On the Unobservability of a Trust Relation in Mobile Ad Hoc Networks

Olivier Heen[1], Gilles Guette[2], Thomas Genet[1]

[1] INRIA Bretagne Atlantique, Rennes, France
olivier.heen(@)inria.fr, thomas.genet(@)inria.fr
[2] Université Rennes 1, Rennes, France
gilles.guette(@)univ-rennes1.fr

Abstract. More and more mobile devices feature wireless communication capabilities. They can self-organize in a mobile ad hoc network in order to communicate and maintain connectivity without any infrastructure component.
In this context, some devices may benefit from established trust relations in order to communicate private data. Various solutions already exist for establishing and detecting such trust relations. But is it still possible to detect a trust relation in an unobservable manner? That is, in a way that an attacker cannot understand whether devices share a trust relation or not.
We exhibit a solution to this problem. Our solution guaranties the anonymity and the unobservability of participants against passive and active attackers. The security properties of the solution are machine checked with the AVISPA framework [2] and the SPAN tool [5].
The main applications could be found in mobile ad hoc networks and in vehicular networks [6, 7] where anonymity and unobservability contribute to a better privacy.

1 Introduction

These last years we see an explosion of the number of mobile devices featuring wireless capabilities. Such devices can self-organized as the *nodes* of a Mobile Ad hoc NETwork, a MANET. Some nodes can join and leave the MANET at any time according to their moves and communication range.

The notion of trust naturally appears in MANET: some nodes can establish long term trust relations in order to better communicate the next time they meet. Typically, trusted nodes may exchange private data while the other nodes only exchange public data. The so-called resurrecting duckling [13] is a well known-solution for the establishment and detection of trust relations between nodes. Other solutions like [11] are particularly suitable for home networks. According to many solutions the user contributes to the establishment of the trust relations, and later on the nodes detect their trust status as soon as they start to communicate without any user intervention. By trust status we mean the fact of sharing a trust relation or not.

In the same time, privacy becomes a serious concern in a variety of domains including many typical application fields of MANET: home networks, Personal Area Networks (PAN), vehicular networks (VANET) [14, 1], etc. The greater the ability of devices to communicate, the greater the risk of disclosing private information. In the field of VANET for instance, malicious observers should not trace the moves of a vehicle for a long period of time, simply based on its network communications [12].

In some cases, even the existence of a trust relation between two nodes must remain private. This can be the case when the nodes belong to a same group of interest, friends, family etc. In the field of VANET, this can be critical for unmarked police cars: they should not be disclosed by their trust relation.

In this paper, we address the following question: *is it possible to detect trust relations in MANET in an anonymous and unobservable manner?* In particular, can two communicating nodes detect their trust status and still be sure that no untrusted node has detected it either?

We mainly focus on the existence of a solution to the problem, in presence of passive or active attackers. To do so, we use a binary notion or trust: a node does trust another node, or does not. Note that some more graduated notions of trust exist, with for instance a parameter for the intensity of trust or taking in account the number of positive vs. negative experiences with other nodes... We do not formally consider these notions. Instead we consider that any non-null trust is a maximal trust, and bring unobservability in this worst case.

The section 2 provides the necessary definitions as well as the constraints we will respect in the design of the solution. The section 3 provides the appropriate notations and gives a full description of our solution. The section 4 gives arguments for the proof of the security, the anonymity and the unobservability properties. A significant part of the verification is performed using the AVISPA [2] framework together with the animation tool SPAN [5].

2 Preliminaries

2.1 Definitions

Firstly, we give the definition related to the nodes and the trust relation.

Node: a node A is a communicating device with a pair of keys denoted (K_A, K_A^{-1}). No key in the pair will be public. The node set is noted $\mathcal{N}$.

We define the trust relation for which we want to achieve unobservability against some attackers.

Trust relation: a node B trusts another node A, denoted $B\top A$, if B knows the key K_A. We can see that $B\top A$ does not imply that $A\top B$.

Trust community: the trust community of a node B, denoted $\mathcal{T}_B$, is the set of nodes trusted by B. $\mathcal{T}_B = \{A \in \mathcal{N}, B\top A\}$

We give in the following the most common definition of the anonymity and some related concepts. These definitions are taken from [9, 8].

