Secure Autoconfiguration and Public-key Distribution for Mobile Ad-hoc Networks

Hongbo Zhou
Dept. of Computer Science
Slippery Rock University
Slippery Rock, PA, USA
hongbo.zhou@sru.edu

Matt W. Mutak
Dept. of Computer Science & Engineering
Michigan State University
East Lansing, MI, USA
mutka@cse.msu.edu

Lionel M. Ni
Dept. of Computer Science
Hong Kong University of Science & Technology
Hong Kong SAR, China
ni@cs.ust.hk

***Abstract*—Security is extremely important for the deployment of a Mobile Ad-hoc Networks (MANET) due to its openness to attackers, the absence of an infrastructure, and the lack of centralized administration. Most research efforts have been focused on secure routing protocols, the distributed certificate authority, and key distribution, while a few projects have focused on secure autoconfiguration. However, the importance of integration of a secure autoconfiguration and public-key distribution has been neglected. This paper presents a secure autoconfiguration and public-key distribution algorithm to achieve uniqueness of address allocation and secure public-key distribution when a new node joins a MANET, which provides the bootstrapping for building a distributed certificate authority (DCA) in the network where a trust relationship is absent.**

Keywords-autoconfiguration; public key distribution; MANET; security

I. INTRODUCTION

A Mobile Ad-hoc Network (MANET) refers to a wireless network consisting of mobile nodes where an infrastructure is absent. In such a network, each node functions as both an end node and router. It initiates connections to other nodes, and forwards packets for other nodes at the same time. Due to the abundance of mobile devices, the speed and convenience of deployment, and the independence of networking infrastructure, a MANET has many applications in the scenarios where it is costly, inconvenient, or impossible to build an infrastructure, such as search-and-rescue, battlefield, and "smart transportation".

Before the deployment of MANETs, there are many issues that are worth our research effort, among which security is extremely important. A MANET is vulnerable to all kinds of attacks due to the following reasons:

(1) In an open system, a malicious node can join and leave the network arbitrarily;

(2) The wireless link between two nodes is a broadcast channel, so the communication is vulnerable to eavesdropping;

(3) The assumption underlying the MANET is that all the nodes (or most nodes) cooperate to function properly. A malicious node can undermine routing fabrics and other services passively (by dropping the packets that need to be forwarded) or actively (by injecting false information into the network or altering the packets in transit);

(4) It is more difficult to identify the source of a message in the MANET than in the hardwired network because of the absence of an infrastructure and the lack of centralized administration.

Thus, a seemingly easy task may become difficult when encountered with attacks. One example is autoconfiguration. Although there have been several autoconfiguration schemes proposed for uniqueness of address allocation when a new node joins the MANET ([1] – [8]), none will work properly in an insecure environment. Therefore, some secure autoconfiguration algorithms were proposed ([9]-[12]) to defeat attacks on autoconfiguration.

However, difficulty arises from the integration of secure autoconfiguration and public-key distribution because of the dual roles of the IP address, which is used for both routing and identification. For instance, after node N joins the network, the association of its IP address and its public key must be announced at the same time of autoconfiguration; otherwise, a malicious node (say node M) will know node N's IP address and use that address to associate with its own public key for "man-in-the-middle" attacks.

To solve the problem, we proposed secure autoconfiguration and public-key distribution, namely the SA-PKD scheme in this paper. It guarantees the uniqueness of IP address allocation. At the same time, it distributes the public key of the new node to all (or most) members in the MANET. In the ideal situation, all the nodes will receive the binding of the public key and IP address from the new node. Thus, it can be used as a temporary certificate authority for the bootstrapping steps in building a distributed certificate authority ([13]-[15]), where a trust relationship is absent. However, it is tolerable for some members to miss the binding

This research was supported in part by the NSF Grants No. OCI-0753362 and CNS-0721441, Hong Kong RGC grant N_HKUST614/07, and HKUST Nansha Research Fund NRC06/07.EG01.

978-1-4244-5113-5/09/$25.00 ©2009 IEEE

from one new node, because the new member can prove its ownership of the identity after autoconfiguration in our scheme.

The paper is organized as follows. Section 2 gives a brief description about pre-existing secure autoconfiguration schemes. Our SA-PKD scheme is presented in Section 3. Section 4 analyzes the attacks on the SA-PKD scheme and demonstrates its invulnerability, which is supported by the simulation results in Section 5. Section 6 suggests future work and concludes the paper.

II. RELATED WORK

This section gives a brief description of four secure autoconfiguration schemes, three of which have been examined in [16]. We used the same nominations from [16], but include their weaknesses from our points of view.

