Improving IP address autoconfiguration security in MANETs using trust modelling

Shenglan Hu[*] and Chris J. Mitchell

Information Security Group, Royal Holloway, University of London
{s.hu,c.mitchell}@rhul.ac.uk

Abstract. Existing techniques for IP address autoconfiguration in mobile ad hoc networks (MANETs) do not address security issues. In this paper, we first describe some of the existing IP address autoconfiguration schemes, and discuss their security shortcomings. We then provide solutions to these security issues based on the use of trust models. A specific trust model is also proposed for use in improving the security of existing IP address autoconfiguration schemes.

1 Introduction

IP address autoconfiguration is an important task for zero configuration in ad hoc networks, and many schemes have been proposed [5, 6, 11]. However, performing IP address autoconfiguration in ad hoc networks securely remains a problem. Most existing schemes are based on the assumption that the ad hoc network nodes will not behave maliciously. However, this is not always a realistic assumption, since not only may some malicious nodes be present, but ad hoc network nodes are often easily compromised.

Of the existing IP address autoconfiguration schemes, we focus here on *requester-initiator schemes*. In such a scheme, a node entering the network (the *requester*) will not obtain an IP address solely by itself. Instead, it chooses an existing network node as the *initiator*, which performs address allocation for it.

The rest of this paper is organised as follows. In Section 2, we review existing requester-initiator schemes, and analyse their security properties. In Section 3, we describe how to improve the security of these schemes using trust models. In Section 4, a new trust model is proposed which can be used to secure the operation of requester-initiator schemes, and an analysis of this trust model is given. Finally, a brief conclusion is provided in section 5.

2 Requester-initiator Address Allocation Schemes

We first briefly review two existing requester-initiator schemes. We then use them to illustrate a variety of security issues that can arise in such schemes.

[*] The work of this author was sponsored by Vodafone.

Nesargi and Prakash proposed the MANETconf scheme [5], a distributed dynamic host configuration protocol for MANETs. In this scheme, when a new node (the requester) joins an ad hoc network, it broadcasts a request message to its neighbour nodes. If the requester is the only node in the network, then it becomes an initiator; otherwise, it will receive a reply message from one or more of its neighbour nodes. It then selects one of its reachable neighbour nodes as an initiator. Each node in the network stores the set of addresses currently being used, as well as those that can be assigned to a new node. The initiator selects an IP address from the available addresses and checks the uniqueness of the address by broadcasting a message to all network nodes. If the IP address is already being used, then the initiator selects another IP address and repeats the process until it finds a unique IP address which it allocates to the requester.

Fazio, Villri and Puliafito [2] proposed another requester-initiator scheme for IP address autoconfiguration based on MANETconf [5] and the Perkins-Royer-Das [6] scheme. In this protocol, a NetID is associated with each ad hoc network, and each node is therefore identified by a (NetID, HostID) pair. When a new node (a requester) joins an ad hoc network, it randomly selects a 4-byte HostID and requests the initiator to allocate it a unique address. The initiator randomly chooses a candidate IP address and broadcasts it to all other nodes to check for uniqueness. When the initiator finds a unique address, it sends the requester this address and the NetID of the network. The NetID is used to detect merging of networks. When partitioning happens, the NetID of each part will be changed.

In all requester-initiator schemes, including those briefly described above, IP address allocation for new nodes depends on the correct behaviour of the initiator and other existing nodes. However, in reality, malicious nodes may be present in an ad hoc network, potentially causing a variety of possible problems. We now note a number of potential security problems.

Firstly, if a malicious node acts as an initiator, it can deliberately assign a duplicate address to a requester, causing IP address collisions. In schemes where a node stores the set of addresses currently being used, it can also trigger IP address allocations for nodes that do not exist, thereby making IP addresses unavailable for other nodes that may wish to join the MANET. This gives rise to a serious denial-of-service attack.

Secondly, a malicious node can act as a requester and send address request messages to many initiators simultaneously, who will communicate with all other nodes in order to find a unique address. This will potentially use a lot of the available bandwidth, again causing a denial-of-service attack.

Thirdly, a malicious node in the network could claim that the candidate IP address is already in use whenever it receives a message from an initiator to check for duplication. As a result no new nodes will be able to get an IP address and join the network. It can also change its IP address to deliberately cause an IP address collision, forcing another node to choose a new IP address. This could lead to the interruption of that node's TCP sessions.