Anonymity set: an anonymity set is a set of nodes having the same attributes and capable to perform the same actions.

Anonymity: a node remains anonymous if we cannot identify it in an anonymity set. Sender (resp. receiver) anonymity is provided when a message analysis do not allow determining its sender (resp. receiver) in the anonymity set.

Unobservability: there is unobservability of an event when an attacker cannot deduce the existence of this event. There is unobservability of the sender (resp. receiver) when an attacker cannot deduce that a message was sent (resp. received).

We will first provide a solution for the unobservability of a trust relation against a passive attacker.

Passive attacker: a passive attacker can dump every message sent on the network. She does not know, *a priori*, any cryptographic key.

Then, we will refine this solution to thwart an active attacker.

Active attacker: an active attacker can dump, add, remove and modify every message on the network. We say that "the attacker *is* the network". Note that this kind of attacker is also called a Dolev-Yao attacker.

2.2 Trust relation

Numerous trust models exist, coming from the simple secret key sharing to certificate chains. For further information on the different trust models, read [3] or [11]. The model we use is particularly adapted to the MANET. We give here the main characteristics and some use cases.

Establishment of trust: the trust relation $B\top A$ is established by transmitting K_A to B over a secure channel. Note that if A changes its key or if B forgets K_A it will be necessary to rebuild the trust relation. If B does not trust A anymore it just has to remove K_A from its key database.

Extension of trust (a.k.a recommendation): in the case $B \top A$, B can decide to recommend A to another node X; it just has to send K_A to X over a secure channel. In the example of vehicular network, a truck B can trust road equipment A (base station, toll, etc.) and transmit this trust to another truck X, even if A is not here during the key transmission.

3 Our solution

Figure 1 gives all the notations used in this paper.

Notation	**Definition**	**Notation**	**Definition**
$A, B, C \dots$	Network nodes	g	Diffie-Hellman generator
K_A, K_A^{-1}	Key pair of node A	*Private*	Message to protect
$\{i.0.db\}_{K_A^{-1}}$	a pseudonym of node A	*Public*	Other message
$B \top A$	B trust A	$m_1, m_2 \dots$	Random messages
$\mathcal{T}_A$	Trust community of A	db	Public number like `0xDEADBEEF`
$i, j, k \dots$ and R	Fresh random numbers (never used before)	$\{\}_K$	Encryption / Decryption algorithm using key K

Fig. 1. Notations

Our solution is based on three significant techniques: renewable pseudonyms, message encryption, periodical broadcast.

The general format of a message in our protocol is:

$$\{i.0.db\}_{K_A^{-1}}.\{j.0.db\}_{K_B^{-1}}.g^i a.m$$

where $\{i.0.db\}_{K_A^{-1}}$ is a pseudonym of the sender, $\{j.0.db\}_{K_B^{-1}}$ is a pseudonym of the receiver, g^i a Diffie-Hellman value, m the encrypted payload. Note that since an observer should not distinguish anything, all the messages must respect the same format.

Our solution builds a Diffie-Hellman key and tries to authenticate this key by using the information contained in the pseudonyms. The authentication will succeed only when a trust relation exists, otherwise the nodes know that they do not share any trust relation (refer to [4] for more information about authenticated Diffie-Hellman exchanges).

We now provide the sequence of message according to our solution in various cases.

Case $B \top A$ (B knows K_A) and A starts:

1. $A \rightarrow All$: $\{i.0.db\}_{K_A^{-1}}.R.g^i.m_1$ (with R some padding)
2. B checks that $\{\{i.0.db\}_{K_A^{-1}}\}_{K_A} = i'.0.db$ and $g^{i'} = g^i$
3. $B \rightarrow All$: $\{i.j.db\}_{K_A}.\{i.0.db\}_{K_A^{-1}}.g^j.m_2$
4. A checks that $\{\{i.j.db\}_{K_A}\}_{K_A^{-1}} = i'.j'.db$ and $i' = i$ and $g^{j'} = g^j$
5. $A \rightarrow All$: $\{i.0.db\}_{K_A^{-1}}.\{i.j.db\}_{K_A}.g^k.\{Public\}_{g^{ij}}$
6. $B \rightarrow All$: $\{i.j.db\}_{K_A}.\{i.0.db\}_{K_A^{-1}}.g^l.\{Private\}_{g^{ij}}$

At the step 2, B detects that $\{i.0.db\}_{K_A^{-1}}$ is a pseudonym of a trusted node. A simple manner for detecting this is to sequentially[3] try all the keys in $\mathcal{T}_B$ until the key K_A successfully decrypts the pseudonym of A.

