

# Identification in Ad hoc Networks

Frank Kargl, Stefan Schlott, Michael Weber  
 Media Informatics Department, Ulm University, Germany  
 Email: {givenname.surname}@uni-ulm.de

**Abstract**—In this paper we address the concept of how nodes in Mobile Ad hoc Networks (MANETs) should be identified and authenticated. After discussing related works and the concept of identities and identifiers in MANETs, we present the MANET-ID system which can be used to reliably identify nodes in an ad hoc network with properties like uniqueness, irreversible ties with the identified object, immutability throughout the lifetime of the object and non-transferability. We also evaluate the overhead and discuss the advantages and disadvantages of our approach. We conclude the paper with a summary and outlook to future work.

## I. INTRODUCTION

Multi-hop Ad hoc networks are a promising paradigm for providing connectivity to nodes in a large number of exciting new application areas. In order to establish that connectivity, ad hoc networks rely heavily on the cooperation of all participating nodes. A lot of previous publications document the harmful effects that selfish or malicious nodes create in MANETs [1].

Our work focuses on creating a complete security framework to address these problems. We have identified three different aspects that need to be solved in order to secure MANETs:

**Identification:** A first step towards security in MANETs is the possibility to uniquely identify nodes. Without that many additional security mechanisms will fail, because nodes can spoof the identity of other nodes or can create an arbitrary number of additional identities. This allows a large number of potential attacks that can be prevented only by introducing unique IDs. In the rest of the paper we will discuss, how such an identification mechanism may be realized given the special constraints of MANETs.

**Secure Routing:** Like explained in [1], an unprotected routing protocol can be attacked by means of e.g. Black Hole routing, etc. Therefore, we need a routing protocol that protects the integrity of the contained routing data. This is usually accomplished by some cryptographic mechanism.

**Prevention of Selfish Nodes:** Whereas a secure routing protocol detects only active attacks (i.e. attacks, where an attacker alters the usual behavior of the routing protocol), it is not able to detect selfish behavior, where a node denies participation in routing activities. So we need some way of preventing such nodes from abusing the network without contributing own resources. There are two main ideas, how this can be accomplished: either by giving an incentive to cooperating nodes or by detecting and punishing selfish nodes. Such a punishment may be the exclusion of a selfish node from the network. Especially in the later case it is evident, that in order to exclude a selfish node, it needs to be identified without doubt. Otherwise wrong nodes may be accused or excluded selfish nodes may return to the network with a freshly generated id.

A lot of research has been done in each of these fields. But many approaches fail to recognize the dependencies between all the different aspects. So our approach is to realize an integrated architecture that respects these dependencies and delivers a complete security solution for MANETs. We call this system *Security Architecture for Mobile Ad hoc Networks* or *SAM* [2].

SAM consists of three components. MANET-IDs are the component responsible for ensuring a unique identification for each MANET device. These IDs are then used by the Secure Dynamic Source Routing protocol to authenticate nodes participating in the routing process [3]. Finally there is the Mobile Intrusion Detection System that uses different sensors in order to detect selfish behavior like nodes not forwarding packets as they are supposed to [4]. As these nodes may then be excluded from further participation in the network, it is important to have a secure authentication of nodes in order to prevent false nodes from being excluded or selfish nodes from returning to the network with false identities.

So using such a reputation system, there is a clear relationship between the trust that a node can put in any other and the reliability by which this other node can be authenticated. Wikipedia defines

authentication as [5]

... the process by which a computer, computer program, or another user attempts to confirm that the computer, computer program, or user from whom the second party has received some communication is, or is not, the claimed first party.

This raises another important question: When building an authentication system for ad hoc networks, what are we going to authenticate? The user to which certain nodes belong? The nodes themselves? Or one among many programs running on a node?

In this paper we will first look at related work on this issue, then discuss the question how an identity and authentication mechanisms for secure ad hoc networks should be constructed and finally present our own solution that fits into the requirements of SAM.