The main purpose of this paper is to consider means to address these threats. We start by showing how a trust model satisfying certain simple properties can be used to reduce these threats. Note that all three threats identified above result in denial-of-service attacks — this provides us with an implicit definition of what we mean by a malicious node, i.e. a node seeking to deny service to other network nodes; the nature of a malicious node is discussed further below.

3 Solutions Based on Trust Models

According to clause 3.3.54 of ITU-T X.509 [9], trust is defined as follows: “Generally an entity can be said to ‘trust’ a second entity when the first entity makes the assumption that the second entity will behave exactly as the first entity expects”. In this paper, trust (in the form of a *trust value*) reflects the degree of belief that one entity has in the correctness of the behaviour of another entity. It is dynamic, and reduces if an entity misbehaves, and vice versa.

For the moment we do not assume any particular method of computing trust values; we simply suppose that such a method has been selected. We use the following terminology. For any nodes A and B in an ad hoc network, the trust value held by A for B, i.e. the level of trust A has in B, is a rational (floating point) value denoted by $T_A(B)$. Every node A has a *threshold trust value*, denoted by T^*_A. That is, A deems B as trustable if and only if the trust value that A currently assigns to B is at least its threshold trust value, i.e. $T_A(B) \geq T^*_A$; otherwise node A will regard node B as a potentially malicious node.

Each node maintains its own threshold value T^*_A, i.e. different nodes may choose different trust thresholds. Hence the definition of malicious node may vary from node to node, depending on local policy. Every node also keeps a *blacklist*. Whenever it finds another node for which its trust value is lower than its threshold trust value, it deems this node a malicious node and adds this node to its *blacklist*. It will regularly calculate its trust values for the nodes in its *blacklist* and update its *blacklist* based on these new trust values. Except for the messages used for calculating trust values, it will ignore all other messages from nodes in its *blacklist* and will not route any other messages to these nodes.

One underlying assumption in this paper is that the number of malicious nodes in an ad hoc network is small. We also assume there is a trust model available with the following two properties, where the neighbour nodes of a node are those nodes that are within direct transmission range. (1) Any node can make a direct trust judgement on its neighbour nodes based on the information it gathers in a passive mode. If one of its neighbour nodes is malicious, it can detect the misbehaviour of this malicious node. It maintains trust values for all its neighbour nodes and regularly updates them. (2) Any node is able to calculate the trust values of non-neighbour nodes based on the trust values kept by itself and/or other nodes. We now show how such a trust model can be used to improve the security of any requester-initiator scheme.

3.1 Choosing a Trustable Node as the Initiator

When a node N joins a network, it broadcasts a request message $Neighbour\_Query$ containing T^*_N to its neighbour nodes. If the requester is the only node in the network, then it becomes an initiator; otherwise, each of the other nodes receiving a *Neighbour_Query* message will check the trust values it hold for its neighbour nodes, and send N a reply message *InitREP*, containing identifiers of the nodes for which the trust values it holds are greater than or equal to T_N^*. Once N has received *InitREP* messages from its neighbour nodes, it combines these messages and chooses as its *initiator* the responding neighbour node which appears in the most received *InitREP* messages. A malicious node is unlikely to appear in *InitREP* messages generated by honest neighbour nodes. Hence, given our assumption that a majority of the nodes in the network are honest nodes, the probability that a malicious node will be chosen as an initiator is low.

3.2 Checking for Duplication of the Candidate Address

When an initiator A chooses a candidate IP address for a new node, it broadcasts an *Initiator_Request* message to check for duplication. In order to protect a requester-initiator scheme against a DoS attack caused by malicious nodes claiming the possession of arbitrary candidate IP addresses chosen by initiators, the trust model can be used to discover possible malicious nodes. If initiator A receives a reply message *Add_Collision* from an existing node (node B, say) indicating that B is already using the candidate IP address and B is not in A's *blacklist*, then A will react to this reply as follows.

A either maintains a trust value for B or can calculate one (if B is not a neighbour node). If A's trust value for B is greater than or equal to T_A^*, then A believes that the candidate IP address is already being used. Node A will then choose another candidate IP address and repeat the procedure to check for duplication. Otherwise, A deems B a malicious node. In this case, A adds B to its *blacklist* and ignores this *Add_Collision* message. A also broadcasts a *Malicious_Suspect* message about B to all other nodes. When it receives A's message, each node uses its trust value for node A (if necessary calculating it) to decide if it should ignore A's message. If A is deemed trustworthy, then it will calculate its trust value for B. If its newly calculated trust value for B is lower than its threshold acceptable trust value, then it adds B to its *blacklist*. As a result, misbehaving nodes will be permanently excluded from the network.