A. *Self-authentication scheme*

In the self-authentication scheme [9] (which is an application of Cryptographically Generated Address [17]), a new node generates its public/private key pair randomly and then uses the hash value of its public key as the IP address. To detect address conflict, the new node broadcasts a Duplicate Address Probe message (whose role is similar to the Duplicate Address Detection message in [1]) throughout the MANET. The message contains a timestamp and some signed information to prevent replay attacks and IP spoofing attacks from a malicious node.

This method is simple and elegant. To verify a node's ownership of the public key, another node merely performs the same hash function on the public key and compares the hash value with the IP address. With this scheme, a certificate authority is not needed. However, such a tight relationship between the public key and IP address brings the following problems:

(1) The scheme limits one public/private key pair per node. However, a node usually needs two pairs of public/private keys: one pair for signing/verifying, and the other for encryption/decryption. If a node uses only one key pair, it is vulnerable to chosen ciphertext attack [18]. Thus, with the self-authentication scheme, a node is going to have two IP addresses, and thus some method is necessary to bind these two IP addresses;

(2) The change of one leads to the change of the other. For example, if a public/private key pair expires in the middle of the communication, the IP address needs to change accordingly. Similarly, if there is an address conflict after two MANETs merge, one node needs to change both its IP address and public/private key pair simultaneously;

(3) In the case that a MANET is connected to the Internet with a gateway, the private address of the mobile node in the data packets needs to be changed with NAT, thus the relationship between the IP address and public key does not hold any more.

B. *Challenge-response scheme*

The challenge-response scheme [10] is based upon the buddy system used in [5]. The procedures include two steps: the first step is authentication, the second is address allocation. Firstly, a new node uses its MAC address as the temporary address to send both its MAC address and public key to all its neighbors with one-hop broadcast, and then expects to receive a unique nonce encrypted with the pubic key from each of the neighbors. Once the new node decrypts the nonce, it increases it by one, signs the message, and sends it back to its neighbor. After the authentication, the neighbors will record the mapping between the new node's public key and MAC address. The new node then chooses a neighbor randomly as the address allocator. The allocator divides its address pool into halves and assigns one half to the new node. It is different from the scheme in [5] in implementation details that the nodes do not maintain the actual address pools, instead they keep only the pointers pointing to the previous and next used addresses in the pool, which will lessen the complexity in maintenance of the address range if a node leaves the MANET abruptly.

The scheme has two problems. Firstly, only one-hop broadcast is used in the announcement of the public key, and thus the public key is distributed to only the one-hop neighbors; secondly, if the allocator is a malicious node, it can assign a non-disjoint address pool to the new node, which will lead to address conflicts in the current and subsequent address allocations.

C. *Trust model scheme*

There are two secure autoconfiguration schemes based on a trust model. The one proposed in [11] is based upon the MANETconf algorithm [6]. It assumes that the number of malicious nodes in the MANET is small. Each node in the network maintains a trust value for each of its neighbors. The neighbor whose trust value is greater than or equal to a threshold is considered as a trustworthy node. For a remote destination node, the source node gathers the trust values along the path between the source and destination to calculate the destination's trust value. With the trust model, a new node chooses only a trustworthy neighbor as a requestor. The requestor chooses a random IP address for the new node, and broadcasts a DAD message to detect an address conflict. The requestor will ignore all the veto messages from non-trustworthy nodes to prevent DoS attacks. This scheme can be easily defeated by Sybil attacks [19] in which a malicious node can forge multiple non-existent identities. They can conspire to increase each other's trust value.

The other secure autoconfiguration scheme in [12] is based upon the buddy system in [5] and a threshold cryptography-based distributed certificate authority (DCA) in [13]. The scheme assumes that a DCA is available in the MANET when a new node joins the network. Before requesting a free IP address pool, the new node first needs to collect at least k partial certificates from its one-hop neighbors to form a full certificate. From then on, all the control messages can be

authenticated. The problems with this scheme are that firstly, at least k pre-configured DCA server nodes must be present in the MANET without autoconfiguration; secondly, because only one-hop communication is utilized by the new node to apply for partial certificates, the scheme also assumes that the new node must have at least k DCA server nodes as its direct neighbors; thirdly, if the DCA is built on-the-fly, it is vulnerable to Sybil attacks, as we illustrated in [15].

III. SECURE AUTOCONFIGURATION AND PUBLIC-KEY DISTRIBUTION

The public key of the new node needs to be distributed at the same time as the secure autoconfiguration. Otherwise, a malicious node can impersonate the new node in registering or distributing the public key. This section presents the SA-PKD scheme that achieves two goals: uniqueness of address allocation and secure distribution of the public key.

A. Network model

We assume that the MANET is a densely connected network, in which there are multiple paths between any two nodes. Other scenarios, such as partitioning of the network, are studied in subsection III.C.