## II. RELATED WORK

An early work addressing authentication between mobile devices is called "The Resurrecting Duckling" [6], [7] by Frank Stajano and Ross Anderson. Here two devices establish a trusted relationship by means of a physical contact where a secure symmetrical key is exchanged. Obviously, this solution is rather restricted: it is only useful for single-hop communication with a master-slave relationship, requires an initial physical interaction and does not scale well. Given  $n$  interacting devices, each device needs to manage  $\binom{n}{2}$  pairwise keys. The big benefit of Stajano's solution is the fact that no asymmetrical cryptography is needed at all. By using physical interaction between two devices, it is implicitly clear that devices are authenticated.

### A. CA-based Approaches

Another group of projects uses asymmetric cryptography for authentication. Here you need to verify the relationship between a public key and the identity of the key holder (which can be a user, a device or a program), the rest of the authentication is a simple signature check. In regular networks a trusted third party (TTP) is used for connecting the public key to an identity. This TTP is then called certification authority (CA) and the statements about a key and an identity are called certificates.

Using the straightforward CA approach has a number of drawbacks when being applied to ad hoc networks [8], [9], e.g. the CA needs to be reachable

for new certificates, certificate renewal or revocation which cannot be guaranteed in a general MANET. If the CA is part of the MANET the question arises which node should take this responsibility and what happens if this node leaves the network or gets compromised.

So a number of proposals were made how to leverage these issues. In [10] the authors suggest a distributed certification authority for MANETs, which uses threshold cryptography for signing keys.  $n$  nodes within the MANET form the distributed CA. For CA operations like signing keys at least  $k$  of these nodes need to cooperate. If an attacker succeeds in compromising a maximum of  $k - 1$  nodes, this brings him no benefit and he gains no additional knowledge.

Some details remain open here, e.g. how the partial keys of the CA nodes should be renewed periodically. The authors of [9] address some of these issues, e.g. what nodes should be selected to become CA nodes. Furthermore they discuss a mechanism called  $\beta$ -multicast to efficiently reach enough of the CA nodes so that the threshold cryptography will work. Depending on the number of known and valid routes to CA servers, the CA requests are either sent as a number of unicasts or flooded as broadcast. Whereas the former solutions differentiate between CA and non-CA nodes, the authors of [8], [11], [12] describe, how a completely distributed CA can be designed. Here all nodes of a MANET participate in the CA. Here a node wanting e.g. a signature can simply pick  $k$ <sup>1</sup> random nodes in the network.

All these distributed CA schemes do not address the issue, *when* the CA should issue a certificate and what the meaning of this certificate should be. Only [9] suggests that signatures should be issued based on some "external observations". So without going into any details, the authors suggest that the certificate not only binds the public key to a fixed identity, but also expresses some kind of trust in the correct operation of the node. In contrast to that, we will later give a precise description of the semantics of a signature in MANET-IDs.

The concept of a distributed CA seems appealing at first, as it positions the CA inside the MANET while at the same time protecting the CA secrets from attacks. But it has also a number of severe drawbacks: first there is no concept how such a CA should be bootstrapped when the first few nodes meet to form a MANET. So there is a need for a

<sup>1</sup>The authors fail to realize that selecting  $k - 1$  nodes is also sufficient, as the originating node also has a part of the shared secret.

TTP that initializes the first distributed CA nodes. Another disadvantage is the threshold cryptography itself which is very bandwidth and CPU intensive.

Another open issue is how to select the parameters  $k$  and  $n$ , how to adapt them to changing network conditions and what to do when networks get partitioned or two separate network join.

So the approaches using distributed CAs are not really practical and a lot of questions still have to be solved.

### B. Web of Trust

Another approach for managing identities and trust in ad hoc networks is based on the "Web of Trust" idea which is e.g. the base of the popular PGP encryption software [13]. Here nodes can mutually sign their respective keys. By signing another persons key, a PGP user expresses his believe that the signed key actually belongs to the user the key claims to belong to. This concept is also called *validity* of a key. If another user believes this signature depends on the so called *trust*-level. Users can assign trust to other's keys, expressing their believe in the ability of another user to correctly apply signatures to other keys. The trust level may also imply transitive trust.

As an example, if user  $A$  uses his key  $k_A$  to sign the key  $k_B$  of user  $B$ , he expresses his believe that  $k_B$  actually belongs to  $B$ . The same applies when  $B$  signs the key of user  $C$ . Now if user  $D$  trusts  $A$  to create only valid signatures, he can validate the key  $k_B$  being actually the key of user  $B$  by checking the signature from user  $A$ . If he even trusts  $A$  transitively, then he can even validate  $k_C$  after validating key  $k_B$ .

