Implementing a Fully Distributed Certificate Authority in an OLSR MANET

D. Dhillon
RSA Security Inc.
Vancouver, BC, Canada
ddhillon@rsasecurity.com

T. S. Randhawa
Computer Systems Technology Dept.
British Columbia Inst. Of Tech.
Burnaby, BC, Canada
tejinder_randhawa@bcit.ca

M. Wang, L. Lamont
Wireless Networking Group
Communications Research Center
Ottawa, Ontario, Canada
louise.lamont@crc.ca

Abstract--OLSR (Optimal Link State Routing) is one of the four base routing protocols being considered for use with MANETs (Mobile Ad hoc Networks) by the IETF's MANET working group. OLSR belongs to the proactive class of routing protocols in which the connection setup delay is minimized at the expense of heavier control traffic load on the wireless channel. Existing IETF draft proposals on OLSR do not yet address security issues. Although a PKI (Public Key Infrastructure) based security is deemed more appropriate for MANETs including OLSR MANETs, care should be taken to ensure that such an infrastructure does not add to the already heavy control traffic load in OLSR and, as much as possible, the existing OLSR control packets are utilized to support such infrastructures as well.

In this paper we describe our approach in which a PKI is tightly coupled with an OLSR MANET at the network layer level and the OLSR control packets are leveraged to support various security related activities as well. We have implemented a fully distributed CA (Certificate Authority) and integrated it with an existing implementation of OLSRv4 (OLSR for IP version 4). Intricate details of our implementation are presented to develop insight into key aspects of the proposed solution.

Keywords—Security; Mobile Ad-hoc Networks; OLSR; Public Key Encryption; Distributed Certificate Authority.

I. INTRODUCTION

The Mobile Ad-Hoc Networks (MANETs) are autonomous systems of mobile nodes (handheld user devices) interconnected by wireless links. Instead of using fixed base stations or access points, intermediate mobile nodes in the MANET act as mobile routers to support connectivity to other mobile nodes that are out of each other's range. The mobile routers are free to move randomly and organize themselves arbitrarily. This inherent flexibility allows for ease of deployment. Originally conceived for mostly military purposes, the inherent flexibility that these networks offer is also appealing to various commercial applications such as convention meetings, electronic classrooms, and search-and-rescue etc. A side effect of this flexibility is the ease with which a node can join or leave a MANET. Lack of any fixed physical and, sometimes, administrative infrastructure in these networks makes the task of securing these networks extremely challenging. Novel schemes are desired to block unauthorized nodes from joining the MANET and to prevent authorized nodes from compromising the security of the MANET.

Most security mechanisms are either based on Symmetric Encryption or Asymmetric (Public Key) Encryption. In symmetric encryption the sender encrypts the message using a secret key and the receiver decrypts the message using the same key. The parties involved in communication thus must posses the same secret key before the communication can begin. The symmetric encryption requires that the communication channel over which the distribution of the secret keys takes place must be secure. The public key encryption schemes, on the other hand, depend on the use of two different, but mathematically related, keys. Each party generates a public/private key pair. The sender encrypts a message using the private key and the receiver uses the public key of the sender to decrypt that message. Again, the involved parties require each other's public keys before starting the communication. Public key encryption does not impose as stringent requirements as symmetric encryption on the communication channel used for the key exchange. Public key encryption schemes only require an authenticated channel, as opposed to a secure channel, for the distribution of public keys. A trusted CA (Certificate Authority) is typically deployed in the security infrastructure to validate the authenticity of the public keys. Before distributing its public key to the intended parties, a node requests the CA to authenticate its key. The CA then issues a digital certificate binding the public key (contained in the digital certificate) to that specific node. The CA uses its own private key to sign this digital certificate. Any other node with the authentic public key of the trusted CA can verify the certificate and thereafter use the public key of the node and be sufficiently sure that it indeed belongs to that node.