Ideally, there is a path that contains no malicious node between the new node and each of the members. However, even if there is a malicious node on the path, since our scheme is going to use multi-hop broadcasts to distribute encrypted and signed information, each node is monitored in forwarding packets to detect message modification, as illustrated in Fig. 1.

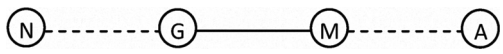

Figure 1. Figure 1. A path between new node N and member A

In Fig. 1, there is a malicious node M between the new node N and a member, node A. We assume that node M's direct upstream neighbor node G is a good node. Because broadcast is used in data communications, if node M modifies the control message, node G will receive the modified copy. Node G can move around or increase its transmitting power and forward the control message again, trying to reach the nodes beyond node M. In the end, node A will receive both authentic and modified control messages. Node A needs to keep both messages for verification.

If node M drops the control message silently, it seems to node G that node M leaves the network or moves away. If there is more than one path between the new node and the member, the control message can arrive at the member along other paths. However, if there is only one path, node A will not receive any message. To solve the problem, we resort to periodic HELLO messages in routing protocols [20]. To maintain routing fabrics, the interval of HELLO message broadcast is quite small (1 second for AODV). If we require that the control message be repeated several times, and that its interval be longer than that of HELLO messages, node G will be aware of the malfunction of node M. Therefore, it can move around or increase its transmitting power and forward the control message again.

When there is only one path from the new node to a member and there are two consecutive malicious nodes along the path, these two malicious nodes may conspire to tamper with the control message without being detected by the upstream node. Since we assume a densely connected network and nodes are free to move arbitrarily, this is very unlikely to happen. Moreover, according to the analysis in Section 4, the attack coordinated by these two malicious nodes cannot substitute the new node's public key, which alleviates the effect of the attack.

B. Procedures

The procedures of SA-PKD scheme include the following steps:

1) Generation of parameters: We based our SA-PKD scheme on [1], in which the new node chooses its IP address randomly. To be more specific, the new node, node N, generates the following parameters on its own:

(1) A public/private key pair (Pb_N / Pr_N);

(2) A random number (R_N);

Node N can generate more than one pair of public/private keys, which is similar to the steps described below. To generate its tentative IP address, it applies a hash function on R_N ($Addr_N$ = *Hash*(R_N)). The hash function can be a default hash function, or one from a list of hash functions. Here we do not let the new node choose its random address directly, because the relationship between R_N and $Addr_N$ may be utilized in the proof of the ownership of the IP address after autoconfiguration.

2) Broadcast of Duplicate Address Detection (DAD) message: Like the scheme in [1], node N chooses a temporary random IP address from a special address pool to broadcast a DAD message several times to detect address conflict. This temporary address is used only for autoconfiguration. Once the address generated in step 1) is confirmed to be free, it will be discarded.

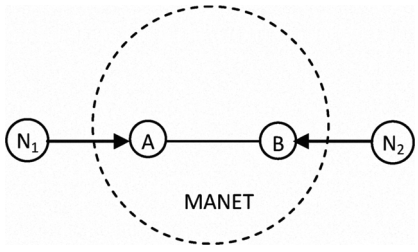

Figure 2. Two new node choose the same temporary address simultaneously

It is tolerable for two or more new nodes to choose the same temporary address simultaneously because our scheme uses application-level routing, as illustrated in Fig. 2. In Fig. 2, nodes N_1 and N_2 are both new nodes joining the MANET at the same time. Node A is a neighbor of node N_1, and node B is

a neighbor of node N_2. We also assume a direct link between nodes A and B for easy demonstration.

Suppose that these two new nodes choose the same temporary address (say x) in broadcasting the DAD message. In routing protocols, only one routing entry for x is saved in the routing table. For example, if node A receives the DAD message from node N_1 first and then that from node N_2, the next hop in the routing entry for address x will be node N_1 first and then will be updated to node B. Thus, if node A needs to send some reply message back to node N_1, the message will be sent to node B and then node N_2. In contrast, our SA-PKD scheme stores routing entries at the application level. Node A saves two entries for address x: one pointing to node N_1, the other pointing to node B. Both entries will be used to send a reply message to node N_1.Thus, node N_1 will not miss any reply message in the course of secure autoconfiguration. Although node N_2 will also receive the message destined for node N_1, we can always use sequence numbers, timestamps, or random numbers to differentiate between the reply messages.

In the DAD message, node N puts the following parameters in addition to a sequence number and a timestamp:

(1) The hash value of its IP address ($Hash(\text{Addr}_N)$, which is in fact $Hash(Hash(\text{R}_N))$);

(2) The IP address signed with its private key ($Sign_N(\text{Addr}_N)$).