As the Web of Trust works completely without infrastructure, it is a tempting idea to apply this to the domain of ad hoc networks like described in [14], [15]. One important problem is how to store and retrieve keys and associated signatures. The authors suggest a distributed storage scheme, where each node stores a portion of the overall keys/signatures used in the network. The *Shortcut-Hunter* or *Maximum Degree* algorithms are then used to locate and retrieve needed keys/signatures.

Again there are a number of problems arising here. In an initial phase of the network, when not many signatures have been issued, transitive trust links between keys can hardly been found, so nearly no keys can be validated. If a node cannot validate the key of its communication partner, both nodes cannot communicate securely.

An even more fundamental questions: when should users sign the keys of other users? This is a manual

and thus time consuming procedure that implies checking the other persons identity (e.g. by checking his passport). One can guess that many users will not take the time to issue many signatures or check the identity carefully. Many users will probably not understand the concepts behind all these processes and there role in this scenario anyway and can therefore not issue signatures reliably.

Finally the authors mix up the concept of validity of a key and trust in a key. In their system signing a key means both that the key actually belongs to the person it claims to belong to and that this person can be trusted to issue correct signatures<sup>2</sup>. There is very good reason that most systems like PGP strictly keep these concepts apart. Ultimately this approach assumes the honesty and consciousness of all partners receiving any signature from anybody. In this case simply distributing one shared key would be sufficient.

So applying the Web of Trust to ad hoc networks leaves a lot of open questions that need to be resolved before actually applying the technology in real networks.

### C. Crypto-based Identities and Identity-based Cryptography

Other approaches to verify identities directly relate the identities of nodes to their cryptographic keys. In [16] the authors apply the concept of crypto-based identities (CBI) and crypto-based addresses (CBA) like described in [17], [18] to the domain of ad hoc networks. Here identifiers and addresses are generated by applying hash functions to the public keys of nodes. When communication with a another entity, one can simply check whether the key and the pretended identity or address match. The authors of [16] use this to validate routes in ad hoc networks. Their routing protocol prevents nodes from modifying routing information or spoofing other's identities. There is no need for certificates or trusted third parties.

Unfortunately this system does not provide any information about the communication partner. The only information contained is that this other entity knows an asymmetric key pair. There is no place for linking names or similar information to the keys. Additionally it is very easy for nodes to create an arbitrary number of identities by simply creating new key pairs and deriving CBIs or CBAs.

<sup>2</sup>and that the persons this person gives signatures to can also be trusted ... etc.

Identity-based cryptography (IBC) goes the opposite way. A binary representation of an identity is taken as the public key of an asymmetric cryptography scheme. The IBC algorithm then creates a matching private key. Nodes can simply check if the identity and the public key of other nodes match. If all nodes can create matching private keys, this scheme is of no practical use, because knowing an identity (aka public key) immediately leads to knowing also the private key. So practical approaches include a trusted-third party that acts as a private key generation service (PKGS). This PKGS owns a secret that is needed for private key generation.

Based on an identity-based cryptography algorithm developed by Boneh and Franklin [19], Khalili e.a. apply this idea to ad hoc networks [20]. Using threshold cryptography, the PKGS function is distributed among MANET nodes.

The properties of their system are very similar to the distributed CA approaches discussed earlier. Instead of signing keys as the CA does, the PKGS generates private keys. This leads to the same problems that distributed CAs show in MANETs: the cryptographic scheme consumes a lot of CPU and communication resources, and there is a bootstrap problem when establishing a MANET.

When a node enters a network, the node has no secret key it can use to communicate with the PKGS nodes. In this case a node can only securely establish a secret key by direct, physical contacts with at least  $k$  PKGS nodes. This is clearly not acceptable for most users and scenarios. Finally it is not clear when the PKGS should issue a secret key and what steps are necessary to identify a user that requests a key for a given identity.

### III. IDENTITIES IN AD HOC NETWORKS

So all works discussed so far suffer from disadvantages. Often nodes can create new identities on their own, the properties of an identity are not clearly defined, and it is not clear what preconditions must be met before a CA, Web-of-Trust node, or a PKGS can issue a certificate or generate a private key. Before presenting our own identification solution for MANETs, we first want to address some basic questions.

