A Lightweight Mechanism to Secure OLSR.

Conference Paper · January 2006

Source: DBLP

3 authors, including:

Lin Chen
Université Paris-Sud 11
116 PUBLICATIONS **771** CITATIONS
SEE PROFILE

Jean Leneutre
Télécom ParisTech
40 PUBLICATIONS **302** CITATIONS
SEE PROFILE

A Lightweight Mechanism to Secure OLSR

Lin Chen, Xiaoyun Xue, Jean Leneutre
Département INFRES - CNRS LTCR-UMR5141
École Nationale Supérieure des Télécommunications
46, Rue Barrault, 75634 Paris Cedex 13, France
{Lin.Chen, Xiaoyun.Xue, Jean.Leneutre}@enst.fr

Abstract

Routing protocols in ad hoc networks are vulnerable to various attacks. Many solutions have been proposed to secure ad hoc routing protocols in recent research but at the price of significant traffic and processing overhead, which may be undesirable for the ad hoc networks with limited bandwidth and processing power. In this paper, we present and evaluate a lightweight mechanism to secure OLSR (Optimized Link State Routing [1]). In our approach, the mechanism of coherence check is applied to secure the link state update. To secure the neighbor establishment, we adopt the idea based on proof introduced in ADVSIG [2]. We identify a security flaw in ADVSIG and then propose a solution to counter it. Besides, we use the hash chain technique to reduce the processing overhead of HELLO messages in the neighbor establishment. Simulation results show that our approach is both lightweight and robust without degrading significantly the performance of OLSR.

Keywords: OLSR, Routing protocol, Security, Mobile ad hoc networks, One-way Hash chain

1 Introduction

Ad hoc networks are a new paradigm of wireless communication for mobile hosts (nodes). In an ad hoc network, there is no fixed infrastructure such as base stations or switching centers. Nodes that are within each other's radio range communicate directly via wireless links, while those that are far apart rely on other nodes to relay messages as routers.

A routing protocol in such networks finds routes between nodes, allowing a packet to be forwarded through other nodes towards its destination. Due do its nature, ad hoc networks are more vulnerable to various attacks than traditional wired networks. Unfortunately almost all of the widely used ad hoc routing protocols have no security considerations and trust all the participants to correctly forward routing and data traffic, which may be easily exploited by attackers to disrupt the routing protocol and disable communication

Recently, a number of solutions have been proposed to secure wireless ad hoc routing protocols [7], [9], [8]. Concerning OLSR, SOLSR proposed in [5] applies the wormhole detective mechanism and neighbor authentication to strengthen the neighbor relationship establishment of OLSR. The authors also use digital signature to protect the routing packets and hash chain to protect TTL and hop count. However, their approach cannot counter internal attacks. [4] proposes a mechanism using signature and timestamp to protect OLSR from external attacks. A more sophisticated signature mechanism ADVSIG [2] based on signed proofs is proposed to counter internal attacks. Although ADVSIG secures OLSR against both external and internal attacks, the traffic and processing overhead of performing a large number of asymmetric cryptographic operations to generate and verify proofs may significantly degrade the performance of OLSR.

In this paper, taking into account both the security and performance aspects, we propose a lightweight and robust mechanism to ensure the discovery of correct network topology in OLSR. The mechanism is efficient in terms of traffic and processing overhead and powerful to thwart various uncoordinated external and internal attacks. In the rest of the paper, Sec.2 gives an overview of OLSR followed by a discussion on the security issues of OLSR. In Sec.3, we present our approach in detail. Sec.4 and Sec.5 provide performance evaluation and security analysis on our approach. Sec.6 discusses several important issues of our approach. Sec.7 concludes the paper.

2 Security Issues in OLSR

2.1 OLSR Overview

OLSR is a proactive link state routing protocol for mobile ad hoc networks proposed by IETF MANET working group in 2003 (RFC 3626, [1]). It has two main optimizations over classical link state protocols.