For the reason of simplicity and readability, when calculating the parameters, we deliberately omitted the sequence number, timestamp, and some random numbers inside the hash function and signing function. For parameter (1), node N can use a different hash function. If it is not the default hash function, the new node also needs to notify other members within the DAD message the specific hash function used to calculate the hash value of the IP address.

The DAD message is sent with multi-hop broadcasts. As discussed in Subsection III.A, node N will know its direct neighbors by means of periodical broadcasts of HELLO messages, thus node N can monitor the forwarding of the DAD packet. If node N has only one neighbor that is a malicious node, and if the malicious node refrains from forwarding the packet, node N will not receive any forwarded copy in a limited time frame. Therefore, node N can move around or increase its transmitting power and try broadcasting again. If there is no forwarding at all after several trials, it can infer that it is the first node in the MANET and can perform self-configuration.

3) Receipt of Duplicate Address Detection (DAD) message: When a node receives the DAD message, it calculates the hash value of its own IP address. If it is the same as the hash value in the DAD message, it generates a veto message (an NACK message) and sends it back to node N. The most important parameter in the NACK message is the source IP address in the IP packet header, in addition to other parameters such as a sequence number and a timestamp. To prevent a malicious node on the path from dropping the NACK message, we resort to application-level routing as well: every node records the upstream neighbor on receipt of each copy of the DAD message, and sends NACK message to all the upstream neighbors.

It is trivial for a malicious node to forge an NACK message if it can find an IP address that has the same hash value as the new node generates. For 10.0.0.0/8 in IPv4, if we use MD5 hash function[1], with enough storage space (e.g., approximately 320MB for MD5), a malicious node can save a table of all the available IP addresses in this range together with their corresponding hash values. To solve this problem, either we use an address range in IPv6, or we allow many hash functions to choose from. For example, suppose that we have 16 different hash functions that all generate 128-bit results, the storage requirement increases to approximately 5GB. The third solution is to use the concept of seed in the encryption of a password in UNIX: a random number appended to the IP address in the calculation of hash value in step 2), which increases the storage requirement exponentially. For example, with a 2-bit random number, the storage requirement increases to 4×320 MB; with an 8-bit random number, it increases to 256×320 MB.

However, even if the malicious node can find an IP address that has the same hash value, the result is not as damaging as it seems. According to the nature of a hash function, there will always be more than one number mapped to the same hash value. Thus, even if the hash value is the same, the source IP address forged by the malicious node may not be the same as the IP address generated by node N. Therefore, the NACK message may be legally ignored as in step 6) below.

4) Forwarding of DAD message: Each member needs to forward the DAD message to its neighbors even if it has the same hash value of IP address and sends back a NACK message, as illustrated in Fig. 3. In Fig. 3, node N is the new node, node G is a good node, node M is a malicious node, and node A is another member. There is a direct link between node G and node M. Even if $Hash(\text{Addr}_N) = Hash(\text{Addr}_G)$, their addresses may still be different. Thus, node G still needs to forward the DAD message to other members, because node A may have the same address as node N. It is the NACK message from node A that prevents the actual address conflict.

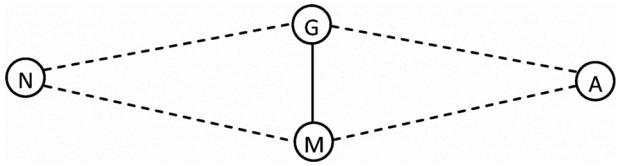

Figure 3. Forwarding of DAD packet

[1] Here we use MD5 hash function for illustration and simulation, which does not necessarily mean that MD5 should be used in the scheme. A more secure hash function, such as SHA-2, should be used in practical deployment.

Generally speaking, to prevent the same data packet from being forwarded multiple times by the same node, each node needs to check the Flooded Packet Identifier (FPI) introduced in [21] for each copy of the broadcast packets. For IPv4 packets, the FPI includes the source IP address, IP identification value, and fragment offset value, which are not enough for the SA-PKD scheme. For example, if node M in Fig. 3 modifies one parameter in the DAD message, it should be regarded as a different message. Thus, each node needs to check both the header and payload and forward the message if necessary.

5) Forwarding of NACK message: It should be noted that unlike broadcast of DAD messages, NACK message is forwarded with unicast. Thus, if a malicious node modifies the source IP address in the packet header, it will not be directly detected by the upstream node. Although in step 3), we suggest that the NACK message be sent through multiple reverse paths, if there is a malicious node on each path, they could conspire to replace the source IP address with another one.

However, the problem is not too serious either. Firstly, because the possibility that the new node chooses an occupied address is so low that it is very unlikely to happen, and thus the NACK message is not so important as the DAD message in step 2) and the Commit message in step 7); secondly, suppose that node A sends back a NACK message and that the NACK message is lost or modified, and thus node N cannot receive the IP address of node A, node N will keep broadcasting the DAD message with the same hash value for several times. Once node A receives the second DAD message, it infers two possible reasons:

(1) The NACK message is lost or modified;

(2) The NACK message is legally ignored by node N (as described in the next step).