#### A. Scenario

A lot of the requirements for authenticating nodes is derived from the scenario and requirements of the solutions used for MANET routing and security. We

assume a generic MANET scenario, where nodes are owned and operated by individual users with no common context or organizational structure. Arbitrary nodes can join together to form a MANET. In this paper we focus on identifiers for the routing layer of a MANET. On the application layer, certain applications will probably have different requirements and it makes sense to introduce additional "application layer identities".

The already mentioned SAM security architecture provides the SDSR routing protocol [3], which needs unique identifiers per node. Additionally each node needs a public/private key pair for signing and verifying messages. Further there is the MobIDS intrusion detection system [4] which is used for detection and exclusion of selfish or malicious nodes.

MobIDS requires unique identifiers where nodes are not able to create new identities on their own. Otherwise when one identity of a node gets excluded, it could simply generate a new one. Having a random amount of identities available also supports virtual collusion attacks. Finally, MobIDS needs a way of invalidating identifiers as a mean of permanent MANET exclusion.

#### B. Identities

Before constructing an identification system to fulfill these requirements, a number of questions describing the design space of identifiers need to be answered. We start with two definitions:

**Definition Identity:** The identity of an object is a *unique*<sup>3</sup> property that is *tied irreversibly to this single object*. The identity is *unchangeable* throughout the lifetime of this object and *cannot be transferred* to other objects.

Sometimes a large quantity of identical objects are constructed, that are indistinguishable at first. By attaching *identifiers*, these objects become unique and get their own identity.

**Definition Identifier:** An *identifier* is a property (or group of properties) which is suited to identify an object, i.e. doubtlessly determine its identity according to the properties defined above.

An example to explain these concepts are cars. When constructed in a factory, all cars of one model are (almost) identical. By attaching a unique vehicle identification number to the chassis, the car gets an identity that is

- 1) unique.

<sup>3</sup>also in the sense that an object cannot have an arbitrary number of different identities

- 2) tied irreversibly to this single object.
- 3) cannot be changed throughout the lifetime of this object
- 4) cannot be transferred to other objects.

Whereas traditional routing IDs usually need to be unique, the other criteria are not met. So for security reasons other forms of identifiers are needed. Questions arising include:

*What identity should those identifiers identify?* In our scenario it is possible to tie identifiers to users, mobile devices or network interfaces. On the routing layer of ad hoc networks it is reasonable to identify nodes. Identifying users leads to problems when devices are owned by multiple users, when devices are sold or get stolen. If e.g. a stolen device is abused and the identity invalidated by MobIDS, all other devices of an user would then also be unable to participate in the network. Having IDs per network interface means that a device own multiple identities which is also problematic.

*What property is used as an identifier?* If a device has some kind of "natural" device dependent unique ID (e.g. a processor serial number), this could be used as identifier. Otherwise generic identifiers (e.g. UUIDs) or cryptographic keys could be used. The advantage of using cryptographic keys as identifiers is that you do not need to tie the keys (which are needed anyway for signatures, etc.) to the identifiers by means of a e.g. a CA.

*How is the creation of new IDs being prevented?*

**Claim:** *If the creation of an identifier is done by a node alone, then creation of new IDs cannot be prevented.*

**Rationale:** Let  $\mathcal{A}(X)$  be the algorithm, which node  $X$  carries out for creation of its identifier  $ID_X$ . Besides  $X$  there are no other (external) arguments needed for running  $\mathcal{A}$ . The algorithm  $\mathcal{A}$  needs to be known to  $X$  to create  $ID_X$ . Consequently  $X$  can run the algorithm with another argument  $Y$  and gets  $\mathcal{A}(Y) = ID_Y$  as a result. If  $ID_X$  is a valid identifier, then also  $ID_Y$  is valid.

So some external entity needs to be involved in this process. As this external entity must be trusted not to collude with  $X$  to produce other IDs, we call this entity a *Trusted-Third-Party* (TTP). So without a TTP creation of new IDs cannot be prevented. This is why our system needs a TTP. The goal is to make the TTP as compatible with the MANET operation as possible.

*How are identities and addresses linked?* If you use strings, public keys or similar data as identifiers, you need a way to link them to the addresses used for

routing. This can be achieved by signatures of a TTP or by using the concept of crypto-based addresses presented in the related work section.