A PKI aided by a trusted CA appears to be the most promising solution for securing MANETs. However the ease with which nodes may join or leave a MANET obviates the fact that the CA functionality

needs to be distributed in the MANET rather than assigned to a single node. A single mobile node functioning as a CA will bring the entire MANET to a halt if it moves out of the MANET and also act as a single point of failure if it becomes compromised. Replicated CAs may be used to avoid this security bottleneck. A closer look at this option reveals that it is not scalable from administration perspectives and that it creates several points of compromise if any CA node is compromised. An ability to distribute the signing authority among a large number of MANET nodes in such a way that multiple trusted nodes are required to coalesce to sign a certificate would be ideal. Such paradigm is a natural adaptation of trust based security apparatus prevalent in communities to protect the integrity of their membership. Initially there are only a few trusted members that collectively authenticate any incoming node. Newcomers are issued certificates that are perhaps valid only for a short duration. The nodes that have recently migrated to a MANET are thus expected to request for renewal of their certificates very often. As the time progresses, the well behaving nodes are rewarded by extending the expiry of their certificates. The nodes that are observed to have violated any rule are denied renewal of their certificate thus disabling them from effectively participating in communications sessions or other MANET operations. The nodes that continue to behave well will eventually become trustworthy enough that they are also bestowed the responsibility to authenticate future incoming nodes in cooperation with other trusted nodes.

Shamir [1] proposed a secret sharing technique that allows a secret to be shared among a group of users in such a way that at least k out of n shareholders must coalesce to reconstruct the secret. Any group of less than k shareholders are not able to deduce the secret based on the knowledge of their own shares. This approach was considerably extended by Luo and Lu [2] and adopted to develop a fully distributed CA. Proactive secret sharing procedures were outlined, verified and analyzed by the authors in [3][4]. Although a PKI built upon a fully distributed CA and aided by a sound network monitoring system is quite appealing from MANET security perspectives, however, like any other distributed scheme, it requires exchange of additional messages between multiple participants to accomplish the same task as compared to a centralized approach. Any adoption of such security mechanism in MANETs thus must take into account the additional control traffic load that may get injected into the MANETs that are inherently bandwidth starved and prone to packet collisions due to contention. This especially is the case with proactive MANETs that have substantially heavier control traffic load on the wireless channels as compared to their reactive counterparts [5][9][10][11].

In this paper we present our approach to integrate a fully distributed CA in a proactive ad hoc routing protocol named OLSR (Optimal Link State Routing). IETF's MANET working group has identified OLSR as one of the four base routing protocols for use in ad hoc networks. The other three are AODV (Ad-hoc On-Demand Distance Vector), DSR (Dynamic Source Routing) and TBRPF (Topology Broadcast Based on Reverse-Path Forwarding) routing protocols. Our approach addresses the concerns of control traffic overload by tightly coupling the operations of a fully distributed CA at the network layer level. The existing OLSR specific packet types, identified in the IETF's draft proposal on OLSR, are used, as much as possible, to also support the proposed PKI. A real test-bed has been constructed in which the existing implementation of OLSRv4 [12] was utilized and a fully distributed CA was introduced. This is to our knowledge the first attempt to address the security issues of OLSR. The paper thus provides valuable insight by detailing the implementation and evaluation of the proposed approach.

Security in MANETs is an active area of research. Besides the work of Luo *et al*., a secure DSR protocol was proposed by Hu *et al.* in [6]. A secure AODV was presented in [13]. None of these approaches employ threshold cryptography and hence do not render a self-securing MANET. MOCA [8], on the other hand, employs threshold cryptography but implements only a partially distributed CA. As it is always possible to compromise any authentication system, a sound network monitoring system is always a necessity to facilitate blacklisting nodes that are not well behaving. Such a network monitoring system has been proposed by Marti *et al.* in [7]. We haven't yet incorporated a monitoring system in our implementation and thus this aspect is part of our future considerations.

The rest of the paper is organized as follows. In Section II we provide a brief summary of OLSR, its control packets and control operations. We identify various security attacks that can impact the integrity of OLSR, and point out criterion for defending against these attacks. In section III an overview of threshold cryptography is provided to lay proper foundation for Section IV that details our approach to integrate a fully distributed CA in OLSR. Finally, Section V makes some conclusions and, once again, highlights the main contributions of this work.