At the step 4, A detects that $\{i.j.db\}_{K_A}$ is a pseudonym of a trusted node, nevertheless A does not know that this node is B.

The nodes A and B may then continue to use there pseudonym in the further exchanges. They can send as many encrypted messages as they need, with A sending encrypted *Public* messages and B sending *Public* or *Private* messages:

$A \rightarrow All$: $\{i.0.db\}_{K_A^{-1}}.\{i.j.db\}_{K_A}.g^l.\{Public\}_{g^{ij}}$
$B \rightarrow All$: $\{i.j.db\}_{K_A}.\{i.0.db\}_{K_A^{-1}}.g^m.\{Private\}_{g^{ij}}$

If one node changes its pseudonym, a new detection of trust will happen and re-establish the communication.

Case without trust relation: We describe in this section the sequence of exchanged message between two nodes A and C that do not share any trust relation. Nevertheless, A and B want to communicate with each other.

1. $A \rightarrow All$: $\{i.0.db\}_{K_A^{-1}}.R.g^i.m_1$ (with R some padding)
2. $B \rightarrow All$: $\{j.0.db\}_{K_B^{-1}}.\{i.0.db\}_{K_A^{-1}}.g^j.m_2$
3. $A \rightarrow All$: $\{i.0.db\}_{K_A^{-1}}.\{j.0.db\}_{K_B^{-1}}.g^k.\{Public\}_{g^{ij}}$

In this case, A and B do not share any key and then they cannot collaborate for checking the Diffie-Hellman values. The key used to encrypt the payload is the Diffie-Hellman key g^{ij} generated with the third parts of the messages 1 and 2. Note that a *man-in-the-middle* attack is possible here against the Diffie-Hellman part of the solution; this point is further discussed in 4.4.

[3] If B trusts a lot of nodes the set $\mathcal{T}_B$ is large and the detection can take a lot of time. There exist many ways to improve the efficiency, like trying to decrypt with the most often used key first.

Case $B \top A$ and B starts: this case is not managed in a particular way : since messages are periodically broadcasted one cannot predict if A or B starts the protocol. Here, B starts as in the case without trust. Since A does not trust B, A continues as in the case without trust. Only then B has the possibility to detect the trust relation. At this point the situation is exactly the same as when A started the communication. In particular B will drop its old pseudonym $\{j.0.db\}_{K_B^{-1}}$ for choosing a pseudonym of the form$\{j.k.db\}_{K_A}$ very much like in step 3 when A starts.

Case $A \top B$ and $B \top A$: this case is not managed in a particular way: depending on which node starts the case $A \top B$ or the case $B \top A$ is resolved first. Nevertheless, once the anonymous secure channel is built between A and B, it is always possible for a node to ask the authentication of the other node, for instance by signing a random value. For instance, if the case $B \top A$ is resolved first, A can authenticate itself to B by sending:
$A \rightarrow All\ : \{i.0.db\}_{K_A^{-1}}.\{i.j.db\}_{K_A}.g^l.\{n.\{n\}_{K_A^{-1}}\}_{g^{ij}}$,
B checks $\{n\}_{K_A^{-1}} = n'$ and $n' = n$

4 Analysis of the solution

4.1 Basic security properties

Our solution must provide confidentiality of communications between nodes A and B when $B \top A$.

We use the AVISPA framework to prove this property. We first write a full specification of the protocol (given in appendix), then we specify the secrecy property and we run the AVISPA detection tools. The specification corresponds to the case $B \top A$ and we check the two situations: when A starts the communication and when B starts the communication. No attack was found. The case $A \top B$ is verified by symmetry of the roles of A and B.