All these questions need to be answered when designing an identification and authentication system for MANETs. We have designed such a system called MANET-IDs which we will present in the next section.

## IV. MANET-IDs

### A. General Architecture

Each node that is supposed to work as a MANET node receives a public/private key pair  $(PK_X, SK_X)$  on construction time. The public key  $PK_X$  is used as the identifier or MANET-ID of node  $X$ :  $ID_X = PK_X$ . This identifier can and must be used in any MANET that supports the SAM security architecture. For routing purposes the node creates a crypto-based address from the MANET-ID by using a hash function:  $CBA_X = h(ID_X)$ . When using IPv6, the hash function should provide a 64 bit host identifier. The network identifier can be selected statically like suggested in [21].

As stated earlier, in order to prevent creation of new MANET-IDs by nodes, we need to introduce a TTP. This TTP is a certification authority (CA) signing the MANET-IDs:

$$cert = E_{SK_{CA}}(PK_X)$$

It is important to recognize that the signature does not bind any identifying information to the public key, as it is normally done with CAs. Instead the only purpose of this signature is to mark the key as valid for MANET use. Thus signatures can be created without any identity checking etc.

These certificates are also created together with the keys at production time of the device. We do not try to include the TTP in order to prevent the disadvantages of threshold cryptography or similar approaches. Instead our system tries to minimize the interaction of MANET nodes with the CA so that the MANET can work autonomously.

### B. Certificate Revocation

When nodes get compromised or misbehave in the MANET, the MobIDS intrusion detection system will detect this and send a complaint of this misbehavior to the MANET-ID CA. After receiving a certain amount of these complaints, the CA decides that the MANET-ID needs to be invalidated, excluding

it from further participation in any MANETs using SAM.

But how should nodes in the field be informed about invalid certificates? The classical solution in fixed networks makes nodes checking the validity online with the CA or uses certificate revocation lists distributed from time to time to participants. Both approaches are not usable in MANET scenarios. Online checks at the time of validation are obviously impossible when the MANET has no Internet connection. CRL lists of all invalid certificates can easily reach a size of multiple megabytes which can neither be stored on resource-constrained mobile devices nor transmitted regularly through the MANET.

We suggest another approach for efficient certificate revocation. The idea is based on the so called *Certificate Revocation System* by Silvio Micali [22] and has been adapted to ad hoc networks. For certificate revocation we extend the certificate by a serial number, a verifier field  $Y$  and an expiration date.

$$cert = E_{SK_{CA}}(PK_K, serial, Y, valid\_until)$$

We divide the validity period of the certificate  $T$  into  $n$  sections of duration  $t$  where  $t = T/n$  (see figure 1). When issuing a certificate, the CA creates a hashchain.  $Y = Y^n$  is the end of this hashchain where  $Y^{i+1} = h(Y^i)$  using a cryptographic hash function  $h$  (e.g. SHA-1). The randomly chosen start value  $Y_0$  is kept secret by the CA server. Whenever a node presents its certificate to other nodes, it also needs to supply the verifier that is valid for the current section  $i$   $V_i = H^{n-i}(Y_0)$ . The other nodes then verify, whether

$$h^i(V_i) = h^i(h^{n-i}(Y_0)) = H^n(Y_0) = Y$$

If this is true, the verifier is valid, otherwise it is invalid. When a new section  $i+1$  starts, a node needs to contact the CA and request a new verifier  $V_{i+1}$ . When a identification is invalidated, the CA simply refuses issuing new verifiers.

When we chose  $T$  to be one year and  $n$  to be 365 we need one verifier per day. This means that a node must contact the CA once a day. This can be achieved e.g. when a MANET has an Internet gateway or using a GPRS connection. As we assume only loosely synchronized clocks, we decide that also the next and previous verifiers will be accepted by nodes, i.e.  $V_{i+1}$  and  $V_{i-1}$  will be accepted in section  $i$ . This also means that nodes can still proof the validity of their certificates when they cannot access the CA on one day. Furthermore, if a node receives a certificate from another node which contains no valid

