A Security Framework for Buddy System based MANET Address Allocation Scheme

Abdelhafid Abdelmalek[1], Zohra Slimane[1], Mohamed Feham[1] and Abdelmalik Taleb-Ahmed[2]

[1] STIC Laboratory University of Tlemcen Algeria

[2] LAMIH Laboratory University of Valenciennes and Hainaut Cambrésis France

Abstract
Security for dynamic address allocation service in Mobile Adhoc Networks is still an open issue. This paper proposes a security framework to thwart all possible attacks related to the buddy system based Distributed Dynamic Address Assignment Protocol (DDAAP) proposed by Thoppian and Prakash. Our proposed protocol called SHEAP (Secured tHrEshold based DDAAP) is based on a fully distributed PKI and a threshold based certified pool address allocation model. NS2 simulation results show that, while SHEAP is deployed, the increasing in communication overhead and latency is quite reasonable. The availability and the security are so guaranteed for the MANET auto-configuration service while still ensuring efficiently both network and security parameters for a newly arrived node.

Keywords: *Address Allocation, Buddy System, Mobile Adhoc Network, Security, Threshold Cryptography, PKI, SHEAP*

1. Introduction

Until now, none of the IP address assignment protocols for Mobile Adhoc Networks (MANETs) has been standardized. One of the main reasons is the security issue which is discussed in this paper. The secure initialization of a new joining node with network and security parameters to be trusted and to be able to participate actively in the MANETs (i.e. to get securely a unique IP address and a cer tified secret key) is a fundamental and difficult problem. Many IP address assignment schemes were proposed. However, some only have been secured and unfortunately all the proposed security mechanisms are weak [1].
In this paper, we focus on the Distributed Dynamic Address Assignment Protocol (DDAAP) [4] to build our security framework using threshold cryptography; the security of the proposed scheme is based on the intractability of the Discrete Logarithm Problem (DLP).

The paper is organized as follows. Section 2 gives a brief description about DDAAP. Our proposed model is presented in Section 3. Section 4 is devoted to the description and specification of pour protocol SHEAP. Simulation results are presented in Section 5. Section 6 concludes the paper.

2. DDAAP Scheme

Buddy System based protocol is one of the most interesting stateful dynamic configuration solutions for manets. The approach was first proposed by A. Misra et al. [2] , then respectively improved by M. Mohsin in [3] and M. Thoppian in [4]. In such scheme, nodes hold disjoint address pools using the concept of binary split similar to the buddy system for memory management. Subsequently, each node can independently configure any new node with a free IP address and a pool of free IP addresses. In [4], the proposed protocol DDAAP guarantees unique IP address assignment under all the following network scenarios:

- MANET Initialization
- New node joining the MANET
- Migration of Requester (the new node joining the MANET)
- Node Departure
- Concurrent Address Requests
- Message Losses
- Network partition and merge

The protocol is efficient. Most address allocations require unicast messages and local broadcast leading to low communication overhead and latency.

The IP address allocation process of DDAAP works as follows:
MANET starts with a single node (Initiator). The Initiator re-broadcasts a message requesting an IP address for a constant number of times after which it assigns itself the first IP address from the known IP address block and forms its free_ IP pool from the remaining addresses. After this initialization, every time a n ewly arrived node (Requester) requests an IP address, one of the already

configured nodes (Allocator) within communication range of the Requester initiates address assignment process for the Requester. If the Allocator has a non-empty free_IP pool, it allots the second half of the addresses from its free_IP pool to the Requester. Otherwise, it performs an expanding ring search to look for a node with a non-empty free_ IP pool.

As pointed out in our previous work [1], this scheme is vulnerable to the following known attacks: IP Spoofing, Exhaustion Address Space, Conflict Address, and Sybil Attacks. A malicious node could either spoof an already assigned or a free IP address. Getting several IP addresses by a Sybil node facilitate much its attack. A malicious node could also request address allocation for phantom nodes. By acquiring all valid addresses, it can deny others nodes to be configured. Another possibility, a malicious node could allot to a requester a duplicate address with possibly a non free_IP pool.

The next section presents a security framework to prevent all these possible attacks. Our approach relies on a distributed cooperative scheme using threshold signatures and joint free_IP pool and public key certificate.

3. Proposed Model

This section first defines the components of our threshold cryptography based trust model. Then we describe the principle of our secure auto-configuration scheme and state the assumptions under which it is operational.

In order to fill all the requirements quoted in [1] and to thus counter all possible attacks on auto-configuration service, we need a trust model and an auto-configuration scheme which will collaborate in providing a robust and secure scheme. It is obvious today that the design of any service scheme for Mobile ad hoc networks within hostile environment must avoid use of any centralized entity. Such centralized server should constitute an attractive target for attacks. In order to be protected from any single point of failure, we adopt in our solution for both schemes a fully distributed approach based on threshold cryptography.