4.2 Anonymity properties

We first remark that all the participants X to a communication are using pseudonym $\{i.0.db\}_{K_X^{-1}}$. In order to keep the long term secret K_X undisclosed, the cryptographic algorithm $\{\}_{K_X^{-1}}$ must reveal nothing about its key. This is one basic property of asymmetric encryption algorithms.

We also remark that the nodes are regularly updating their pseudonyms. In particular, they choose a new pseudonym each time they want to establish a trusted connection with other nodes. Thus, one single node can use many pseudonyms at a same time: some for untrusted relations, some other for trusted relations. Depending of whether there is trust or not, the way pseudonyms are built varies but neither form of the pseudonyms reveals anything about long term secrets.

Since no permanent secret is revealed, the effective anonymity only relies on the size of the anonymity set. If the anonimity set is restricted to one single device, of course the anonimity does not hold. But in the case of VANET for instance, the typical anonymity set is the set of all pseudonyms used by all vehicles communicating in the attacker's vicinity during the observation period.

4.3 Unobservability against passive attackers

Regarding unobservability, our solution exhibits three properties:

1. All communications are broadcasted.
2. All nodes are regularly sending messages.
3. All message components are encrypted or have the form g^i.

According to [8] §8, properties 1 and 3 imply receiver unobservability. Properties 2 and 3 imply sender unobservability. These to properties imply message unobservability, which in turns implies the unobservability of the trust relation by the argument hereafter:
Ad absurdum, we assume message unobservability but not trust relation unobservability. We consider the shortest message sequence that leads the attacker to observe the trust relation. We consider the last message of this sequence: it is responsible for the detection of the trust relation (otherwise this message can be removed from the sequence, which contradicts the assumption that this is the shortest). We remark that attacker is not able to detect the trust relation without observing this last message (otherwise this also contradicts the assumption that the sequence is the shortest). Thus, reading just the last message, the attacker detects the trust relation. In other terms, the attacker is able to observe the last message, which contradicts message unobservability.

In order to better illustrate the unobservability property, we provide the two tables hereafter. The first table shows the case $B \top A$ and what a passive attacker can deduce. The second table shows the case without trust and what a passive attacker can deduce. α, β, γ are observed pseudonyms that the attacker cannot decrypt. The g_x are observed g^i values that the attacker cannot reduce (she does not know i). The m_x are observed encrypted payloads.

Communication when $B \top A$	Observations of a passive attacker
$A \rightarrow All\ : \{i.0.db\}_{K_A^{-1}}.R.g^i.m_1$	$X_1 \rightarrow All\ : \alpha.\beta.g_1.m_1$
$B \rightarrow All\ : \{i.j.db\}_{K_A}.\{i.0.db\}_{K_A^{-1}}.g^j.m_2$	$X_2 \rightarrow All\ : \gamma.\alpha.g_2.m_2$
$A \rightarrow All\ : \{i.0.db\}_{K_A^{-1}}.\{i.j.db\}_{K_A}.g^k.\{Public\}_{g^{ij}}$	$X_3 \rightarrow All\ : \alpha.\gamma.g_3.m_3$
$B \rightarrow All\ : \{i.j.db\}_{K_A}.\{i.0.db\}_{K_A^{-1}}.g^l.\{Private\}_{g^{ij}}$	$X_4 \rightarrow All\ : \gamma.\alpha.g_4.m_4$

Communication when there is no trust	Observations of a passive attacker
$A \rightarrow All\ : \{i.0.db\}_{K_A^{-1}}.R.g^i.m_1$	$X_1 \rightarrow All\ : \alpha.\beta.g_1.m_1$
$B \rightarrow All\ : \{j.0.db\}_{K_B^{-1}}.\{i.0.db\}_{K_A^{-1}}.g^j.m_2$	$X_2 \rightarrow All\ : \gamma.\alpha.g_2.m_2$
$A \rightarrow All\ : \{i.0.db\}_{K_A^{-1}}.\{j.0.db\}_{K_B^{-1}}.g^k.\{Public\}_{g^{ij}}$	$X_3 \rightarrow All\ : \alpha.\gamma.g_3.m_3$
$B \rightarrow All\ : \{j.0.db\}_{K_B^{-1}}.\{i.0.db\}_{K_A^{-1}}.g^l.\{Public\}_{g^{ij}}$	$X_4 \rightarrow All\ : \gamma.\alpha.g_4.m_4$