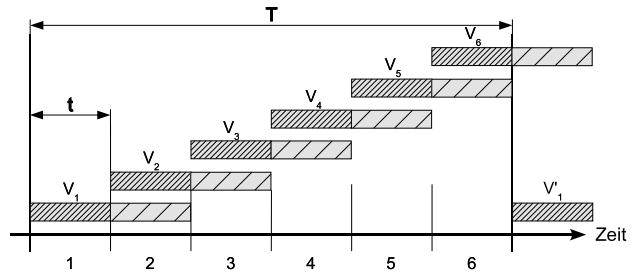


Fig. 1. Validating certificates using verifiers: The validity period  $T$  has been divided in  $n = 6$  sections. The CA distributes one verifier  $V_i$  per CA, which is valid in the respective section (densely hatched) and in the following section (lightly hatched).

verifier, the node can contact the CA itself and ask for the current verifier.

On the other hand this means that after invalidating a certificate and refusing to give out new verifiers, a node may still be able to use his keys for at least one section and at most for three sections. This can usually be tolerated by the system.

After the certificate has expired, the node  $N$  needs to contact the CA and request a certificate renewal. The communication works like shown here:

- 1)  $N \rightarrow CA : cert\_serial\_number$
- 2)  $CA \rightarrow N : E_{PK_N}(cert\_serial\_number, N_{CA})$
- 3)  $N \rightarrow CA : N_{CA} + 1$
- 4)  $CA \rightarrow N : E_{SK_{CA}}(cert', Y', valid\_until')$

In step one, the node sends a renewal request including its certificate serial number. The CA answers with a signed response acknowledging the serial number and a nonce to prevent replay attacks and authenticate node  $N$ . In the 4th message, the CA sends the new cert, including the new expiration date and end of the hash chain.

The MANET-IDs are used in the Secure Dynamic Source Routing protocol to authenticate nodes and to prevent malicious modification of routes. See [3] for details.

### C. Evaluation

The concept of MANET-IDs is based on the idea of traditional CAs. In contrast to them, only sparse communication with the CA is necessary. Furthermore certificates do not bind keys to names but confirm only the validity of keys. All questions concerning when and under what circumstances to grant certificates are therefore eliminated. As these certificates serve a well define purpose, proofing validity, their are no ambiguities regarding their meaning.

The MANET-IDs fulfill (almost) all the requirements for identifiers. They are:

- 1) ...statistically unique. The chance of creating two identical public/private key pairs is negligible. In addition, the CA, which also generates the keys, can store hash values of all emitted keys and check them to prevent duplicates.
- 2) ...tied irreversibly to this single object. Each node receives only one MANET-ID. The node cannot create new IDs on its own, as the CA certificate is then missing.
- 3) ...cannot be changed throughout the lifetime of this object. It is not intended to change the MANET-ID during the lifetime of an object. If a new MANET-ID is ever to be issued to a device (e.g. after ID invalidation), this at least involves a manual process and a contact with the CA.
- 4) cannot be transferred to other objects. This is not completely true. A node can give his public/private key pair to other nodes voluntarily or after being compromised. This can be prevented by storing at least the private key in a safe place, e.g. a smartcard. One can assume that nodes will not give away their keys voluntarily, as they risk to have their MANET-ID being invalidated. This means they cannot use MANETs anymore.

There are three potential attacks on the MANET-ID system remaining:

- The CA server can be attacked. After an successful attack, the overall security of the system is compromised. This is the same problem in any PKI system.
- The time synchronization between nodes can be attacked. As we only require a very loose synchronization in the category of hours (to identify the correct section for the revocation system), this synchronization can be achieved easily when contacting the CA to get new verifiers each day. As this communication may also be outside of the MANET (using GPRS), this is relatively hard to attack.
- There is a chance for Denial of Service attacks that prevent a node from reaching the CA and getting new verifiers. As this communication may also be outside of the MANET (using GPRS), this is relatively hard to attack. When using the MANET and an Internet gateway for communication, the SDSR protocol prevents traffic redirection attacks like blackhole routing, so an attacker needs to be directly in the route to e.g. drop packets. But dropping packets may

be recognized by the MobIDS intrusion detection system which could result in an exclusion of the malicious node.