In either case, better safe than sorry. Node A chooses to broadcast the NACK message. Thus, the modification of NACK message can be detected. If node A still receives the third DAD message with the same hash value, it can infer the second possibility.

6) Receipt of NACK message: Once node N receives the NACK message, it compares the source IP address with the address computed from R_N it chooses. If they are the same, node N chooses another R_N and repeats step 2) until there is no duplicate address or a limited number of retrials have been accomplished. For the NACK message whose source IP address is not the same as its own, it is legally ignored.

7) Broadcast of Commit (CMT) message: Node N will repeat broadcasting DAD messages several times to make sure none misses the message. Once DAD procedures finish, node N broadcasts its public key (Pb_N) in a Commit (CMT) message throughout the MANET. In addition to its public key, other parameters, including the sequence number and timestamp, will also be used. The source IP address of CMT message is now the hash value of R_N, the IP address that it generates. This address will also be used for all the subsequent data communications. The temporary address chosen for step 2) is discarded.

On receipt of the CMT message, each node uses the source IP address and the public key to verify the hash value of the address and the signature contained in the previous DAD message, and thus gets the association between the public key and the IP address of the new node. Each node also forwards the CMT message to its neighbors, just like the forwarding of DAD messages. In addition, node N also broadcasts the CMT message several times.

After SA-PKD scheme finishes, node N will have a unique IP address, and all the other members have the association of its public key and its IP address.

C. Other scenarios

If there is a partition in the network, the members in one partition will be unaware of new nodes' IP addresses and public keys in the other partition. Once these two partitions merge, a malicious node in one former partition can initiate "man-in-the middle" attack against another node in the other former partition. Thus, we require the merger of two partitions be viewed as the merger of two independent MANETs.

To solve this problem, we use the same concept of Network ID (NID) from other autoconfiguration schemes. Every MANET has a unique NID, which is updated by the new node. The new node generates a random number for NID and puts NID in the CMT messages. On receipt of the CMT message, each node updates its NID. If two or more nodes join the MANET or partition at the same time, the NID chosen by the node with the largest IP address will be adopted by all the members. The NID is also piggybacked in periodic HELLO messages to detect merger of two MANETs. If there is no new node joining any partition, the partitions are going to have the same NID and no further action is necessary. Otherwise, they are going to have different NIDs.

If two MANETs merge, we require that the nodes in one network join the other and perform SA-PKD procedures again. Although the IP address will be changed, with the measures from [22], the communication overhead can be minimized.

IV. Analysis of Attacks on SA-PKD

This section analyzes the attacks focused on SA-PKD scheme. Other attacks, such as repeated forwarding of modified DAD or CMT messages to consume computation and power resources, are not directly related to our scheme, and thus are skipped in our analysis. We also ignore replay attacks since we include sequence numbers, timestamps, and random numbers in the messages.

The DAD message contains two of the most important parameters: the hash value of the IP address and signed IP address. Because the random number (R_N) and the public key (Pb_N) are kept secret in the beginning by node N, none can

determine the IP address and public key from these two parameters. As a matter of fact, these two parameters seem like two random bit strings to all the other members.

To attack the SA-PKD scheme, the malicious node M needs to modify these two parameters in the DAD message. Node M has two typical choices:

(1) Node M chooses another public/private key pair ($Pb_{N'}$ / $Pr_{N'}$) and another random number ($R_{N'}$), replace $Sign_N(Addr_N)$ with $Sign_{N'}(Addr_{N'})$ and $Hash(Addr_N)$ with $Hash(Addr_{N'})$, as illustrated in Fig. 4. We denote this modified message as Substituted Message;

(2) Node M replaces them with two random bit strings. The modified message is denoted as Random Message.

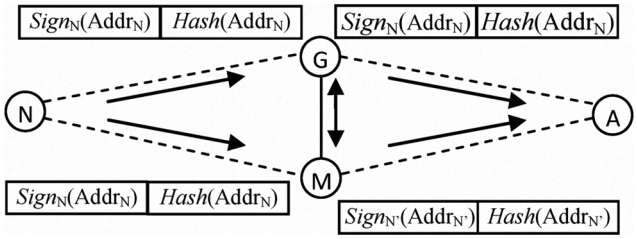

Figure 4. Attacks on DAD procedures for case 1

Node A will receive and record all the values for verification on receipt of the CMT message.