3.1. Fully Distributed Trust Model

There have been many proposed solutions dealing with trust model problem in MANETs (we refer the reader to [5], [6] for a survey of the work in this area). These solutions are based either on symmetric or asymmetric cryptography. The first approach includes in particular the Key Agreement model, the Resurrecting Duckling model and Pairwise Key Pre-Distribution model. Such schemes are suitable for low intrusion tolerance and small networks where nodes are limited in resources. In the second approach, due to the lack of a fixed infrastructure many proposed solutions emulate the Certification Authority needed for issuing, distributing, renewing and revoking public key certificates. This category includes Self-Certified public key model, Self-Organization model, Identity Based model and Distributed Certification Authority (CA) model.
In this work, we focus on the Fully Distributed CA scheme [7] in conjunction with threshold cryptography which seems a perfect solution for spontaneous Mobile Ad hoc Networks in potential hostile environment [8], [9]. Threshold cryptography has received considerable attention in literature (refer to [10] for a survey). In a $(\boldsymbol{t},\boldsymbol{n})$ threshold cryptosystem , the network consists of $\boldsymbol{n}$ nodes, each node holds a share of the Network's private key. The On-line Certification Authority service is achieved transparently by a large enough subset of nodes (i.e. a number greater or equal to the threshold $\boldsymbol{t}$). An adversary needs to compromise more than $(\boldsymbol{t}-1)$ nodes in order to learn the secret, and corrupt at least $\boldsymbol{n}-(\boldsymbol{t}-1)$ to break the availability of the service. Hence, a robust threshold scheme should tolerate up to $(\boldsymbol{t}-1) < \boldsymbol{n}/2$ malicious faults.

3.1.1. Threshold Cryptographic Tools

In this section we describe some existing tools necessary to implement a threshold cryptosystem in a spontaneous MANETs. We need:

- To generate randomly and in a distributed manner (without a trusted party) a pair of MANET's private/public keys, to split the MANET's private key among the network and to allow shareholders to verify the correctness of their shares. This is done by a joint verifiable random secret sharing protocol based on Shamir's secret sharing [11], [12].
- To provide for any new joining node with a share of the MANET's private key.
- To provide a threshold digital signature scheme to sign issued, renewed or revoked certificates.

Let $P = \{P_1, P_2, \dots, P_n\}$ a set of $\boldsymbol{n}$ nodes ($|P| = \boldsymbol{n}$) forming the network. Each node P_i has a unique identity u_i. We denote *(s,h)* the pair of private/public keys of the fully distributed On-line Certification Authority (i.e. the MANET's keys).
The security of the proposed scheme is based on the intractability of the Discrete Logarithm Problem (DLP).

This being, let p and q be large primes such that $q|p\text{-}1$, and let $g \neq 1 \in (Z_p = GF(p))$ be an element of order q (i.e. g^q=1(mod$q$)).
The private key $s \in GF(q)$ and its corresponding public key h are bound by :

$$h = g^s(modp) \tag{1}$$

where (p,q,g,h) are public and s is secret.

a) Shamir's secret sharing

Following [11] a threshold (***t,n***) secret sharing scheme allows us to split a secret s in n secret shares s_i $(i \in \{1,..,n\})$ among ***n*** nodes in such a way that any a subset of ***t*** or more nodes can cooperate to recover the secret and less than ***t*** nodes cannot obtain any information about the secret.
A polynomial $f(z)$ of degree ***t***-1 is randomly chosen in $GF(q)$:

$$f(z) = s + \sum_{l=1}^{t-1} a_l\, z^l (modq) \tag{2}$$

which satisfies $f(0) = s$.

The secret share s_i is given by:

$$s_i = f(u_i) = s + \sum_{l=1}^{t-1} a_l\, {u_i}^l (modq) \tag{3}$$

According to some subset $Q = \{P_{i_1}, P_{i_2}, \dots, P_{i_t}\}$, where $(i_1 < i_2 < \cdots < i_t) \in \{1,..,n\}$, we recover the secret by:

$$s = \sum_{l=1}^{t} B_l(0).s_{i_l} \tag{4}$$

where $B_l(z) = \prod_{j \neq l} \frac{z - u_{i_j}}{u_{i_l} - u_{i_j}}$

b) Joint Verifiable Random Secret Sharing

We need in our trust model a distributed and secure protocol to perform the following operations:

- Generation of the On-line Certification Authority private/public keys
- Distribution of secret shares
- Checking correctness of shares (verifiability)

In doing so, we adopt here the secure Verifiable Secret Sharing (VSS) protocol [12] which allows to a set P of ***n*** nodes to jointly select a random secret and share it among the network. We modify slightly this protocol in such a way that no information is leaked to adversary to perform a biasing attack. Here is the protocol :

Step1. Initialization
Each $P_i \in P$ chooses randomly a secret $x_i \in GF(q)$ and calculates a partial public key:

$$h_i = g^{x_i}(modp) \tag{5}$$

In contrast of [12], this value is broadcast to all nodes secretly (see step 2).
The On-line Certification Authority public key h is computes by:

$$h = \prod_{i=1}^{n} h_i (modp) \tag{6}$$

so that its private key is:

$$s = \sum_{i=1}^{n} x_i (modq) \tag{7}$$

Step2. Distribution
In this step, the following operations are performed: generation of shares, then distribution of shares and partial public keys secretly, and lastly broadcast some public information (commitment) needed for verification.
Each $P_i \in P$ selects randomly in $GF(q)$ a polynomial $f_i(z)$ such that:

$$f_i(z) = x_i + \sum_{l=1}^{t-1} a_{il}\, z^l (modq) \tag{8}$$

For each $j \in \{1,2,..,n\}$, P_i computes the share:

$$s_{ij} = f_i(u_j)\ (modq) \tag{9}$$

and unicasts to node j an encrypted and signed message containing s_{ij} and h_i . For this purpose, P_i and P_j have to use their Off-line Certificates to establish a secure channel. In addition, P_i broadcasts, for $l \in \{1,2,..,t-1\}$, the values:

$$A_{il} = g^{a_{il}}(modq) \tag{10}$$

Step3. Verification
Thanks to homomorphism property, each node $P_i \in P$ can verify if the share s_{ji} received from P_j is correct by checking, for $j \in \{1,2,..,n\}$:

$$g^{s_{ji}} = \prod_{l=0}^{t-1} (A_{jl})^{{u_j}^l}\ (modp) \tag{11}$$

Note that $A_{j0} = h_j = g^{x_j}\ (modp)$.
If this holds, P_i computes its share s_i (partial private key) by:

$$s_i = \sum_{j=1}^{n} s_{ji}\ (modq) \tag{12}$$

and then signs the MANET's public key h with its partial private key, otherwise it broadcasts a warning and publishes a signed s_{ij}.