What overhead is imposed on the MANET when using MANET-IDs? The next table shows the size of a certificate:

$PK_N$	1024 Bit	Public Key, e.g. RSA or ElGamal
$Y$	160 Bit	End of hashchain (when using SHA-1)
$serial\#$	32 Bit	Serial number
$valid\_until$	16 Bit	Expiration date
Signature	1024 Bit	digital signature of the SHA-1-hashed values from above
<b>total</b>	<b>2256 Bit</b>	$\cong$ 282 Byte

So each certificate has a size of 282 Byte. When storing e.g. 80 million certificates<sup>4</sup> plus  $Y_0$  and up to 100 revocation requests, the CA needs a capacity of some dozens of gigabytes. This is no problem for current server hardware. Of course the CA could also be realized in a decentralized manner for performance and scalability reasons.

The situation inside the MANET is more critical. The memory-constrained nodes need to store the own data plus certificate data from communication partners. Assuming about 100 communication partners per node, the results are shown in the next table:

$cert_N$	2256 Bit	Own certificate
$SK_N$	1024 Bit	Secret Key $SK_N$ , $PK_N$ contained in certificate
$V_i$	160 Bit	Cur. verifier
$cert_{N_i}$	2256 Bit	Certificate of other node
$V_{N_i}$	160 Bit	Verifier of other node
$k_{NN_i}$	128 Bit	Shared secret key, exchanged via SDSR
$\times$ 100 nodes	254400 Bit	
<b>total</b>	$\approx$ 31 kByte	per node

So the storage requirements per node are also within acceptable bounds when assuming devices like cellphones or PDAs. Finally the communication overhead when contacting the CA might be a problem.

<sup>4</sup>roughly one for every inhabitant of Germany

	send	32 Bit	S.nr. of cert.
Get verific.	recv.	160 Bit	
	total	192 Bit	$\cong$ 24 Byte
	send	96 Bit	serial-#, 64 Bit Nonce
Renew cert.	recv	3280 Bit	signed message and certificate
	total	3376 Bit	$\cong$ 422 Byte

Given that verifiers needs to be updated roughly once per day and certificates need to be renewed roughly once per year, the communication costs can also be neglected.

The drawbacks of our approach are the need of the CA infrastructure and the seldom required communication between the nodes and the CA. As we have argued we think that strong identities cannot be realized without that.

## V. SUMMARY AND OUTLOOK

We have presented the concept of MANET-IDs, which are a way to realize strong and unforgeable identities for MANETs. They provide properties like uniqueness, irreversible ties with the identified object, immutability throughout the lifetime of the object and non-transferability.

MANET-IDs also include an efficient mechanism for revocation, e.g. in case of constant misbehavior detected by the MobIDS intrusion detection system.

There is one problem with strong identification. When you are able to reliably identify nodes all of the time, then these nodes can be tracked wherever they move in the MANET. So our next goal is to provide a mechanisms that allows nodes to use changing pseudonyms instead of their constant identity while retaining all the desired properties of MANET-IDs.