In the end, node N broadcasts a CMT message that contains two important parameters: $Addr_N$ and Pb_N. Now node M has three typical choices to modify the CMT message:

(1) Replace Pb_N with $Pb_{N'}$

Node A will receive both public keys and use them to verify the DAD messages and get the following results:

TABLE I. PROCESSING OF DAD PARAMETERS FOR CASE 1

CMT parameters	Previously received DAD message	Result
Pb_N	Authentic Message (G→A)	Node A recovers $Addr_N$, which is the same as the source IP address in CMT message, and hash value verification succeeds
	Substituted Message (M→A)	Node A cannot recover the address
	Random Message (M→A)	Node A cannot recover the address
$Pb_{N'}$	Authentic Message (G→A)	Node A cannot recover the address
	Substituted Message (M→A)	Node A recovers $Addr_{N'}$, which is not the same as the source IP address in CMT message, and thus verification fails
	Random Message (M→A)	Node A cannot recover the address

Node A will get the association of $Addr_N$ with Pb_N from the authentic messages.

(2) Replace ($Addr_N$, Pb_N) with ($Addr_{N'}$, $Pb_{N'}$)

Node A will receive both public keys and IP addresses, and use them to verify the DAD messages and get the following results:

TABLE II. PROCESSING OF DAD PARAMETERS FOR CASE 2

CMT parameters	Previously received DAD message	Result
($Addr_N$, Pb_N)	Authentic Message (G→A)	Node A recovers $Addr_N$, which is the same as the source IP address in CMT message, and hash value verification succeeds
	Substituted Message (M→A)	Node A cannot recover the address
	Random Message (M→A)	Node A cannot recover the address
($Addr_{N'}$, $Pb_{N'}$)	Authentic Message (G→A)	Node A cannot recover the address
	Substituted Message (M→A)	Node A recovers $Addr_{N'}$, which is the same as the source IP address in CMT message, and hash value verification succeeds
	Random Message (M→A)	Node A cannot recover the address

Thus, node A will get the association of $Addr_N$ with Pb_N from the authentic messages, and the association of $Addr_{N'}$ with $Pb_{N'}$ from the Substituted Message.

(3) Replace any parameter in CMT message with a random value

If node M replaces the public key with a random bit string, node A cannot recover $Addr_N$ with the modified CMT message. Similarly, if node M replaces the source IP address in CMT packet with a random IP address, the hash value of that random IP address is not equal to the saved copy of the hash value, and thus verification fails. However, with authentic DAD and CMT messages, node A can still get the association of $Addr_N$ with Pb_N.

Because DAD messages go before CMT messages, even if node M knows node N's IP address and public key on receipt of the CMT message, it cannot substitute node N's public/private key pair with another key pair used for the DAD message stored in other nodes. Thus, in any case, node N will get a unique IP address. At the same time, all the other nodes will always get the correct association of $Addr_N$ with Pb_N, and probably the association of $Addr_{N'}$ with $Pb_{N'}$, which seems like another new node joining the MAENT.

In the case that node A misses either DAD message or CMT message, node A cannot get the association of the new node's IP address and public key. In the case that two consecutive malicious nodes conspire to tamper with the DAD message and CMT message, node A cannot get the association from the new member either. However, from the analysis above, the malicious nodes cannot substitute the new node's

public key. Since the new node still keeps R_N secret, the new node still has the chance to prove its ownership of the IP address and make announcement of the binding of its IP address and its public key if necessary.

V. SIMULATION

We implemented the simulation of SA-PKD scheme in ns-2 (version 2.33) with CMU extension for ad hoc networks [23]. We ran the simulations in both secure and insecure environments. In the latter, we increased the percentage of malicious nodes to examine its invulnerability.

A. Simulation setup

The random waypoint mobility model is adopted in the simulation, in which all the nodes are constantly moving inside a square area. The maximum speed is 20 m/s, while the minimum speed is 5 m/s. The pause time is set to 0 second. The sizes of the area were adjusted to accommodate different sizes of MANET from 10 nodes to 50 nodes. Once the simulation starts, each node joins the MANET every 10 seconds. We let each node broadcast DAD messages and CMT messages three times in intervals of 3.0 seconds, which can be adjusted according to different applications.

To calculate hash values, we integrated the implementation of MD5 algorithm from [24]. For signing and verification operations, we implemented a simplified RSA algorithm. Because our scheme relies on multi-hop broadcasts and application-level routing, the simulation has no preference on underlying routing protocols.