We see that the passive attacker observes the same information in both tables. In fact, the general expression of what the passive attacker observes at step n in both cases is $obs(n)$:

$$obs(n) = \begin{cases} X_1 \rightarrow All\ : \alpha.\beta.g_1.m_1 & \text{if } n = 1, \\ X_n \rightarrow All\ : \gamma.\alpha.g_n.m_n & \text{if } n = 2p, \\ X_n \rightarrow All\ : \alpha.\gamma.g_n.m_n & \text{if } n = 2p+1 \end{cases}$$

4.4 Active attack against unobservability

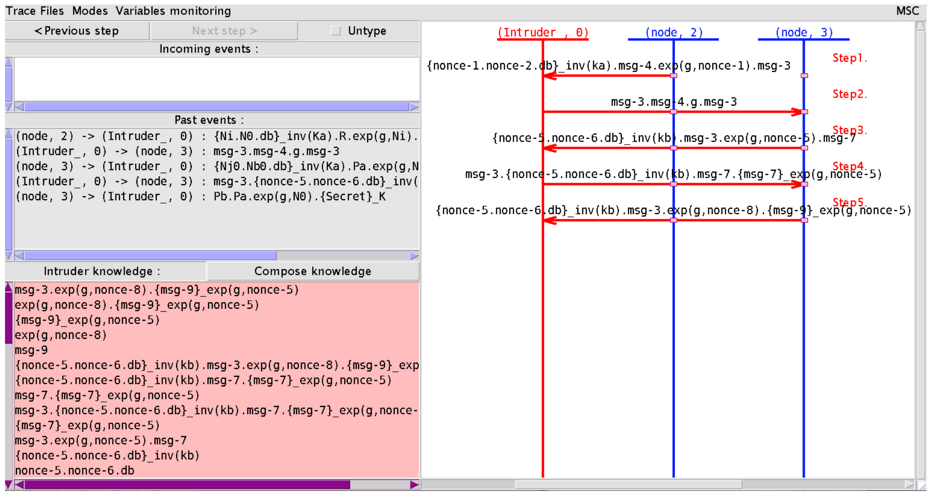
We show here an attack found by the OFMC tool of AVISPA. The countermeasure is given in 4.5 but we find it profitable to precisely explain the attack as it is illustrative for:

– the concrete benefit of using automated verification tools;
– one way to defeat unobservability while keeping anonymity;
– the power of active attackers and the practical conditions of the attack.

If there is no trust between A and B (see 4.4) the secrecy of their subsequent communications only depends on the Diffie-Hellman key agreement performed within the third part of each message. Thus the secret only holds against a passive attacker but not against an active attacker because she can perform a *man-in-the-middle* attack. The practical conditions of this attack may be very complicated but one variant is much simpler. This variant is automatically found by OFMC tool of AVISPA: the figure 4.4 shows the corresponding message sequence chart as redrawn by the tool SPAN [5]. The attack works as follows:

1. The attacker D captures a pseudonym, for instance $\{j.0.db\}_{K_B^{-1}}$.
2. The node A normally starts a communication by sending:
   $A \rightarrow All\ : \{i.0.db\}_{K_A^{-1}}.R.g^i.m_1$ (with R some padding)

