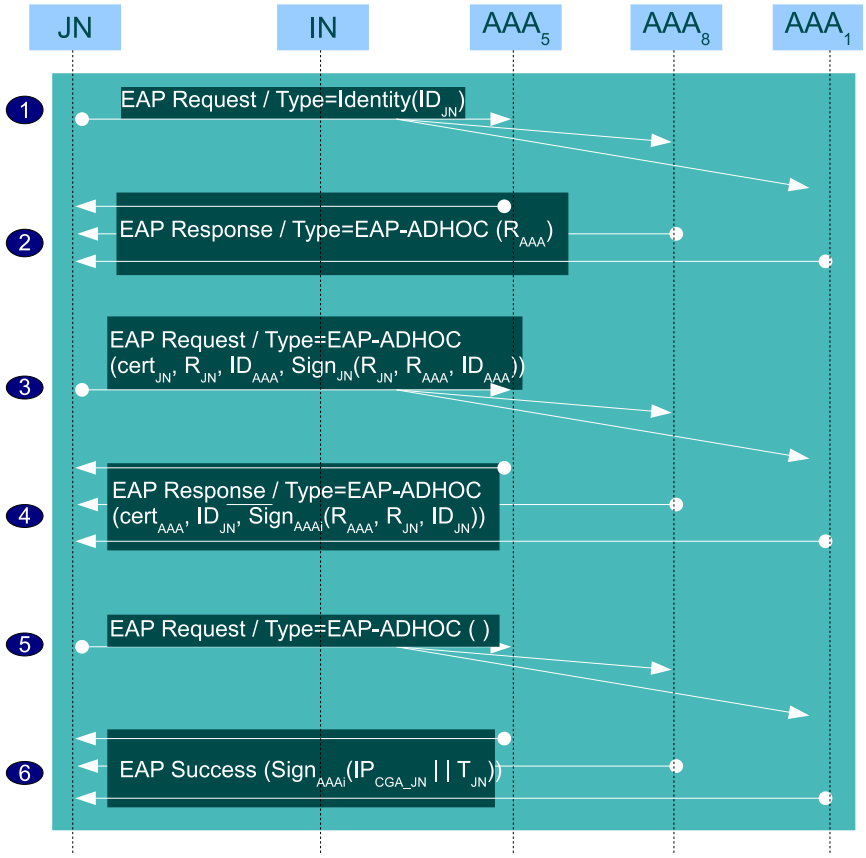

Figure 4: Sequence diagram of EAP messages ($t = 3$)

5.3 ACCESS TOKEN

We use IP CGA (Cryptographically Generated Addresses) (cf. section 4) in our solution [Aur05], which is necessary for the authorization step. JN computes its IP CGA, $IP_{CGA_{JN}}$, before connecting to the network and uses it in its future exchanges.

AAA servers authorize JN to access the network for a limited period of time until time T_{JN}. We assume that the same time windows are known by AAA servers so any AAA server uses the same T_{JN} for JN. Each server computes the partial JN's ACCESS TOKEN, using $IP_{CGA_{JN}}$ and T_{JN}. After getting t-first partial tokens, JN combines them into the following ACCESS TOKEN AT_{JN}, as follows:

$AT_{JN} = T_{JN}|Sign_{AAA}\{IP_{CGA_{JN}}|T_{JN}\}$, where "|" means concatenation.

Henceforward, JN precedes all its traffic with its ACCESS TOKEN AT_{JN}.

5.4 Access to network

IN, the immediate neighbor of JN, is an Enforcement Point and checks the validity of AT_{JN} before relaying JN's traffic. To do so, it relies on two kinds of information: the ACCESS TOKEN and its neighbor cache filled in during SEcure Neighbor Discovery Protocol (SEND) [AKZN05] exchanges. SEND is executed when a node is booting on a new network or when a nearby node detects it as new in its neighboring area and sends SEND solicitation. After SEND execution, the neighbor cache of the nodes is updated with the IP CGA of the node.

We suppose that IN and JN already validated their Cryptographically Generated Addresses thanks to SEND exchanges and IN needs to check AT_{JN}.