c) *Assigning a secret share to new node*

To provide the On-line Certificate services, a new joining node must get a secret share (i.e. a partial MANET's private key). In our proposal, we suggest to authenticate the new joining node by its {Joint Free_IP Pool and Public Key Certificate} (see 3.2) before delivering it a secret share.
In doing so, we adopt the distributed algorithm proposed by [7].
Given a subset $Q = \{P_{i_1}, P_{i_2}, \dots, P_{i_t}\}$, where $(i_1 < i_2 < \dots < i_t) \in \{1,..,n\}$, a new node P_{new} with identity u_{new} can compute its partial private key s_{new} by:

$$s_{new} = \sum_{l=1}^{k} B_l(z = u_{new})\, s_{i_l} \qquad (13)$$

The values $B_l(u_{new})s_{i_l}$ are computed, signed and unicasts to node P_{new} by each node $P_{i_l} \in Q$. As $B_l(u_{new})$ are publicly known, a random shuffling scheme was proposed to keep s_{i_l} as a secret to only its owner P_{i_l}. In the shuffling technique, a random nonce is securely exchanged between any two servers in Q. The node with small u_{i_l} chooses and treats the nonce as negative while the other side treats it as positive.
Finally, each server will have (***t****-1*) nonces that it adds to the value $B_l(u_{new})s_{i_l}$ to obtain a shuffled partial secret share (see [7] for details).

d) *Threshold Digital Signature*

Our scheme requires secure threshold signatures. In a (***t,n***)-threshold signature scheme, a legitimate node wishing a signature from the Network Authority for any given valid message m, must collect at least k verifiable partial signatures from any subset $Q \subset P$ $(|Q| = \boldsymbol{t})$. A partial signature is produced by each server by means of its partial Network's private key. A full signature on message m is built by combining all these $\boldsymbol{t}$ partial signatures resulting in the same regular signature that will be produced with the Network's private key. Basically, a regular signature scheme is a triple of efficient randomized algorithms:

- A key generation algorithm which outputs a pair (s,h) of private/public keys.
- A signing algorithm which on input a message m and the private key s, it outputs $\widetilde{m}$ a signature of the message m.
- A verification algorithm which on input a message m, its signature $\widetilde{m}$ and the public key h, it checks whether this signature has been produced with the private key s.

In a (***t,n***)-threshold signature scheme, the key generation process is achieved by a distributed key generation protocol such as that described in (3.1.1-(b)), the threshold signature is produced also by a distributed signature protocol, whereas the verification algorithm is the same as that in the regular signature scheme. With regard to the threshold signature protocol, a variety of discrete log based schemes have been proposed [13] including Nyberg-Rueppel or ElGamal- like and Elliptic Curve threshold digital signatures. In this work, we do not specify any particular implementation, we assume robust and unforgeable threshold signature scheme with verifiable partial signatures.
Note that in our scheme, each node must hold:

* On one hand, a valid share of the MANET's private key with which it will be able to participate in providing only threshold primitives mainly threshold signatures for the On-line certificates and partial secret shares for new joining nodes.
* On the other hand, a pair of private/public keys approved by the On-line Certification Authority, with which it will be able to carry out regular cryptographic primitives such as encryption, decryption, and regular signature.

We assume in the remainder of the paper that the threshold cryptosystem is implemented by a DPKI (Distributed Public Key Infrastructure) module including all necessary components described above (Fig. 1).

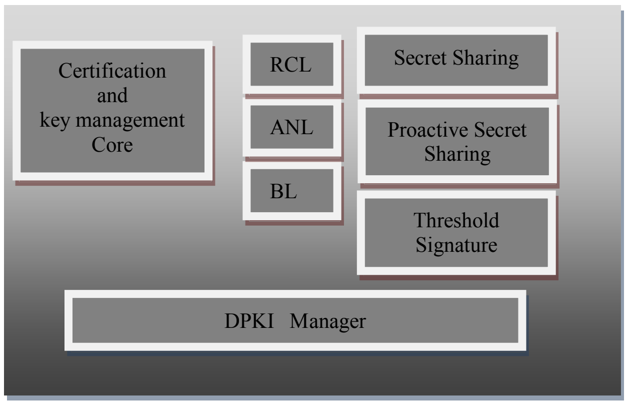

Fig. 1 Distributed PKI Module (Functional Blocks)

3.2. Joint Free_IP Pool and Public Key Certificate

IP address must be bound to node's identity [1]. To do so, we have two possibilities:

– Separate Certificates: one certificate for the IP address and another one for the public key.
– Joint Certificate: in which both the IP address and the public key are tied to node's identity.

However, in order to be able to take an active part in the network, a new node must hold among other things:

– A valid IP address
– A Free_IP Pool
– An On-line public key Certificate
– A MANET's private key share

That means that in case of Separate Certificates a new node having received already one of the two certificates must wait for the other. Consequently, it seems to be obvious to use a Joint Certificate for efficiency purposes (communication overhead and latency).