II. THE OLSR PROTOCOL

Consider a MANET in which all nodes use public key encryption. A MANET node *vw* that wants to

send a message to another MANET node *vz* will do so after encrypting the message using *vz*'s public key. The node *vz* will decrypt the message using its private key. If the public key that *vw* used was actually injected into the network by an intruder pretending to be *vz*, then the intruder will be able to decrypt the message using its own private key and will have all the information that the was intended for *vz*. Obviously *vw* needs to ensure that the public key indeed belongs to *vz*. Each node in the MANET, after producing a public/private key pair, is required to send its public key to the CA for signing. The CA will confirm the identity of the sender and then create a certificate indicating that the public key specified in the certificate belongs to the node identified in the certificate, and then digitally sign the certificate using its own private key. An expiry time is also indicated in the certificate. Any node *vz* that wants to receive information from another node *vw* must produce a signed certificate to convince *vw* that the public key specified in the certificate belongs to *vz* and that it's the rightful requester of the information. In short-term MANETs, for example a meeting room scenario, all participating nodes can exchange their public keys in the beginning. As each node now knows the public keys of every other participating node, it will ignore any additional keys injected in the network. Since an intruder cannot inject its own public key into the network and it does not have the private key of any other node it cannot eavesdrop. However, in long-term MANETs, for example a conference scenario that spans multiple days and the participants can come and go at will, such a one-shot key exchange is not viable. Moreover a frequent refreshing of the keys may also be needed to ensure the integrity of keys. A trusted CA is required to assure the authenticity of the keys and their rightful owners. Before we describe our approach to implement a trusted and fully distributed CA in OLSR, it is important to identify salient features of OLSR protocol and point out security loopholes to justify such CA.

Mobile ad-hoc routing protocols are typically classified as proactive or reactive. The reactive protocols such as AODV [9] and DSR [10], attempt to discover routes only on-demand by flooding a route request in the network. These protocols thus reduce control traffic at the expense of increased latency in finding the route to the destination. The proactive protocols on the other hand employ periodic messages to maintain topology information. Some messages are exchanged locally for neighbor discovery yet others are sent to the entire MANET to propagate the knowledge of topology among all the nodes. The latency in finding a route is thus minimized at the expense of heavier control traffic.

OLSR is a proactive protocol. Its main functionality is to construct a routing table for each node in the MANET. The OLSR protocol is a variation of the pure LSR protocol and is designed specifically for MANETs. The OLSR protocol achieves optimization over LSR through the use of MPR (Multi Point Relay) nodes. The MPR nodes are selected and designated by neighboring nodes. Unlike LSR, where every node declares its links, only MPR nodes declare links. Also, unlike LSR, where each node forwards messages for their neighbors, only MPR nodes forward messages for those neighbor nodes that selected them as a MPR node.

Each node selects its MPR set of nodes in a way that, through them, it can reach all of its two hop neighbors. A node learns about its one-hop and two-hop neighbors from its one-hop neighbors' HELLO messages. By exchanging HELLO messages, a node finds out which neighbors have chosen it as a MPR. The neighbors that select a node as MPR form that node's MPR Selector set. A TC (Topology Control) message is sent periodically by each MPR in the network to declare its MPR Selector set and is used in the construction of routing tables.

As the existing IETF's draft on OLSR [5] does not specify any security mechanism, numerous opportunities exist for intruding OLSR nodes (unwanted nodes that are running an OLSR daemon) to launch active or passive attacks. Passive intruders could eavesdrop on confidential information that is being exchanged over the wireless channels. Our PKI allows the encryption of sensitive data. An active attacker, on the other hand, can alter control packets or send incorrect control packets to compromise the integrity of the routing protocol. For example, an intruding node may broadcast its HELLO messages specifying neighbors that don't exist. This will allow the intruding node to become an MPR because of the way the MPR set is chosen. Alternatively, an active attacker may simply send TC messages claiming to be MPR for nodes it is not. As the network depends on the MPRs for routing services, an intruder that manages to become an MPR can easily launch a black-hole attack. The black-hole attack occurs when a compromised MPR simply drops some or all of the packets it was supposed to forward. By using digital signatures, we ensure the integrity and authenticate the sender of HELLO and TC messages to prevent such attacks. The authenticated channel for key distribution is realized using a trusted and fully distributed CA, as explained in the next section.