3.3 Dealing with Possible Address Collisions

Suppose node E detects a collision of IP addresses between two existing nodes. It will then send a message about the collision to both of them. Any node (node F, say) receiving such a message considers its trust value for node E (calculated if necessary). If $T_F(E) \geq T_F^*$, then F will assign itself a new IP address. Otherwise, F will keep its current IP address and add E to its *blacklist*.

3.4 Brief Analysis

The above protocol enhancements significantly improve the security of requester-initiator schemes. Only trusted nodes will be chosen as initiators for address allocation. Malicious nodes will be detected and isolated with the cooperation of other network nodes. The DoS attack caused by a malicious node claiming the possession of candidate IP addresses is prevented. Only minor computational costs will be incurred if the number of malicious nodes is small.

However, when a malicious node acts as a requester and simultaneously asks many initiators for IP address allocation, each initiator will treat this malicious node as a new node and will not be able to calculate a trust value for it (since there will no history of past behaviour on which to base the calculations). This kind of attack cannot be prevented by using the trust model approach described in this paper. Some other method outside the scope of trust modelling is required.

4 A Novel Trust Model

In the solutions described in section 3, we assume that there is a trust model by which the trust values between any two nodes can be calculated. Many methods have been proposed for trust modelling and management, see, for example [3, 10]. Unfortunately, none of them has all the properties discussed in Section 3. Thus, they cannot be straightforwardly adopted for use in our scheme. In this section we propose a trust model specifically designed to be used in this environment.

In our trust model, each trust value is in the range 0 to +1, signifying a continuous range from complete distrust to complete trust, i.e. $T_A(B) \in [0, +1]$. Each node maintains a *trust table* in which it stores the current trust value of all its neighbour nodes. When a new node joins a network, it sets the trust values for its neighbour nodes in its trust table to an initial value T_{init}, and dynamically updates these values using information gathered. A node computes its trust value for another node using one of the following two methods, depending on whether or not the other node is a neighbour node.

4.1 Calculating Trust Values Between Neighbour Nodes

If B is a neighbour node of A, then A calculates $T_A(B)$ based on the information A has gathered about B in previous transactions, using so-called passive mode, i.e. without requiring any special interrogation packets. Here we adopt the approach of Pirzada and McDonald [10] to gather information about neighbour nodes and to quantify trust. Potential problems could arise when using passive observation within a wireless ad hoc environment [8, 12]. The severity of these problems depends on the density of the network and the type of Medium Access Control protocol being used. This is an open issue which needs further research. In our paper, we assume that these problems will not occur.

Information about the behaviour of other nodes can be gathered by analysing received, forwarded and overheard packets monitored at the various protocol layers. Possible events that can be recorded in passive mode are the number and accuracy of: 1) Frames received, 2) Streams established, 3) Control packets forwarded, 4) Control packets received, 5) Routing packets received, 6) Routing packets forwarded, 7) Data forwarded, 8) Data received. We also use the following events: 9) The length of time that B has used its current IP address in the network compared with the total length of time that B has been part of the network, and 10) How often a collision of B's IP address has been detected.

The information obtained by monitoring all these types of event is classified into n ($n \geq 1$) trust categories. Trust categories signify the specific aspect of trust that is relevant to a particular relationship and is used to compute trust for other nodes in specific situations. A uses the following equation, as proposed in [10], to calculate its trust for B:

$$T_A(B) = \sum_{i=1}^{n} W_A(i) T_{A,i}(B)$$

where $W_A(i)$ is the weight of the ith trust category to A and $\sum_{i=1}^{n} W_A(i) = 1$; $T_{A,i}(B)$ is the situational trust of A for B in the ith trust category and is in the range [0,+1] for every trust category. More details can be found in [10].

Each node maintains trust values for its neighbour nodes in a trust value table, and regularly updates the table using information gathered. If a neighbour node moves out of radio range, the node entry in the trust table is kept for a certain period of time, since MANETs are highly dynamic and the neighbour node may soon be back in the range. However, if the neighbour node remains unreachable, then the entry is deleted from the trust table.