In our scheme, each authenticated node in the MANET must obtain a {Joint Free_IP Pool and Public Key Certificate} signed by the On-line Certification Authority. This {Joint Free_IP Pool and Public Key Certificate} contains, in addition to information provided by the X.509 certificate, the free_IP pool and the IP addresses of the ***t*** signers which have produced the threshold signature of this certificate. The knowledge of free_IP pool leads to that of node's IP address because every time a Requester gets a block of free IP addresses, it assigns itself the first address and the remaining addresses will form the free_IP pool. Hence, the IP address, the Free_IP pool and the public key of any participating node in the MANET are tied to its identity by means of the {Joint Free_IP Pool and Public Key Certificate}.

3.3. Threshold based buddy system IP Address Allocation Model

We consider a standalone MANET. DDAAP is known as a stateful protocol with no replication of state information. The IP address is assigned exclusively by an Allocator from the MANET. This helps to manage the address space and prevents malicious nodes to freely take part in the MANET. Furthermore, nodes in the MANET hold disjoint free_IP pools. The scheme we propose here is quite similar to DDAAP. We modify slightly DDAAP to take into account security. The new protocol is called SHEAP (**S**ecured t**H**r**E**shold based distributed dynamic adddress **A**ssignment **P**rotocol). First, the MANET is bootstrapped by a coalition of at least ***t*** nodes (called the ***t*** Bootstrapping nodes) instead of one node (Fig. 3). This avoids any single point of failure. Secondly, when a new node wants to join the MANET, a mutual pre-authentication must take place between this IP address Requester and the auto-configuration servers (clarified below). These authentications are performed using off line Public Key certificates. Thirdly, when a Requester has found and chosen an Allocator with non empty free_IP pool, before allotting the second half of free_IP pool to the Requester this Allocator must request two new online certificates: {Joint Free_IP Pool and Public Key Certificate}, one for the Requester and another for itself. So, for each new requested certificate it must collect ***t*** partial signatures to combine them in a threshold signature. To do so, the Allocator performs an expanding ring search; the hop count is initiated to one. Note that the Allocator can be one of the ***t*** signers. Consequently, the mutual pre-authentication of the Requester is done with the Allocator and the (***t***-*1*) others signers. Before delivering partial signatures, the (***t***-*1*) signers look if the Allocator is not in Black List nor its certificate is revoked by asking the t signers of its old {Joint Free_IP Pool and Public Key Certificate}. When the new {Joint Free_IP Pool and Public Key Certificate} are signed for the Requester and the Allocator, the new Allocator's certificate is published by these signers to the ***t*** signers of its old one.

3.4. Assumptions

i. In our scheme nodes may enter or leave the MANET dynamically, the only requirement is that a node must hold a valid and unrevoked certificate. We assume an existing Off-line Certification Authority for signing for any legitimate node that will participate in the MANET an Off-line {Public Key Certificate}. To join the network, each node must hold its Off-line {Public Key Certificate} and the public key of this Off-line Authority to be able to verify the validity of any Off-line {Public Key Certificate}. If a node joins the network for the first time or if its {Joint Free_IP Pool and Public Key Certificate} has expired, it must use its Off-line {Public Key Certificate} to be authenticated and to get hence a {Joint Free_IP Pool and Public Key Certificate} for participating actively in the network. The node will be accepted if its Off-line {Public Key Certificate} is valid and not revoked. If a node has leaved the MANET for a short time and wishes to re-enter, it will use its {Joint Free_IP Pool and Public Key Certificate} under condition that it is accepted (not expired and not revoked).

ii. Each node in the MANET needs to be able to verify at any time whether a p ublic key is revoked, hence a revocation scheme is needed within the MANET. Nodes need to be able either to revoke their own public keys or to revoke the public keys of malicious/compromised nodes, which can be achieved by the so-called *accusation schemes* [14] using threshold signatures. We assume that such scheme is implemented within the MANET and that each node holds in its DPKI module three tables containing respectively revoked public keys, accused nodes, and malicious nodes. We refer to these tables respectively as ***RCL*** (Revoked Certificates List), ***ANL*** (Accused Nodes List), and ***BL*** (Black List). The ***RCL*** indicates here the list of individual certificate revocations. Each revocation must be signed by the On-line Certification Authority. The ***ANL*** contains nodes who are accused by a number of honest nodes lower than the threshold. If the number of accusations signed by different nodes has reached the threshold, depending on the security policy either the accused node will be considered as malicious for the rest of time (Black List) or just its certificate will be revoked (Revoked Certificates List).

iii. MANETs are not guaranteed to consist of certain number of nodes all the time, especially at the time of Network bootstrapping. We make here two assumptions: First, we assume that at any instant during the lifetime of the MANET there will be at least ***t*** honest nodes (not compromised) to serve correctly any new joining node. Secondly, we assume that at the birth of the MANET there will be in a same neighborhood a subset of at least ***t*** nodes able to initialize the network using exclusively local links (i.e. one-hop c ommunications). The acceptability of these assumptions depends on the threshold value ***t***. In general, the choice of this parameter must achieve a tradeoff between security and QoS [8].

iv. We also assume that a proactive secret sharing scheme [15] is implemented in the case of a long-lived MANET for refreshing periodically the secret shares, to defend against repeated attacks conducted by strong, dynamic and determined adversaries.

4. SHEAP Description and Specification

We provide a *building block approach* based decomposition of the protocol SHEAP in modular components, each having its own specific functionality. The properties of these functional components and their interaction define the overall behavior of the protocol. The SHEAP module contains the following blocks (Fig. 2):

- Neighbors Discovery: this block allows a new arrived node to discover its one hop neighbors.
- MANET Bootstrapping: used for MANET initialization.
- IP Address Requesting: this block allows the new node to choose an allocator to perform for it the IP Address allocation.
- IP Address Allocation Service Handler: it is the block responsible of the collect of threshold signature for the {Joint Free_IP Pool and Public Key Certificate}.
- Partitioning and Merging Handler
- Node Departure Handler

In this paper, we have addressed the three blocks concerning configuration of a new joining node. The remaining blocks are left for further work.