A replay attack can also occur when an attacking node listens to packets and then broadcasts the same packets. This attack is possible even when the packets are encrypted. A variation of the replay problem is the wormhole problem that involves 2 or more attacking

nodes. A bi-directional wormhole could be used to make another node appear in 2 places of the MANET. To address the replay attack will require determining if a message arrived too late. Replay attack is out of the scope of our current implementation and has not been addressed herein. However we are in the process of enhancing our MANET security infrastructure to defend against wormhole attacks.

PKI solutions address many OLSR security concerns and only require an authenticated channel for key distribution. This channel is realized by using a fully distributed CA. In the next section the theoretical foundation of a fully distributed CA is presented to provide proper background.

III. A FULLY DISTRIBUTED CA

Our fully distributed CA is based on an approach described by Luo and Lu in [2]. In this section, we briefly describe their approach and point out corrections to their original algorithms.

The CA is an RSA key pair with public key pk_{CA}, private key $sk_{CA,}$ and modulus N. In the fully distributed approach, sk_{CA} is distributed using Shamir's Secret Sharing method by embedding sk_{CA} as the root of a polynomial $f(x) = sk_{CA} + a_1x + \ldots + a_{k-1}x^{k-1}$. Each shareholder with a unique non-zero identity i receives a share $S_i = f(i) \bmod N$. With knowledge of at least k shares the polynomial can be evaluated by calculating:

$$f(x) = \sum_{i=1}^{k} S_i \cdot l_i(x) \bmod N$$

where $l_i(x)$ is the Lagrange coefficient defined as:

$$l_i(x) = \prod_{j=1, j\neq i}^{k} \frac{x-j}{i-j}$$

The secret sk_{CA} can be recovered by solving for $f(0)$.

Any coalition of k shareholders may sign a message by generating a message digest and encrypting it with their additive shares which produces a partial signature:

$$sign_i = digest^{P_i} \bmod N \text{ where}$$

$$P_i = S_i l_i(0) \bmod N$$

A candidate signature can be generated from k partial signatures as:

$$SIGN' = \prod_{i=1}^{k} sign_i \bmod n .$$

By applying the k-bounded coalition offsetting algorithm, a proper signature $SIGN$ (which is verifiable by pk_{CA}) can be recovered. [2] describes an incorrect algorithm, we correct it as:

$$Z := digest^{-N} \bmod N$$

$$j := 0, Y := SIGN'$$

$$\texttt{while } j < k \texttt{ and } (digest \neq Y^{pk_{CA}} \bmod N) \texttt{ do}$$

$$Y := Y \cdot Z \bmod N, j := j + 1$$

$$SIGN = Y$$

[2] also suggests an improvement where the signing coalition can be dynamically chosen by performing all of the Lagrange calculations on the requesting node instead of the serving nodes; thereby allowing the coalition to be determined after receiving any k replies. However, the authors make an incorrect assumption that the equations $digest^{S_i l_i(0) \bmod N} \bmod N$ and $digest^{S_i l_i(0)} \bmod N$ are equivalent. Through our implementation, we have found that when using this method there is no guarantee that the candidate certificate will coalesce within k offsets. Because of the increased computational complexity, this alternative method is impractical with large RSA keys. We have notified the authors who have acknowledged this error.

It is also possible for k shareholders to serve a requesting node that has a unique and verifiable non-zero identity r, its own share by first generating partial shares:

$$S_{r,i} = S_i \cdot l_i(r) \bmod N$$

A share is thereafter generated by adding k partial shares:

$$S_r = \sum_{i=1}^{k} S_{r,i} \bmod N$$

When initializing a new node with its own share, the requesting node must not know the value of each partial share $S_{r,i}$ or else it can easily determine the serving node's share S_i as $l_i(r)$ is publicly known. Therefore, an extra round of communication is required to shuffle the partial shares $S_{r,i}$ before these are dispatched to the requesting node. Shuffling is a method where each serving node mathematically offsets its partial share in such a way that each partial share's original value is not recoverable but the sum of all partial shares is still equivalent to a full share.