The checking consists in concatenating the IP CGA of the received packet with the end time of validity AT_{JN} and verifying its consistency with the signature of the JN's ACCESS TOKEN. This verification can be processed thanks to the public key of the AAA service. This public key was previously registered by IN during its own AAA exchanges with the servers. In case of successful verification, IN registers the ACCESS TOKEN and the end time of validity in its cache entry for JN. Henceforward IN no longer proceeds the previous operations until the end of time of validity. It simply checks if the ACCESS TOKEN and the deadline are correct. Note that this check is only executed by the immediate neighbors of the JN when they relay its traffic. The other relaying nodes and the destination of the traffic do not repeat the checking operations.

6 Conclusions and perspectives

Our solution handles mainly the authentication and authorization issues of ad-hoc networks. It proposes a distributed AAA infrastructure where nodes joining the ad-hoc network mutually authenticate with the AAA servers. We designed a distributed authentication protocol based on the threshold cryptography and ISO [9798-3] three-way protocol. We also proposed a formula for an ACCESS TOKEN needed to authorize nodes to access the network. A validity check procedure of the ACCESS TOKEN was detailed as well.

Our AAA infrastructure is in fact a light infrastructure. We believe that new services will be created to be used over ad-hoc network and that service providers will need to charge these services. So accounting is a real issue in ad-hoc networks. [CLM07] proposes a model for accounting when there is a close relationship between the ad-hoc network and the trusted third party. It is still necessary to define a distributed model for a quasi-independent ad-hoc network. It is also necessary to define the accounting messages exchanges and their format.

Besides an implementation or/and simulation is to be accomplished to evaluate several parameters such as the density of AAA traffic. Thereafter we can improve our solution.

References

[ABV+04] B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson, and H. Levkowetz. Extensible Authentication Protocol (EAP). RFC 3748, June 2004.

[ACG+00] B. Abobaa, P. Calhoun, S. Glass, T. Hiller, P. McCann, H. Shiino, G.Zorn, G. Dommety, C. Perkins, B. Patil, D. Mitton, S. Manning, M. Beadles,

P. Walsh, X. Chen, S. Sivalingham, A. Hameed, M. Munson, S. Jacobs, B. Lim, B. Hirschman, R. Hsu, Y. Xu, E. Campbell, S. Baba, and E. Jaques. Criteria for Evaluating AAA Protocols for Network Access. RFC 2989, November 2000.

[AKZN05] J. Arkko, J. Kempf, B. Zill, and P. Nikander. SEcure Neighbor Discovery (SEND). RFC 3971, March 2005.

[Aur05] T. Aura. Cryptographically Generated Addresses (CGA). RFC 3972, March 2005.

[CLG+03] P. Calhoun, J. Loughney, E. Guttman, G. Zorn, and J. Arkko. Diameter Base Protocol. RFC 3588, September 2003.

[CLM07] H. Chaouchi and M. Laurent-Maknavicius. SAACCESS: Secured Ad hoc ACCess framework. NTMS 2007, Paris, May 2007.

[EHZ05] P. Eronen, T. Hiller, and G. Zorn. Diameter Extensible Authentication Protocol (EAP) Application. RFC 4072, August 2005.

[HFPS99] R. Housley, W. Ford, W. Polk, and D. Solo. Internet X.509 Public Key Infrastructure Certificate and CRL Profile. RFC 2459, January 1999.

[iso] ISO [9798-3]. `http://www.iso.org/iso/fr/search.htm?qt=9798-3&published=on&active_tab=standards`.

[KLMC08] A.R. Khakpour, M. Laurent-Maknavicius, and H. Chaouchi. 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.

[Lam03] Pradip Lamsal. AAA and PKI in Ad Hoc Networks, 2003. `http://citeseer.ist.psu.edu/652925.html`.

[MJTY05] P. Matija, A. Jon, P. Thomas, and R. Yves. SKiMPy : A simple key management protocol for MANETs in emergency and rescue operations. Lecture notes in computer science ISSN 0302-9743. Second European workshop, ESAS 2005, Visegrad, Hungary, July 2005.

[NNS98] T. Narten, E. Nordmark, and W. Simpson. Neighbor Discovery for IP Version 6 (IPv6). RFC 2461, December 1998.

[RWRS00] C. Rigney, S. Willens, A. Rubens, and W. Simpson. Remote Authentication Dial In User Service (RADIUS). RFC 2865, June 2000.

[Sha79] A. Shamir. How to Share a Secret. Communications of the ACM, 1979.

[Sho00] V. Shoup. Practical Threshold Signatures. Theory and Application of Cryptographic Techniques, 2000.