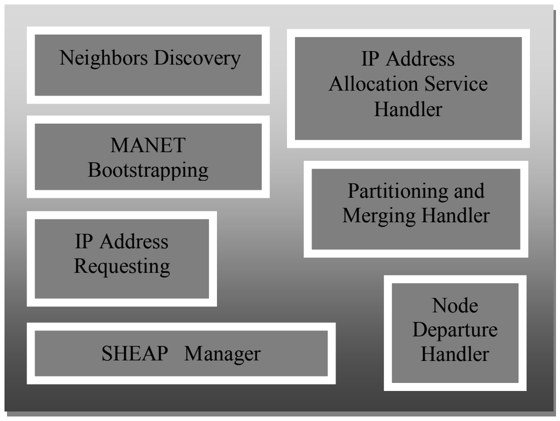

Fig. 2 SHEAP Module (Functional Blocks)

4.1. Node's states

We define in our scheme two node's states:

- ***Unconfigured*** node: any node wishing to join the MANET, and which has no online certificate: {Joint Free_IP Pool and Public Key Certificate}.
- ***Configured*** node: any node within the MANET holding:
  - ✓ a {Joint Free_IP Pool and Public Key Certificate}
  - ✓ an offline {Public Key Certificate}
  - ✓ The public key of the Off-line Certification Authority
  - ✓ A share of MANET's private key
  - ✓ MANET's public key
  - ✓ ***RCL***, ***ANL*** and ***BL*** lists

Such nodes should be able to participate actively in the network.

4.2. Neighbors Discovery

4.2.1. Messages and Timers

- ***NeighborsDiscoveryRequest:*** this message is used by an ***Unconfigured*** node in the *MANET* to discover its one-hop neighbors.
  The message contains the Requester's offline {Public Key Certificate}, the bit configuredFlag set to 0 (***Unconfigured*** node) and the signature of the message with the offline {Public Key Certificate}.

- ***NeighborsDiscoveryReply:*** this message is a reply to the ***NeighborsDiscoveryRequest*** message.
  There are two type of this message, one with the bit configuredFlag set to 0 and the other with the bit configuredFlag set to 1. In the first case, the responder is in state ***Unconfigured,*** the message will contain its offline {Public Key Certificate}, the bit configuredFlag set to 0, its one-hop Neighbors List L_{NEIG} and the signature of the message with the offline {Public Key Certificate}. In the second case, the responder is in state ***Configured,*** the message will contain its {Joint Free_IP Pool and Public Key Certificate}, its offline {Public Key Certificate}, configuredFlag set to 1 and the signature of the message with the offline {Public Key Certificate}.

- ***NeighborsDiscoveryTimer:*** this timer is rescheduled each time the ***NeighborsDiscoveryRequest*** message is broadcast. It permits to the Requester to consider responses in a finite time.

4.2.2. Discovery Process

A new node wishing to join a MANET broadcasts locally and periodically a ***NeighborsDiscoveryRequest*** message and starts the ***NeighborsDiscoveryTimer***.
According to the state of the recipient, the response to this message will be one of the two ***NeighborsDiscoveryReply*** messages described above.

a) Recipient in state ***Unconfigured***
After checking the signature, the recipient (say X_i) saves the Requester's identity in its one-hop Neighbors List $(L_{NEIG})_i$ and its offline certificate in the Certificate List, and replies by a ***NeighborsDiscoveryReply*** message (type: configuredFlag=0).

As long as the ***NeighborsDiscoveryTimer*** has not expired, the Requester stores in its cache all its one-hop neighbors and their L_{NEIG} and their offline certificates.
On timer's expiration, the Requester checks the neighbors lists intersection (B_N *the* set of Bootstrapping Nodes):

$$B_N = \cap_{X_i \in \Omega} (L_{NEIG})_i \qquad (14)$$

where Ω is the set of all one-hop neighbors including the Requester itself.
If $|B_N| \geq \boldsymbol{t}$ then the Requester initiates MANET Bootstrapping, otherwise it clears its cache and performs a new Neighbors Discovery broadcasting.

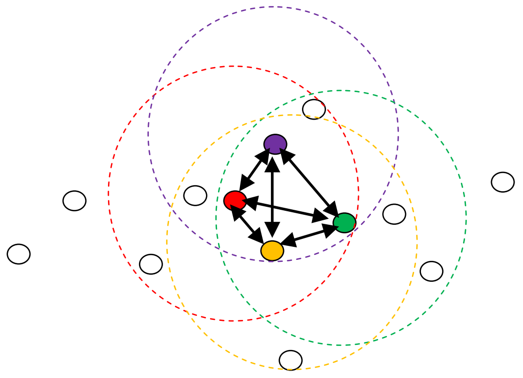

Fig. 3 MANET Bootstrapping
with four Bootstrapping nodes (threshold ***t***=4)

b) Recipient in state ***Configured***
After checking the signature, the recipient replies by a ***NeighborsDiscoveryReply*** message (configuredFlag=1). Note that this message is prior, so it m ust be treated while discarding messages with configuredFlag=0. On receiving this message, the Requester concludes that a MANET exist and has to start the IP Address Requesting.