IV. THE PROPOSED APPROACH

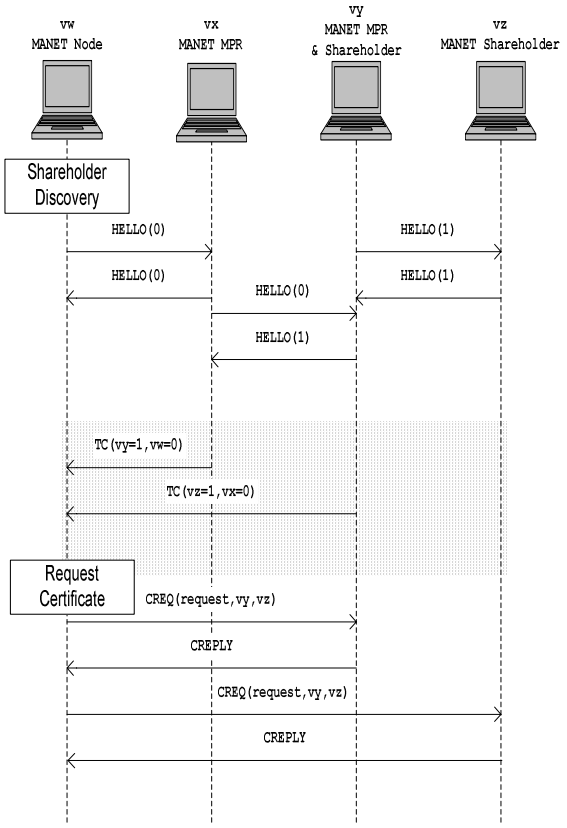

Figure 1. Certificate acquisition process in OLSR

We assume that the network is initialized with at least *k* shareholders. A shareholder can be any trusted OLSR node including MPRs. An incoming node needs to discover at least *k* shareholders from whom to request a certificate. As illustrated in Fig. 1, we have implemented two provisions for this using current OLSR control packets without increasing their length. The first is through HELLO messages. By setting a reserved bit in their HELLO messages nodes can indicate if they have partial shares and are willing to provide service. These modified HELLO messages only identify shareholders that are one hop away. Since there may not be at least *k* shareholders within the one hop distance of a node, we also use TC messages to notify the identity of shareholders. Each MPR uses its TC message to announce which nodes in its MPR selector set claim to be shareholders. These TC messages are propagated to the entire network and provide an effective way to inform every node of all the current shareholders. We modified the TC messages by using the reserved section to specify a count *C* of nodes in the MPR selector set, which claim to be shareholders; we further sort the list of Advertised Neighbor Addresses so that the first *C* are shareholders. When a node receives TC messages, it uses these TC messages to build its routing table and at the same time builds a table of shareholder nodes (maintaining their IP address, which we assume unique and verifiable, as well as their distance in terms of hop count).

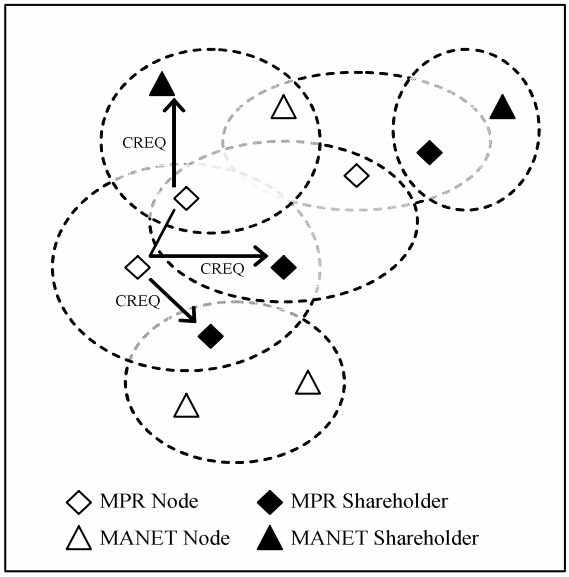

Figure 2. CREQ dispatch to shareholders

The node subsequently requests a certificate from any coalition of *k* shareholders, as shown in Fig. 2. Our certificate request was designed to conform with the X.509 CRMF standard [15] and is shown below.

Certificate Request

Certificate Request ID
Validity
Public Key Info
Proof of Posession (optional)

A node chooses a serving coalition of the k least costly (in terms of hop count) shareholders it is aware of, and sends a CREQ (Certificate Request) message to these serving nodes. If the node itself is a shareholder, it may choose itself as one of the members of the coalition. In the best case, a node will choose amongst its one-hop neighbors. In this case, the node may send out a single CREQ as a broadcast.