## REFERENCES

- [1] F. Kargl, S. Schlott, M. Weber, A. Klenk, and A. Geiß, "Securing Ad hoc Routing Protocols," in *Proceedings of 30th Euromicro Conference*, Rennes, France, Aug. 2004.
- [2] F. Kargl, "Sicherheit in Mobilen Ad hoc Netzwerken," Ph.D. dissertation, University of Ulm, Ulm, Germany, 2003, also available as <http://medien.informatik.uni-ulm.de/~frank/research/dissertation.pdf>.
- [3] F. Kargl, A. Geiß, S. Schlott, and M. Weber, "Secure Dynamic Source Routing," in *Proceedings of the 38th Hawaii International Conference on System Sciences (HICSS-38)*, Hilton Waikoloa Village, HA, Jan. 2005.
- [4] F. Kargl, A. Klenk, S. Schlott, and M. Weber, "Advanced Detection of Selfish or Malicious Nodes in Ad hoc Networks," in *Proceedings of 1st European Workshop on Security in Ad-Hoc and Sensor Networks (ESAS 2004)*, ser. Springer LNCS, Heidelberg, Germany, Aug. 2004.
- [5] Wikipedia, "Authentication," <http://en.wikipedia.org/wiki/Authentication>, 2005.
- [6] F. Stajano and R. Anderson, "The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks," in *Security Protocols, 7th International Workshop Proceedings*, B. Christianson, B. Crispo, and M. Roe, Eds., 1999, pp. 172–194.
- [7] F. Stajano, "The Resurrecting Duckling - What Next?," in *Security Protocols, 8th International Workshop Proceedings*, B. Christianson, B. Crispo, and M. Roe, Eds., 2000, p. 204ff.
- [8] H. Luo and S. Lu, "Ubiquitous and Robust Authentication Services for Ad Hoc Wireless Networks," Department of Computer Science, University of California at Los Angeles, Tech. Rep. TR-200030, 2000.
- [9] S. Yi and R. Kravets, "Key Management for Heterogeneous Ad Hoc Wireless Networks," University of Illinois, Tech. Rep. UIUCDCS-R-2002-2290, UILU-ENG-2002-1734, 2002.
- [10] L. Zhou and Z. J. Haas, "Securing Ad Hoc Networks," *IEEE Network*, vol. 13, no. 6, pp. 24–30, 1999, also available as <http://citeseer.nj.nec.com/zhou99securing.html>.
- [11] J. Kong, P. Zerfos, H. Luo, S. Lu, and L. Zhang, "Providing Robust and Ubiquitous Security Support for Mobile Ad-hoc Networks," in *Ninth International Conference on Network Protocols (ICNP)*, 2001, pp. 251–260, also available as <http://citeseer.nj.nec.com/kong01providing.html>.
- [12] H. Luo, P. Zefros, J. Kong, S. Lu, and L. Zhang, "Self-securing Ad Hoc Wireless Networks," in *Seventh IEEE Symposium on Computers and Communications (ISCC)*, 2002, also available as <http://citeseer.nj.nec.com/507663.html>.
- [13] PGP Corporation, "Whitepaper: An Introduction to Cryptography," [http://pgp.com/products/whitepapers/pgp\\_introtocryptography.pdf](http://pgp.com/products/whitepapers/pgp_introtocryptography.pdf), 2003.
- [14] J.-P. Hubaux, L. Buttyán, and S. Čapkun, "The Quest for Security in Mobile Ad Hoc Networks," in *Proceeding of the ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC)*, 2001, also available as <http://citeseer.nj.nec.com/493788.html>.
- [15] S. Čapkun, L. Buttyan, and J.-P. Hubaux, "Self-Organized Public-Key Management for Mobile Ad Hoc Networks," Swiss Federal Institute of Technology Lausanne (EPFL), Tech. Rep., June 2002.
- [16] R. B. Bobba, L. Eschenauer, V. Gligor, and W. A. Arbaugh, "Bootstrapping Security Associations for Routing in Mobile Ad-Hoc Networks," Institute for Systems Research, UMD, Tech. Rep. TR 2002-44, May 2002, also available as <http://citeseer.nj.nec.com/bobba02bootstrapping.html>.
- [17] G. Montenegro and C. Castelluccia, "Statistically Unique and Cryptographically Verifiable (SUCV) Identifiers and Addresses," Network and Distributed System Security Symposium (NDSS), Feb. 2002, also available as <http://citeseer.nj.nec.com/montenegro02statistically.html>.
- [18] T. Aura, "RFC3972: Cryptographically Generated Addresses (CGA)," <http://www.ietf.org/rfc/rfc3972.txt>, Mar. 2005.
- [19] D. Boneh and M. Franklin, "Identity-Based Encryption from the Weil Pairing," *Lecture Notes in Computer Science*, vol. 2139, pp. 213–229, 2001.
- [20] A. Khalili, J. Katz, and W. A. Arbaugh, "Toward Secure Key Distribution in Truly Ad-Hoc Networks," in *Symposium on Applications and the Internet Workshops (SAINT 2003)*. IEEE Computer Society, 2003, pp. 342–346.

- [21] R. Wakikawa, J. T. Malinen, C. E. Perkins, A. Nilsson, and A. J. Tuominen, "Global connectivity for IPv6 Mobile Ad Hoc Networks," <http://www.wakikawa.net/Research/drafts/draft-wakikawa-manet-globalv6-02.txt>, 2002.
- [22] S. Micali, "Efficient certificate revocation," MIT Laboratory for Computer Science, Tech. Rep. TM-542b, Mar. 1996, also available as <http://citeseer.nj.nec.com/article/micali96efficient.html>.