B. Invulnerability

We let each node print out some debug information during the simulation, like that in Fig. 5:

```
(part 1)
Node 0 address:233 142 211 65
Node 0 public key: 25983 (e) 26329 (n)
Node 0 is configured!
Node 1 address:244 195 49 197
Node 1 public key: 11231 (e) 11461 (n)
Node 1 is configured!
…
(part 2)
Begin node 0 key table:
        Node: 1 Address:244 195 49 197        Public
key: 11231 (e), 11461 (n)
        Node: 2 Address:246 164 11 27  Public key:
8735 (e), 8927 (n)
        Node: 3 Address:248 5 48 74    Public key:
4499 (e), 4681 (n)
        Node: 4 Address:71 223 222 220        Public
key: 1991 (e), 2171 (n)
        Node: 5 Address:246 229 195 201       Public
key: 2483 (e), 2641 (n)
        Node: 6 Address:202 217 183 51        Public
key: 2183 (e), 2279 (n)
        Node: 7 Address:181 223 58 217        Public
key: 9359 (e), 9593 (n)
        Node: 8 Address:188 239 154 244       Public
key: 1055 (e), 1157 (n)
        Node: 9 Address:226 11 229 140        Public
key: 13803 (e), 14101 (n)
End node 0 key table:
…
```

Figure 5. Debug information from a 10–node simulation in secure environments

Fig. 5 shows the debug information from a 10-node simulation in secure environments. After each node successfully configures itself, it prints out its IP address (IPv4 format) and public key (e and n), as node 0 and node 1 in part 1. In the end of a simulation, each node also dumps its key table that records other nodes' IP addresses and public keys. For example, in Fig. 5, node 0's table contains 9 entries since there is no malicious node present.

To simplify the work of matching each entry in each node's key table in part 2 with the IP address and public key that are announced by each node in part 1, we wrote a Perl script to analyze the debug information. The Perl script first builds a separate key table of IP addresses and public keys from part 1, and then checks each node's key table in part 2 for verification.

In insecure environments, we randomly choose nodes as malicious nodes that modify DAD messages and CMT messages, as node M in Fig. 3. For example, we choose node 3 as a malicious node in a 50-node simulation, and we get the debug information as in Fig. 6. In Fig. 6, because node 3 is malicious, node 0 has only one entry for nodes 1, 2, and 3, but two entries for all the subsequent nodes: one is authentic, and the other forged by node 3, such as nodes 4 and 5. However, node 0 still gets the correct association of IP address and public key of those nodes.

```
(part 1)
Node 0 address:233 142 211 65
Node 0 public key: 25983 (e) 26329 (n)
Node 0 is configured!
Node 1 address:97 221 4 239
Node 1 public key: 687 (e) 865 (n)
Node 1 is configured!
Node 2 address:57 76 16 213
Node 2 public key: 3995 (e) 4237 (n)
Node 2 is configured!
Node 3 address:22 194 123 44
Node 3 public key: 1043 (e) 1121 (n)
Node 3 is configured!
Node 4 address:87 238 137 52
Node 4 public key: 4047 (e) 4183 (n)
Node 4 is configured!
Node 5 address:97 137 145 58
Node 5 public key: 3743 (e) 3869 (n)
…
(part 2)
Begin node 0 key table:
        Node: 1 Address:97 221 4 239   Public key:
687 (e), 865 (n)
        Node: 2 Address:57 76 16 213   Public key:
3995 (e), 4237 (n)
        Node: 3 Address:22 194 123 44  Public key:
1043 (e), 1121 (n)
        Node: 4 Address:87 238 137 52  Public key:
4047 (e), 4183 (n)
        Node: 4 Address:147 66 79 237  Public key:
4679 (e), 4867 (n)
        Node: 5 Address:97 137 145 58  Public key:
```

```
3743 (e), 3869 (n)
        Node: 5 Address:217 226 164 132        Public
key: 55215 (e), 55687 (n)
…
```

Figure 6. Debug information from a 50–node simulation with one malicious node

We increased the percentage of malicious nodes from 2% to 4%, 8%, and 10%, and the Perl script showed that all the members in the MANET correctly get the associations of the new nodes.

VI. CONCLUSION

Based on the studies of secure autoconfiguration schemes proposed by other researchers, we proposed the SA-PKD scheme to achieve both uniqueness of IP address allocation and secure public-key distribution simultaneously for a densely connected MANET. According to theoretical analysis and simulation results, it is superior to other known secure autoconfiguration methods in terms of security and simplicity. It is especially suitable for an open system where a trust relationship is absent.

In the paper, we solved the problem that the new node can distribute its public key to all (or most) members securely in the MANET. There is still much work ahead. One direction is to solve the problem that old members can distribute their public keys to the new node securely. Another direction is to integrate SA-PKD with our proposed multiple-key cryptography based distributed certificate authority (MK-DCA) in [15] where SA-PKD provides the solution to initiate secure communications to build a DCA.