Upon receiving a CREQ message, each serving node determines whether it wants to serve this request. It must check that the requesting node is well behaving (not on any blacklists), and that it agrees with the specified validity date. Each serving node, which chooses to serve the request, will then generate an X.509 compliant certificate [14] based on the request. All shareholders generate exactly the same certificate; calculate the same digest; apply the same padding al-

gorithm [16]; and apply their shares to produce partial signatures.

X.509 Digital Certificate

Serial Number
Issuer Name
Validity Period
Subject Name
Public Key Information
Extensions
Partial Signature

The serving nodes return this certificate in a CREPLY message. The requesting node verifies the validity of the partial signature on the returned certificate using verifiable secret sharing techniques as described in [2]. If a serving node has returned an invalid partial signature, it is blacklisted and the process is repeated with a new coalition. If any serving node fails to reply, then the process must be repeated with a new coalition. Upon receiving k valid replies from a coalition, the requesting node extracts the partial signatures, adds them together and applies the coalition-offsetting algorithm to generate a proper signature, as described in the previous section. This proper signature replaces the partial signature in one of the returned certificates and the node now has a valid certificate signed by sk_{CA} and verifiable by pk_{CA}.

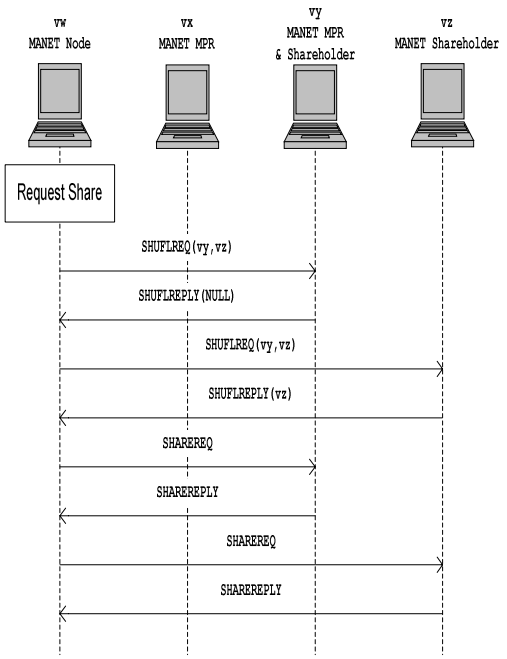

Figure 3. Partial share acquisition process

A node can also request its own partial share. As illustrated in Fig. 3, the requesting node initiates this process by sending a SHUFLREQ message to a coalition.

```
SHUFLREQ ::= {
        Serving IDs (v1, v2, …, vk)
        Digital Signature
}
```

Upon receiving a SHUFLREQ, each node in the coalition determines whether the requesting node should be issued a partial share. If the node agrees to authorize the requesting node, it will generate SHUFLFACTORS as necessary.

```
SHUFLFACTOR ::= {
        Recipient node ID,
        Shuffle factor (encrypted),
}
```

Each shuffling factor is a random integer value encrypted by the recipient node's public key. There is one shuffling factor required per node pairing in the serving coalition; the node with the smaller ID in each pairing generates the shuffling factor. Each node also keeps record of the shuffling factors it has generated until after it has finished generating the shuffled partial share. Each agreeing node replies with a SHUFLREPLY message.

```
SHUFLREPLY ::= {
        Serving node's ID,
        SHUFLFACTORs [0 -k],
        Digital signature
}
```

A SHUFLREPLY is sent, regardless of whether we generated any SHUFLFACTORs, to enable the requesting node to confirm that the entire serving coalition is still available. Once the requesting node has received all k of these SHUFLREPLY messages, it concatenates them without making any modifications to their contents and then sends each serving node this concatenated list as a SHAREREQ message.

```
SHAREREQ ::= {
        SHUFLREPLYs [k],
        Digital Signature
}
```

When a serving node in the coalition receives a SHAREREQ message it searches for shuffling factors that are addressed for it. This serving node will now generate a partial share $S_{r,i}$ for the requesting node and then shuffle it. The method we used is that the node which generated the shuffling factor will add it to its generated partial share and the node which is agreeing to the shuffling factor will subtract. Every serving node replies with a shuffled partial share. These replies must be encrypted otherwise an eavesdropping node can determine the share. The requesting node can produce its own share S_n by adding these k par-

tial shares. As each partial share was shuffled by adding and subtracting shuffling factors, the requesting node can not determine the value of the individual partial shares and, thus, can not determine any serving node's partial share. Similar to the certificate replies, a node verifies each share and black lists nodes that return improper shares. If any node fails to reply, the process must be started with a new coalition.