3. The active attacker D regularly tries to answer A by sending:
   $D \rightarrow All\ : \{j.0.db\}_{K_B^{-1}}.\{i.0.db\}_{K_A^{-1}}.g.m.$
   Note that the third part of the message is g (in fact any low power of g will work, like $g^2, g^3 \ldots$).
4. If A continues the protocol normally, it will send a message with an encrypted payload:
   $A \rightarrow All\ : \{i.0.db\}_{K_A^{-1}}.\{j.0.db\}_{K_B^{-1}}.g^k.\{Public\}_{g^i}.$
   Note that the encryption key g^i is known by any node having received the first message.



```
(node, 2) -> (Intruder_, 0) : {Ni.N0.db}_inv(Ka).R.exp(g,Ni).
(Intruder_, 0) -> (node, 3) : msg-3.msg-4.g.msg-3
(node, 3) -> (Intruder_, 0) : {Nj0.Nb0.db}_inv(Ka).Pa.exp(g,N
(Intruder_, 0) -> (node, 3) : msg-3.{nonce-5.nonce-6.db}_inv(
(node, 3) -> (Intruder_, 0) : Pb.Pa.exp(g,N0).{Secret}_K
```

```
msg-3.exp(g,nonce-8).{msg-9}_exp(g,nonce-5)
exp(g,nonce-8).{msg-9}_exp(g,nonce-5)
{msg-9}_exp(g,nonce-5)
exp(g,nonce-8)
msg-9
{nonce-5.nonce-6.db}_inv(kb).msg-3.exp(g,nonce-8).{msg-9}_exp
{nonce-5.nonce-6.db}_inv(kb).msg-7.{msg-7}_exp(g,nonce-5)
msg-7.{msg-7}_exp(g,nonce-5)
msg-3.{nonce-5.nonce-6.db}_inv(kb).msg-7.{msg-7}_exp(g,nonce-
{msg-7}_exp(g,nonce-5)
msg-3.exp(g,nonce-5).msg-7
{nonce-5.nonce-6.db}_inv(kb)
nonce-5.nonce-6.db
```

```
{nonce-1.nonce-2.db}_inv(ka).msg-4.exp(g,nonce-1).msg-3
msg-3.msg-4.g.msg-3
{nonce-5.nonce-6.db}_inv(kb).msg-3.exp(g,nonce-5).msg-7
msg-3.{nonce-5.nonce-6.db}_inv(kb).msg-7.{msg-7}_exp(g,nonce-5)
{nonce-5.nonce-6.db}_inv(kb).msg-3.exp(g,nonce-8).{msg-9}_exp(g,nonce-5)
```

Fig. 2. Verification of an active attack against unobservability, as found by AVISPA and displayed by SPAN.

Note that some tools that we used are proven complete [15]: when no attack is found this means that there is no attack involving an arbitrary number of intruder operations.

This attack is not serious *per se* since it only discloses information that nodes are willing to send even without trust relation. But the attack has consequences over the unobservability of the trust relation between the nodes with pseudonyms $\{i.0.db\}_{K_A^{-1}}$ and $\{j.0.db\}_{K_B^{-1}}$: if the active attacker is able to decrypt the *Public* payload, this means that there is no trust relation between these nodes. Otherwise the attacker cannot decrypt the payload and then deduces that the node with pseudonym

$\{i.0.db\}_{K_A^{-1}}$ detected $g^i \neq g$. This case only happens if a trust relation exists.

This attack only defeats the unobservability of the trust relation. The anonymity is still guaranteed, since there is still no way for the attacker to infer A from $\{i.0.db\}_{K_A^{-1}}$ nor to infer identity information from the $Public$ payload.

4.5 Countermeasure

The countermeasure repairs unobservability when a message of the form $\{j.0.db\}_{K_B^{-1}}.\{i.0.db\}_{K_A^{-1}}.g^k.m$ is received and when the abnormal situation $j \neq k$ is detected (in the above description of the attack, we had $k = 1$). In this case, the detecting node can suspect an attack. We modify its behavior so that it falls back to a non-trust behavior instead of continuing to enforce trust. In particular it will compute the Diffie-Hellman key g^{jk} and send public data, like $\{Public\}_{g^{jk}}$. Of course, the attacker will be able to decrypt this data. But, since decryption will always work, she will not learn anything about the trust relation.

The trust relation will be successfully detected and enforced only if there is not attack, that is in the nominal case:

$\{j.0.db\}_{K_B^{-1}}.\{i.0.db\}_{K_A^{-1}}.g^k.m$ and $j = k$.

It might be argued that the protocol now silently fails when a node detects an attack, thus giving the attacker an additional way to perform a denial-of-service attack. This is true but in our attacker model the attacker already have the possibility to block *all* messages (as often in presence of mobile wireless communications).