REFERENCES

[1] C. Perkins, J. Malinen, R. Wakikawa, E. M. Belding-Royer, and Y. Sun, "IP address autoconfiguration for ad hoc networks," draft-ietf-manet-autoconf-01.txt, November 2001 (work in progress)

[2] K. Weniger and M. Zitterbart, "IPv6 autoconfiguration in large scale mobile ad-hoc networks," In Proceedings of European Wireless 2002, Florence, Italy, February 2002

[3] N. Vaidya, "Duplicate address detection in mobile ad hoc networks," In Proceedings of the 3rd ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC'02), Lausanne, Switzerland, June 2002

[4] A. Misra, S. Das, A. McAuley, and S. K. Das, "Autoconfiguration, registration, and mobility management for pervasive computing," IEEE Personal Communication System Magazine, Vol. 8, pp. 24-31, August 2001

[5] M. Mohsin and R. Prakash, "IP address assignment in a mobile ad hoc network," In Proceedings of MILCOM 2002, Anaheim, CA, October 2002

[6] S. Nesargi and R. Prakash, "MANETconf: configuration of hosts in a mobile ad hoc network," In Proceedings of the 21st Annual Joint Conference of IEEE Computer and Communication Societies (INFOCOM 2002), New York, NY, June 2002

[7] H. Zhou, L. M. Ni, and M. W. Mutka, "Prophet address allocation for large scale MANETs," In Proceedings of the 22nd Annual Joint Conference of IEEE Computer and Communication Societies (INFOCOM 2003), San Francisco, CA, April 2003

[8] H. Zhou, L. M. Ni, and M. W. Mutka, "Prophet address allocation for large scale MANETs," Ad Hoc Networks Journal, Vol. 1, Issue 4, pp 423-434, November 2003

[9] P. Wang, D. S. Reeves, and P. Ning, "Secure Address Auto-configuration for Mobile Ad Hoc Networks," In Proceedings of the 2nd Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services, *2005* (MobiQuitous 2005), pp. 519-521, San Diego, CA, July 2005

[10] A. Cavalli and J.-M. Orset, "Secure Hosts Autoconfiguration in Mobile Ad Hoc Networks," In Proceedings of the 24th International Conference on Distributed Computing Systems Workshops (ICDCSW 2004), Tokyo, Japan, March 2004

[11] S. Hu and C. J. Mitchell, "Improving IP Address Autoconfiguration Security in MANETs Using Trust Modeling," In Proceedings of the 1st International Conference on Mobile Ad-hoc and Sensor Networks (MSN 2005), Wuhan, China, December 2005

[12] F. Buiati, R. Puttini, and R. D. Sousa, "A Secure Autoconfiguration Protocol for MANET Nodes," In Proceedings of the 3rd International Conference on Ad-hoc Networks and Wireless (ADHOC-NOW 2004), Vancouver, Canada, July 2004

[13] L. Zhou and Z. J. Haas, "Securing ad hoc networks," IEEE Network, Vol. 13, No. 6, pp. 24-30, 1999

[14] S. Čapkun, L. Buttyán, and J. P. Hubaux, "Self-organized public-key management for ad hoc networks," In IEEE Transactions on Mobile Computing, Vol.2, No. 1, January-March 2003

[15] H. Zhou, M. W. Mutka, and L. M. Ni, "Multiple-key Cryptography-based Distributed Certificate Authority in Mobile Ad-hoc Networks," In Proceedings of IEEE Global Telecommunications Conference (GLOBECOM 2005), St. Louis, MO, November 2005

[16] M. Taghiloo, J. Taghiloo, and M. Dehghan, "A Survey of Secure Address Auto-configuration in MANET," In Proceedings of the 10th IEEE International Conference on Communication Systems (ICCS 2006), Singapore, pp. 1-5, October 2006

[17] T. Aura, "Cryptographically Generated Addresses (CGA)," Network Working Group RFC 3972, March 2005

[18] B. Schneier, Applied Cryptography, John Wiley & Sons, Inc. 1996, New York

[19] J. Couceur, "The sybil attack," In Proceedings of the 1st Workshop on Peer-to-Peer Systems (IPTPS'02), Cambridge, MA, March 2002

[20] C. Perkins, E. M. Belding-Royer, and S. R. Das, "Ad hoc on-demand distance vector (AODV) Routing," Network Working Group RFC 3561, July 2003

[21] C. E. Perkins, E. M. Belding-Royer, and S. R. Das, "IP Flooding in Ad hoc Mobile Networks," draft-ietf-manet-bcast-00.txt, November 2001

[22] H. Zhou, M. W. Mutka, and L. M. Ni, "IP Address Handoff in the MANET," In Proceedings of the 23rd Conference of IEEE Communication Society (INFOCOM 2004), Hong Kong, China, March 2004

[23] K. Fall and K. Varadhan (editors), The ns Manual - the VINT Project, http://www.isi.edu/nsnam/ns/ns-documentation.html, June 2008

[24] R. Rivest, "The MD5 Message-Digest Algorithm," Network Working Group RFC 1321, April 1992