V. CONCLUSIONS

An efficient approach to integrate a fully distributed certification authority in OLSR is proposed. The proposed approach enables an OLSR MANET to autonomously self-secure itself without any external administration. The proposed approach minimizes the signaling overhead by supporting security at the network layer level. Existing OLSR control packets are utilized to help exchange security information thus avoiding additional traffic to support the aforementioned service. A real test-bed has been implemented to verify the viability of the proposed approach.

Future extensions of this work will include benchmarking the performance of the proposed approach using a heavily loaded and larger MANET; securing the MANET against wormhole attacks; and generalizing the proposed approach to accommodate additional MANET routing protocols.

REFERENCES

[1] A. Shamir, "How to Share a Secret", Communications of ACM 1979.

[2] H. Luo and S. Lu, "Ubiquitous and Robust Authentication Services for Ad Hoc Wireless Networks", Technical Report 200030, UCLA Computer Science Department 2000.

[3] J. Kong, P. Zerfos, H. Luo, S. Lu and L. Zhang, "Providing Robust and Ubiquitous Security Support for Mobile Ad-Hoc Networks, IEEE ICNP 2001.

[4] H. Luo, P. Zerfos, J. Kong, S. Lu and L. Zhang, "Self-securing Ad Hoc Wireless Networks", IEEE ISCC 2002.

[5] T. Clausen et. al. , "Optimized Link State Routing Protocol", http://www.ietf.org/internet-drafts/draft-ietf-manet-olsr-11.txt, July 2003.

[6] Y.-C. Hu, A. Perrig and D. Johnson, "Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks", MobiCom'02, September 2002.

[7] S. Marti, T. Giuli, K. Lai and M. Baker, "Mitigating Routing Misbehavior in Mobile Ad Hoc Networks", Proceedings of the 6th Annual International Conference on Mobile Computing and Networking, Aug. 2000.

[8] S. Yi and R. Kravets, "Key Management for Heterogeneous Ad Hoc Wireless Networks", IEEE ICNP'02, pp 12-15, Nov. 2002.

[9] C. Perkins, E. Royer and S. Das, "Ad Hoc On Demand Distance Vector (AODV) Routing", http://www.ietf.org/internet-drafts/draft-ietf-manet-aodv-13.txt, February 2003.

[10] D. Johnson, D. Maltz and Y. Hu, "Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)", http://www.ietf.org/internet-drafts/draft-ietf-manet-dsr-09.txt, April 2003.

[11] R. Ogier, F. Templin and M. Lewis, "Topology Dissemination Based on Reverse-Path Forwarding (TBRPF)", http://www.ietf.org/internet-drafts/draft-ietf-manet-tbrpf-10.txt, July 2003.

[12] L. Christensen and G. Hansen, "OLSR Routing Protocol", http://hipercom.inria.fr/olsr/, September 2003.

[13] M. Zapata, "Secure Ad Hoc On Demand Distance Vector Routing", ACM SIGMOBILE Mobile Computing and Communications Review, Vol. 6, Issue. 3, June 2002.

[14] R. Housley, W. Ford, W. Polk and D. Solo, "Internet X.509 Public Key Infrastructure Certificate and CRL Profile", http://www.faqs.org/rfcs/rfc2459.html, January 1999.

[15] M. Myers, C. Adams, D. Solo and D. Kemp, "Internet X.509 Certificate Request Message Format", http://www.faqs.org/rfcs/rfc2511.html, March 1999.

[16] B. Kaliski, "PKCS #1: RSA Encryption Version 1.5", http://www.faqs.org/rfcs/rfc2313.html, March 1998.