5 Conclusion

Privacy issues in network communications become more and more important. In particular in MANET or VANET users take care about their privacy and do not want to reveal anything about their activities and personal or professional travels. In certain cases, just detecting that Bob has meet Alice on some place may reveal partial information about industrial secrets or vendor strategies. In this kind of networks, we have provided a solution for detecting a particular trust relation between two nodes in an anonymous and unobservable way. We believe that these two properties will be of first importance in a near future in the design of security protocols, for instance in the RFID research field [16, 10]. In future work, we plan to address some complexity issues of our solutions:

decreasing the complexity of trust detection algorithm, reduce the use of asymmetric cryptography and add appropriate cryptographic puzzles for mitigating the exhaustion of computation resources. It is also suitable to study different trust models, not necessarily based on asymmetric cryptography.

Acknowledgment

The author would like to thank Ciarán Bryce for valuable discussions and appreciation. The author would like to thank the WISTP 2009 reviewers for many in depth comments and useful remarks.

References

1. F. Dötzer. Privacy Issues in Vehicular Ad Hoc Network. In *Workshop on Privacy Enhancing Technologies*, pages 197–209, 2005.
2. A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuellar, P. Hankes Drielsma, P.-C. Héam, O. Kouchnarenko, J. Mantovani, S. Mödersheim, D. von Oheimb, M. Rusinowitch, J. Santos Santiago, M. Turuani, L. Viganò, and L. Vigneron. The AVISPA Tool for the automated validation of internet security protocols and applications. In K. Etessami and S. Rajamani, editors, *17th International Conference on Computer Aided Verification, CAV'2005*, volume 3576 of *Lecture Notes in Computer Science*, pages 281–285, Edinburgh, Scotland, 2005. Springer.
3. D. Balfanz, D. Smetters, P. Stewart, and H. Wong. Talking to strangers: Authentication in adhoc wireless networks, February 2002. In Symposium on Network and Distributed Systems Security (NDSS '02), San Diego, California.
4. S. Blake-Wilson and A. Menezes. Authenticated Diffie–Hellman key agreement protocols. In *Selected Areas in Cryptography*, pages 339–361, 1998.
5. Y. Boichut, T. Genet, Y. Glouche, and O. Heen. Using Animation to Improve Formal Specifications of Security Protocols. In *2nd Conference on Security in Network Architectures and Information Systems (SARSSI 2007)*, pages 169–182, 2007.
6. E. Fonseca, A. Festag, R. Baldessari, and R. Aguiar. Support of Anonymity in VANETs - Putting Pseudonymity into Practice. In *IEEE Wireless Communications and Networking Conference*, 2007.
7. M. Gerlach, A. Festag, T. Leinmüller, G. Goldacker, and C. Harsch. Security Architecture for Vehicular Communication. In *Workshop on Intelligent Transportation*, 2007.
8. A. Pfitzmann and M. Hansen. Anonymity, unobservability, and pseudonymity: A consolidated proposal for terminology. Draft, July 2008.
9. A. Pfitzmann and M. Köhntopp. Anonymity, unobservability, and pseudonymity - a proposal for terminology. In *Workshop on Design Issues in Anonymity and Unobservability*, pages 1–9, 2000.
10. R. Di Pietro and R. Molva. Information confinement, privacy and security in rfid systems. In *ESORIC 2007*, pages 187–202, 2007.

11. N. Prigent, C. Bidan, J.P. Andreaux, and O. Heen. Secure long term communities in ad hoc networks. In *SASN '03: Proceedings of the 1st ACM workshop on Security of ad hoc and sensor networks*, pages 115–124, New York, NY, USA, 2003. ACM.
12. B. Schneier. Blog article, tracking automobiles through their tires, December 2006.
13. F. Stajano and R. J. Anderson. The resurrecting duckling: Security issues for ad-hoc wireless networks. In *Proceedings of the 7th International Workshop on Security Protocols*, pages 172–194, London, UK, 2000. Springer-Verlag.
14. C.K. Toh. Research challenges in intelligent transportation networks. keynote speach at ifip networking 2008, singapour, May 2008.
15. M. Turuani. *Security of Cryptographic Protocols: Decidability and Complexity.* PhD thesis, Université of Nancy 1, 2003.
16. T. van Deursen, S. Mauw, and S. Radomirović. Untraceability of rfid protocols. In *Workshop on Information Security Theory and Practices*, pages 1–15, 2008.