4.2 Calculating trust values for other nodes

If B is not A's neighbour node (as may be the case in multi-hop wireless ad hoc networks), A needs to send a “Trust Calculation Request” to B, and B returns to A a “Trust Calculation Reply”, which contains trust values for all nodes in the route along which it is sent back to A, as follows.

We adopt the *Route Discovery* method proposed in the Dynamic Source Routing Protocol (DSR) [7] to send A's Trust Calculation Request (TCReq) message to B. Node A transmits a TCReq as a single local broadcast packet, received by all nodes currently within wireless transmission range of A. The TCReq identifies the initiator (A) and target (B), and also contains a unique request identifier, determined by A. Each TCReq also contains a *route record* of the address of each intermediate node through which this particular copy of the TCReq has been forwarded. When a node receives this TCReq, if it is not the target of the TCReq and has recently seen another TCReq from A bearing the same request identification and target address, or if this node's own address is already listed in the route record, it is discarded. Otherwise, it appends its own address to the route record in the TCReq and propagates it by transmitting it as a local broadcast packet. The process continues until the TCReq reaches B.

When B receives the TCReq message, a route from A to B has been found. Node B now returns a Trust Calculation Reply (TCReply) to node A along the reverse sequence of nodes listed in the *route record*, together with a copy of the accumulated route record from the TCReq. In our trust model, the TCReply also contains a *trust value list* of the trust value for each node in the route record, as calculated by its predecessor in the route. That is, whenever a node in this route receives a TCReply, it forwards the message to the preceding node on the route and appends to the trust value list its trust value for the succeeding node on the route, i.e. the node which forwarded the TCReply to it.

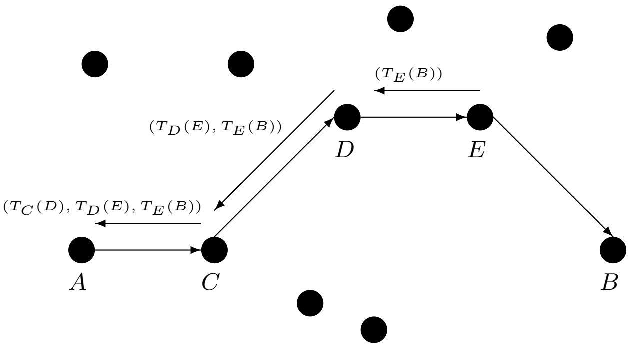

Fig. 1. A route from A to B

For example, as shown in Figure 1, when a TCReq is sent along the route $A \rightarrow C \rightarrow D \rightarrow E \rightarrow B$ and arrives at B, a route is found, and the route record is $ACDEB$. B will send a TCReply back to A along the route $B \rightarrow E \rightarrow D \rightarrow C \rightarrow A$. The immediate nodes E, D, and C append $T_E(B)$, $T_D(E)$, $T_C(D)$ to the trust value list respectively when they send back the TCReply. Therefore, when node A receives the TCReply, it will obtain a trust value list ($T_E(B)$, $T_D(E)$, $T_C(D)$); it will also have $T_A(C)$ from its own trust value table. Thus, A finds a route to B and also trust values for all the nodes in this route.

Following the above procedure, A may find one or more routes to B. Since there might be malicious nodes present, A will check if each route is a *valid route*, i.e., if all the trust values in the TCReply message are above A's threshold acceptable trust value T^*_A. Observe that when a malicious node receives a TCReply, it can change any of the trust values in the trust value list it receives, which only contains trust values for succeeding nodes on the route. However, it cannot change the trust value of any other node for itself. Suppose the route received by A consists of nodes $A = N_0, N_1, \ldots, N_i = B$. If a certain trust value $T_{N_m}(N_{m+1})$ on this route is below T^*_A, then either node N_{m+1} is a malicious node or another node $N_k(k \leq m)$ is a malicious node and has intentionally changed the trust value list when it received the TCReply. Therefore, when A obtains the trust list, it will learn that there is at least one malicious node in the route, and this route is therefore regarded as an *invalid route*.