4.3. Configuration of newly arrived node

4.3.1. Messages and Timers

- ***AllocatorChosen :*** this message is used by the Requester to inform the Allocator that it has been chosen. The message contains the Ip address of the Allocator, a R equester's NONCE encrypted by the public key of the Allocator and the signature of the message.
- ***AllocationSuccess:*** this message is a reply to ***AllocatorChosen*** message when the allocation

process has succeeded. This message contains the Requester's NONCE encrypted by the public key of Requester, the {Joint Free_IP Pool and Public Key Certificate} of the Requester and the signature.

- ***AllocationFail***: this message is sent by the Allocator to the Requester to inform it that the allocation process has failed.

- ***JointCertificateRequest:*** this message is broadcast by the Allocator to get for the Reqiuester. It contains the Requester's offline {Public Key Certificate} and the Allocator's {Joint Free_IP Pool and Public Key Certificate} and the signature.

- ***RevokedCertificateAsk***: this message is unicast to the co-signers of the Allocator's {Joint Free_IP Pool and Public Key Certificate} to ask them if this certificate is revoked or no.

- ***RevokedCertificateReply***: this message is a reply to ***RevokedCertificateAsk*** message.

- ***NewCertificatepublication:*** this message is sent to the co-signers of the old Allocator's {Joint Free_IP Pool and Public Key Certificate} to inform them that the Allocator has a new certificate and that the old one is revoked.

- ***AllocationTimer:*** this timer is rescheduled each time an ***AllocatorChosen*** message is sent.

4.3.2. Configuration Process

When the Requester receives ***NeighborsDiscoveryReply*** messages with configuredFlag set to 1, it chooses the neighbor with the largest free_IP pool, sends an ***AllocatorChosen*** message to the selected neighbor and start the ***AllocationTimer.***
If the Requester receives an ***AllocationFail*** message or if the timer has expired and the Requester has not received an ***AllocationSuccess*** message, then it restarts the Neighbor Discovery process.
When the Allocator receives the ***AllocatorChosen*** message, it looks for the co-signers to perform threshold signature. If it fails to find the sufficient co-signers in one-hop, it performs an expanding ring search (Fig. 4).
If this process failed, then an ***AllocationFail*** message is sent to the Requester. Otherwise, the Allocator combines the partial signature to get the requested {Joint Free_IP Pool and Public Key Certificates} and send an ***AllocationSuccess*** message to the Requester.

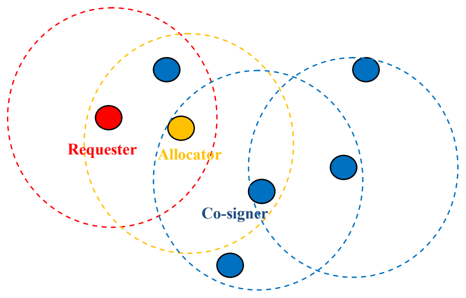

Fig. 4 IP address allocation. Expanding ring search for partial signatures (threshold t=6, the Allocator and 5 co-signers)

5. Performance Evaluation

We implemented the protocol SHEAP in the NS-2 simulator with the CMU mobility extensions [16]. To do so, we created a new C++ agent SHEAP inheriting from Agent class (Fig. 5). We used the RSA-2048 algorithm for ordinary signature and the ECDSA-233 algorithm for threshold signature.

```
int hdr_sed_daap::offset_;
static class SHEAPHeaderClass : public PacketHeaderClass {
public:
SHEAPHeaderClass() : PacketHeaderClass("PacketHeader/ SHEAP",
sizeof(hdr_all_ sed_daap)) {
bind_offset(&hdr_ sed_daap::offset_);
}
} class_ SHEAP _hdr;
static class SHEAPclass : public TclClass {
public:
SHEAPclass() : TclClass("Agent/ SHEAP") {}
TclObject* create(int argc, const char*const* argv) {
assert(argc == 5);
return (new SHEAP((nsaddr_t) Address::instance().str2addr(argv[4])));
}
} class_ SHEAP;
```

Fig. 5 TCL hooks: C++ Code extracted from SHEAP implementation

The efficiency of the protocol is evaluated using the following metrics:

Latency for Dynamic IP Address Assignment: This metric represents the average delay for a newly arrived node to obtain an "On-line Joint Pool address and Public Key Certificate" . This includes all possible delays caused by the signature process, timeouts and messages exchanges.

Communication Overhead: This metric represents the number of SHEAP packets transmitted during the Dynamic IP Address Assignment process.

5.1.1. Simulation parameters

Table 1 summarizes the different parameters set for our experiment simulation.

WirelessChannel Mac type	Mac/802_11
Phy/WirelessPhy frequency	2.4 GHz
Phy/WirelessPhy bandwidth	11 Mbps
Wireless range	250 m
Routing protocol	AODV/DSDV
Mobility Model	random waypoint mobility model
Topology	variable
Threshold ***t***	3, 5, 8
Simulation time	60 s

Table 1 Simulation parameters

5.1.2. Simulation scenarios

The following sets of simulation were performed.

a) Varying Node Population: the effect of MANET size on latency and communication overhead is studied to evaluate the behavior of the protocol on scalability.