We provide here the formal specification of the protocol, as used for the verification of security properties within the AVISPA framework.

```
role node (A:agent,Ka:public_key,KeyRing:(agent.public_key)
set,SND,RCV:channel(dy)) played_by A def=
local State :nat, Ni,Nj,Nj0,N0,Nb0:text,
      Pa,Pb,Mx,R,Dh,Mauth,Private,K:message,
      Kx:public_key, X:agent
init  State:=0
transition
a0.   State=0 /\ RCV(start) =|> State':=1
      /\ Ni':=new() /\ N0':=new() /\ Pa':={Ni'.N0'.db}_inv(Ka) /\ R':=new()
      /\ Mx':=new() /\ SND(Pa'.R'.exp(g,Ni').Mx')
ab1.  State=1 /\ RCV(Pb'.Pa.exp(g,Nj').Mx') /\ Pb'={Ni.Nj'.db}_Ka =|> State':=2
      /\ K':=exp(exp(g,Nj'),Ni) /\ Mauth':=new() /\ N0':=new()
      /\ SND(Pa.Pb'.exp(g,N0').{Mauth'}_K') /\ witness(A,b,bob_alice_na,Mauth')
ab2.  State=2 /\ RCV(Pb.Pa.Dh'.{Private'}_K) =|> State':=3
ac1.  State=1 /\ RCV({R'}_inv(Kx').Pa.Dh'.Mx') =|> State':=2
      /\ Pb':={R'}_inv(Kx') /\ K':=exp(Dh',Ni) /\ Mauth':=new() /\ N0':=new()
      /\ SND(Pa.Pb'.exp(g,N0').{Mauth'}_K')
ac2.  State=2 /\ RCV(Pb.Pa.Dh'.{Private'}_K) =|> State':=3
ba0.  State=0 /\ RCV(Pa'.R'.exp(g,Ni').Mx') /\ Pa'= {Ni'.N0'.db}_inv(Kx')
      /\ in(X'.Kx',KeyRing)  =|> State':=5
      /\ Dh':=exp(g,Ni') /\ Nj':=new() /\ Pb':={Ni'.Nj'.db}_Kx'
      /\ Mx':=new() /\ K':=exp(Dh',Nj') /\ SND(Pb'.Pa'.exp(g,Nj').Mx')
ba1.  State=5 /\ RCV(Pa.Pb.Dh'.{Mauth'}_K) =|> State':=6
      /\ N0':=new() /\ Private':=new() /\ SND(Pb.Pa.exp(g,N0').{Private'}_K)
      /\ request(X,A,bob_alice_na,Mauth') /\ secret(Private',sec,{A,X})
bc0.  State=0 /\ RCV({Ni'.N0'.db}_inv(Kx').R'.Dh'.Mx')
      /\ not(in(X'.Kx',KeyRing)) =|> State':=7 /\ Pa':={Ni'.N0'.db}_inv(Kx')
      /\ Nj':=new() /\ Nb0':=new() /\ Pb':={Nj'.Nb0'.db}_inv(Ka) /\ Mx':=new()
      /\ K':=exp(Dh',Nj') /\ SND(Pb'.Pa'.exp(g,Nj').Mx')
bc1.  State=7 /\ RCV(Pa.Pb.Dh'.{Mauth'}_K) =|> State':=8
      /\ N0':=new() /\ Private':=new() /\ SND(Pb.Pa.exp(g,N0').{Private'}_K)
end role

role environment() def=
local KeyMapA,KeyMapB,KeyMapC,KeyMapD:(agent.public_key) set,
      SND,RCV:channel(dy)
const a,b,c,d,i:agent, ka,kb,kc,kd,ki:public_key,
      g,db:text, sec,nb,alice_bob_nb,bob_alice_na:protocol_id
init  KeyMapA:={} /\ KeyMapB:={a.ka} /\ KeyMapC:={} /\ KeyMapD:={a.ka,b.kb}
      intruder_knowledge={a,b,c,d,g,ki,inv(ki)}
composition
      node(a,ka,KeyMapA,SND,RCV) /\ node(b,kb,KeyMapB,SND,RCV)
end role

goal
      secrecy_of sec
      authentication_on bob_alice_na
end goal
environment()
```