If there is no valid route from A to B, A cannot calculate its trust value for B. In order to prevent certain attacks (see below), A is obliged to regard B as a potentially malicious node. Given that malicious nodes are rare, this situation is unlikely to occur frequently. If there does exist at least one valid route from A to B, then we can calculate the trust value of node A for node B based on the weighted average of the trust values of the nodes preceding node B on all valid routes for node B, and the weight of each route is based on the trust rating of all the intermediate nodes on each valid route. Suppose the route R is a valid route from A to B, denoted by:

$$N_{0,R} \rightarrow N_{1,R} \rightarrow N_{2,R} \rightarrow ... \rightarrow N_{i-2,R} \rightarrow N_{i-1,R} \rightarrow N_{i,R}$$

where $N_{0,R} = A$ and $N_{i,R} = B$. A can then calculate the *trust weight* for route R by computing the geometric average of the trust values listed in the TCReply in route R (all these trust values are above T_A^* given that R is a valid route):

$$W_{A,B}(R) = \prod_{j=0}^{i-2} T_{N_{j,R}}(N_{j+1,R})^{\frac{1}{i-1}}$$

A's trust value for node B is computed as the weighted arithmetic average of the trust values of A for B on all valid routes, $R_1, R_2, \ldots, R_g$, say.

$$T_A(B) = \frac{1}{\sum_{h=1}^{g} W_{A,B}(R_h)} \sum_{h=1}^{g} (W_{A,B}(R_h) T_{N_{i-1,R_h}}(B))$$

Node A only calculates its trust value for B when needed, and A will not store this trust value in its trust table. This is particularly appropriate for highly dynamic ad hoc networks.

4.3 Analysis

An underlying assumption for the scheme described above is that a node is considered malicious if it does not adhere to the protocols used in the network. Under this definition, two types of malicious node can be identified. Firstly, a node may be malicious all the time, i.e. it will behave maliciously when interacting with other nodes for all types of network traffic; it may also be malicious in its behaviour with respect to the trust model by sending incorrect trust values to requesting nodes. This type of malicious node behaviour can be detected by its neighbours, and thus we can expect that an honest neighbour will maintain a low trust value for such a node.

Alternatively, a node can behave honestly for all network interactions, and only behave maliciously with respect to trust model functionality. In our scheme, malicious nodes of this type will not be detected by other nodes, and the calculation of trust values will potentially be affected by these nodes. This is therefore an example of a vulnerability in our trust model approach. Hence, in the analysis below, we assume that all malicious nodes are of the first type, and we can assume that a misbehaving node will be detected by its neighbour nodes.

The trust value of any node A for any other node B in the network can be calculated. We claim that the existence of a small number of malicious nodes will not affect the calculation of the trust values. First note that, if B is A's neighbour node, A calculates its trust value based on information it gathers itself, which will not be affected by the existence of malicious nodes. Otherwise, A calculates its trust value for B based on the trust values listed in the TCReply in one or more routes from A to B.

If there are malicious nodes in a route R from A to B, then there will be a unique malicious node $N_{k,R}$ 'closest' to A on this route − all nodes on this route that are closer to A can therefore be trusted not to modify the trust value list. Consider the route R:

$$A = N_{1,R} \to \cdots \to N_{k-1,R} \to N_{k,R} \to N_{k+1,R} \to \cdots \to N_{i,R} = B$$

where $N_{k,R}$ is malicious and $N_{k-1,R}$ is honest. When the TCReply message from B to A is forwarded to node $N_{k,R}$ by node $N_{k+1,R}$, node $N_{k,R}$ appends $T_{N_{k,R}}(N_{k+1,R})$ to the trust value list and send this TCReply message to node $N_{k-1,R}$. Since node $N_{k,R}$ is a malicious node, it may deliberately modify (lower or raise) some of the trust values in the trust value list, i.e. any of $(T_{k,R}(N_{k+1,R})$, $T_{N_{k+1,R}}(N_{k+2,R})$, $\cdots$, $T_{N_{i-1,R}}(N_{i,R}))$. Moreover, other malicious nodes on this route may collude and raise each other's trust values. However, $T_{N_{k-1,R}}(N_{k,R})$ should be very low, since the maliciousness of $N_{k,R}$ will be detected by its honest neighbour node $N_{k-1,R}$. When A receives the TCReply message, A will regard this route as invalid, and will not use this route to calculate its trust value for B. Hence all intermediate nodes in a valid route must be honest.