Table 2 gi ves the values of node population for these simulations:

Number of nodes	Area x=y (m)	Density (nodes/km2)
50	408	300
100	577	300
150	707	300
200	816	300
250	912	300
300	1000	300

Table 2 Node population values

The simulations were performed for different values of threshold ***t***. No motion was applied in this scenario.

b) Varying network density: We fix the number of nodes at 300 nodes while the density is varied from 10 to 300. Table 3 resumes the chosen values for these simulations:

Number of nodes	Area x=y (m)	Density (nodes/km2)
300	5477	10
300	2450	50
300	1732	100
300	1414	150
300	1224	200
300	1095	250
300	1000	300

Table 3 Network density values

c) Varying network mobility: the behavior of the protocol SHEAP is analyzed with respect to node speed for different values of threshold ***t***. The topology area is set to 1000 m X 1000 m with 300 n odes ensuring a density of 300 n odes/km2. The node speed is varied from 0 to 100 m/s and pause time is set to 0:

Setdest –v2 –n 300 –s 1 –m (nodespeed) –M (nodespped) –t 60 –P 1 –p 0 –x 1000 –y 1000

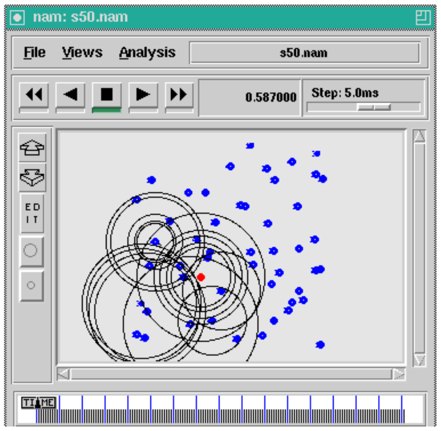

Fig. 6 Dynamic IP Address Assignment with SHEAP implemented, Illustrated by Network Animator. Number of nodes: 50

5.1.3. Simulation results

Each data point represents an average value of seven runs with the same settings, but different randomly generated topology and mobility scenarios.

a) Impact of Node Population: As shown by (Fig. 7), the latency increases sub-linearly with increase in network size. It was observed for instance for a MANET with 50 nodes a latency of about 0.4 s for threshold ***t***=3 and 0.7 s for ***t***=8. Compared to the results provided by [4], our security improvement increases latency only by 0.1s. In most time, unicast messages are used and the broadcast is performed locally. Likely, communication overhead evolves sub-linearly with number of nodes [Fig. 8]. The number of emitted SHEAP packets for a MANET size of 300 nodes and threshold ***t***=8 was less than 100, which is quite low. Thus, the protocol SHEAP preserves scalability.

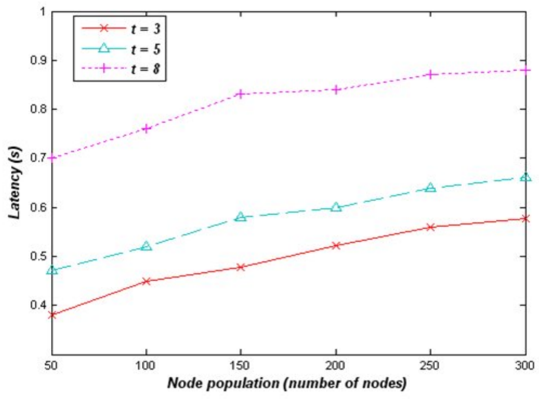

Fig. 7 Latency vs Node population

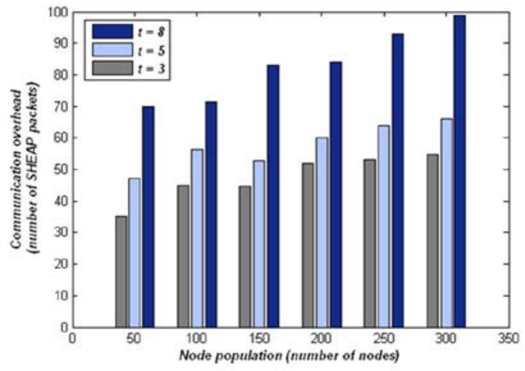

Fig. 8 Communication overhead vs Node population

b) Impact of Network Density: the network density influences directly on the number of nodes in the neighborhood of the Requester as well as of the Allocator. We observe (Fig. 9) an increasing in latency when the density is low. In this case, the number of one-hop neighbors is low, leading to many expanding ring searches particularly when the threshold is high.
The latency also increases when the density is high because of the higher number of messages exchanged causing collisions. This is confirmed by (Fig. 10) reporting the increasing of communication overhead with respect to density. Note that the mean value of latency remains below one second which practical.

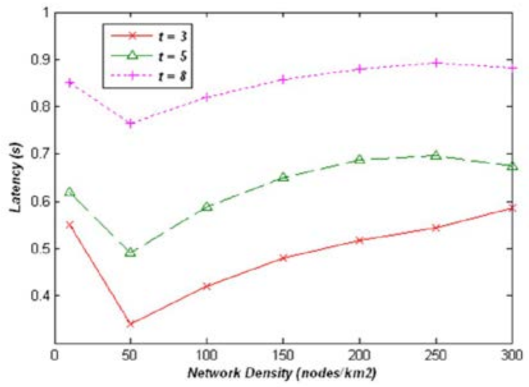

Fig. 9 Latency vs Network Density

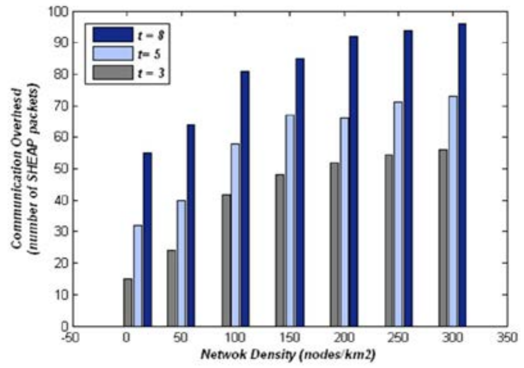

Fig. 10 Communication overhead vs Network Density

c) Impact of Mobility: the mobility has an impact on density and precisely on number of nodes in the neighborhood of a particular node. We intentionally make nodes moving from Requester and Allocator to create a low density around them. It was observed that both latency (Fig. 11) and communication overhead (Fig. 12) increase as node speed increases, but the values reached are not shocking.