The number of malicious nodes that our trust model can tolerate varies depending on the network topology. Our scheme requires at least one valid route from A to B in order to calculate the trust value of A for B. If at any time, no valid route from A to B is found, A cannot calculate a trust value for B. In this case, if A regards B as an honest node, a malicious node B can attack our trust model by modifying the *route record* when B receives a request message, or by just ignoring this request message to prevent A from finding a valid route to B. Thus, as mentioned in Section 4.2, if A cannot calculate a trust value for B, then A must treat B as as a potentially malicious node. If this trust model is used in the scheme described in Section 3, A adds B to its *blacklist*. However, a MANET is highly dynamic. A can calculate its trust value for B and update its *blacklist* frequently as long as a new valid route from A to B can be found when the topology of the network changes. All honest nodes will adhere to our trust model and make sure that the TCReply messages are sent back correctly.

The model suffers from Sybil attacks, where a node fraudulently uses multiple identities. We can use other methods to prevent Sybil attacks; for example, it may be possible to use trusted functionality in a node to provide a unique node identity. A detailed discussion of such techniques is outside the scope of this paper. We also ignored the problem posed by malicious nodes impersonating honest nodes. This problem cannot be completely overcome without using origin authentication mechanisms. We assume that, in environments where impersonation is likely to be a problem, an authentication mechanism is in place, i.e. fake TCReply messages sent by a malicious node will be detected. Many protocols and mechanisms for authentication and key management are available [1]. However, providing the necessary key management to support a secure mechanism of this type is likely to be difficult in an ad hoc environment. As mentioned above, trusted functionality, if present in a device, may help with this problem. Possible solutions to these issues will be considered in future work.

5 Conclusion

This paper focuses on IP address autoconfiguration in adversarial circumstances. The main contribution of this paper is to use a trust model to provide a number of enhancements to improve the security of requester-initiator schemes for IP address autoconfiguration in an MANET. It also gives a new trust model which can be used in these enhancements. Nevertheless other possible trust models with different trust quantification methods can also be applied in our solutions, as long as they satisfy the properties described in Section 3.

References

1. Boyd, C., Mathuria, A.: Protocols for Authentication and Key Establishment. Springer-Verlag, June 2003
2. Fazio, M., Villri, M., Puliafito., A.: Autoconfiguration and maintenance of the IP address in ad-hoc mobile networks. In: Proc. of Australian Telecommunications, Networks and Applications Conference, 2003
3. Huang, C., Hu, H.P., Wang, Z.: Modeling Time-Related Trust. In Jin, H., Pan, Y., Xiao, N., Sun, J. eds.: Proceedings of GCC 2004 International Workshops. Volume 3252 of Lecture Notes in Computer Science., Springer-Verlag (2004) 382–389
4. Mezzetti, N.: A Socially Inspired Reputation Model. In Katsikas, S.K. et al. eds.: Proceedings of 1st European PKI Workshop. Volume 3093 of Lecture Notes in Computer Science., Springer-Verlag (2004) 191–204
5. Nesargi, S., Prakash, R.: MANETconf: Configuration of Hosts in a Mobile Ad Hoc. Network. In: Proceedings of INFOCOM 2002, Volume 2, IEEE 2002 1059–1068
6. Perkins, C.E., Royer, E.M., Das, S.R.: IP Address Autoconfiguration in Ad Hoc Networks. 2002, Internet Draft: draft-ieft-manet-autoconf-00.txt
7. Johnson, D.B., Maltz, D.A., Hu, Y.C.: The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks. (DSR). 2004, IETF Draft: draft-ietf-manet-dsr-10.txt
8. Marti, S., Giuli, T.J., Lai, K., Baker, M.: Mitigating routing misbehavior in mobile ad hoc networks. In Pickholtz, R., Das, S., Caceres, R., Garcia-Luna-Aceveseds, J. eds.: Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking, ACM Press (2000) 255–265
9. International Telecommunication Union: ITU-T Recommendation X.509 (03/2000), The directory – Public-key and attribute certificate frameworks, 2000
10. Pirzada, A.A., McDonald, C.: Establishing Trust in Pure Ad hoc Networks. In: Proceedings of the 27th Conference on Australasian Computer Science, volume 26, Australian Computer Society, Inc. (2004) 47–54
11. Thomson, S., Narten, T.: IPv6 stateless address autoconfiguration. Dec. 1998, IETF RFC 2642
12. Yau, P, Mitchell, C.J.: Reputation methods for routing security for mobile ad hoc networks In: Proceedings of SympoTIC '03, IEEE Press (2003) 130–137