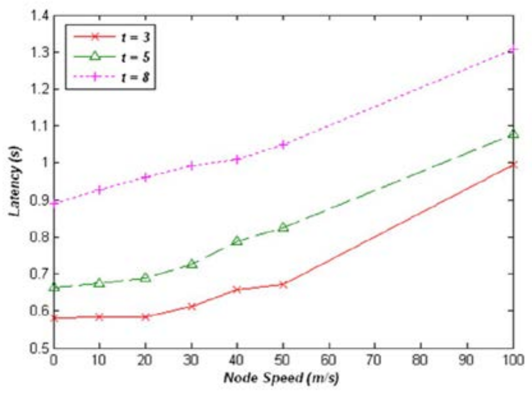

Fig. 11 Latency vs Network Mobility

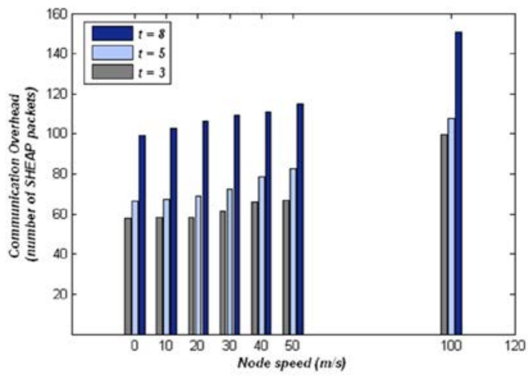

Fig. 12 Communication overhead vs Network Mobility

6. Conclusion

The paper proposes the new protocol SHEAP which provides security for MANET stateful buddy system based dynamic address allocation protocols. Our solution relies on a distributed PKI and a threshold certified cooperative address allocation scheme. The new proposed security mechanisms thwart all possible imaginable attacks on auto-configuration in MANET. The cost of the security improvement brought by SHEAP is a low increasing in latency and communication overhead, which remains acceptable. Simulation results show that the protocol SHEAP incurs practical values of these metrics as well as the insecure protocols proposed up to now.

References

[1] A. Abdelmalek, M. Feham and A. Taleb-Ahmed. On Recent Security Enhancements to Autoconfiguration Protocols for MANETs: Real Threats and Requirements. International Journal of Computer Science and Network Security, Vol.9, No.4, PP.401–407, April 2009.

[2] A. Misra, S. Das, A. McAuley, and S. K. Das. Sun. Autoconfiguration, Registration and Mobility Management for Pervasive Computing. IEEE Personal Communications, vol. 08, Issue 04, Aug. 2001.

[3] M. Mohsin and R. Prakash, "IP address assignment in mobile ad hoc networks," in *Proc. IEEE MILCOM*, pp. 856-861, Sept. 2002.

[4] M. Thoppian and R. Prakash, "A distributed protocol for dynamic address assignment in mobile ad hoc networks," IEEE Transactions on Mobile Computing, pp. 4–19, 2006.

[5] K. Hoeper and G. Gong. Models of Authentications in Ad Hoc Networks and Their Related Network Properties. Technical Report 2004, Centre of Applied Cryptographic Research. Available at http://www.cacr.math.uwaterloo.ca/

[6] A. Baayer ,N. Enneya , M . El Koutbi. A Recent Survey on Key Management Schemes in MANET. 3rd International Conference on Information and Communication Technologies: From Theory to Applications, 2008. ICTTA 2008.

[7] Kong, P. Zerfos, H. Luo, S. Lu, and L. Zhang. Providing Robust and Ubiquitous Security Support for MANET. In IEEE International Conference on Network Protocol*s*, pages 251.260, Nov. 2001.

[8] M.A Azer, S.M. El-Kassas and M.S El-Soudani. Threshold Cryptography and Authentication in Ad Hoc Networks Survey and Challenges. Second International Conference on Systems and Networks Communications, 2007. ICSNC 2007. Volume , Issue , 25-31 Aug. 2007
http://www.cacr.math.uwaterloo.ca/

[9] G. Di Crescenzo, G. Arce and R. Ge. Threshold Cryptography in Mobile Ad Hoc Networks. In Springer Berlin / Heidelberg editor, Security in Communication Networks, volume 3352 of Lecture Notes in Computer Science, Springer 2005

[10] Y. Desmedt and Y. Frankel, "Threshold cryptosystems",In Gilles Brassard, editor, Advances in Cryptology –*CRYPTO '89*, volume 435 of Lecture Notes in Computer Science, Santa Barbara, California, U.S.A., August 1989. Springer-Verlag, pp. 307–315.

[11] A. Shamir. How to Share a Secret. *Communications of the ACM*, 22(11):612–613, 1979.

[12] T. Pedersen, *A threshold Cryptosystem without a Trusted Party*, in Proc. of Eurocrypt 91

[13] M. Hwang and T. Chang. Threshold Signatures: Current Status and Key Issues. International Journal of Network Security, Vol.1, No.3, PP.123–137, Nov. 2005

[14] C. Crépeau and C.R. Davis. A Certificate Revocation Scheme for Wireless Ad Hoc Networks. Proceedings of ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN '03), pp.54-61, 2003.

[15] A. Herzberg, S. Jarecki, H. Krawczyk, M. Yung, Proactive secret sharing or: How to cope with perpetual leakage, *CRYPTO'1995*, LNCS 963, pp. 339-352, Springer-Verlag 1995

[16] The Network Simulator manual, The NS2 homepage http://www.isi.edu/nsnam/ns