Firstly, OLSR minimizes the flooding overhead of the routing control traffic by using only selected nodes to relay control messages. These nodes are called MPR (Multi-Point Relay) nodes and are selected by a node among its neighbors such that messages emitted by the node and relayed by the MPR nodes are received by all the 2-hop neighbors of the node. Secondly, OLSR requires only MPR link to be flooded to provide shortest routes. The MPR nodes declare their selector lists and flood (via Multi-Point Relay) them into the network.

The routing control traffic in OLSR is carried by two different types of messages: HELLO and TC messages.

Each node periodically broadcasts its one-hop neighbor list with the link state in HELLO messages. The link states declared in HELLO messages are LOST_LINK, ASYM_LINK, SYM_LINK, SYM_NEIGH and MPR_NEIGH, which denote respectively a lost link, an asymmetric link, a symmetric link, a symmetric neighbor and an MPR neighbor.

TC (Topology Change) messages diffuse topological information in which each node declares its MPR Selector Set (the list of neighbors who select the sender as MPR node). Upon receiving TC messages, each node updates its topology set and calculates its routing table using information in the topology set.

2.2 Security Issues

Security is not taken into consideration in the initial design of OLSR:

- OLSR does not protect the routing packets in networks, so an attacker can easily modify them without being detected.
- An attacker can forge incorrect routing control messages. Receivers of the forged messages will then select their MPR nodes and update their routing tables according to the incorrect topology information.
- There is no security mechanism for a node to distinguish an attacker from well-behaved nodes. Once being selected as MPR node, an attacker can create a black-hole which drops all or some of data packets from or to the selector or tamper them.

2.3 Attacker Model

Attackers may disrupt the operation of OLSR by exhibiting arbitrary malicious behavior: e.g., replay, forge, corrupt routing control messages to influence the topology view of benign nodes. These attacks can be classified as external attacks and internal attacks based on the information the attackers have. External attacks are launched by external attackers who do not have the cryptographic credentials (e.g., the keys used by cryptographic primitives) that are necessary to participate in the routing process. Internal attacks are launched by internal attackers who have compromised legitimate nodes, and therefore have access to the cryptographic keys owned by those nodes. Obviously internal attacks are far more difficult to detect and sometimes cannot be countered by pure cryptographic primitives. Our approach aims to protect OLSR against both internal and external attacks.

3 Our Approach to Secure OLSR

Mobile ad hoc networks are by nature heterogeneous environments where nodes range from laptops, handsets to PDAs. In such environments it is expensive or even prohibited to perform large number of asymmetric cryptographic operations. The situation is exaggerated in the ad hoc networks with limited bandwidth and processing power, where we argue that expensive and cumbersome security mechanisms in existing approaches such as ADVSIG may reduce the effectiveness of OLSR, and significantly consume network or node resources, leading to many new opportunities for possible Denial-of-Service (DoS) attacks.

Our motivation is to secure OLSR by adopting a lightweight and robust approach without degrading its performance. More specifically, our approach ensures authentication, integrity, non-repudiation of the routing messages and aims to satisfy the following requirements under condition that no colluding attack exists: (1) Routing messages cannot be spoofed; (2) Forged routing messages cannot be injected into the network; (3) Routing messages cannot be altered or replayed; We implement two strategies to achieve our goal: the secure link state update procedure protects TC messages from malicious alteration and disallows advertisement of forged links; the secure neighbor establishment procedure prevents declaration of forged neighbors and ensures that the 1-hop and 2-hop neighbor lists are correctly established.

3.1 Secure Link State Update Procedure

In our approach, to prevent a node from generating incorrect topology information in TC messages, the MPR relationship between two nodes AB is declared in the TC message of both A and B. If either of A or B does not declare the relationship, the relationship is regarded as invalid and is not used to update the routing table.

In standard OSLR, TC messages contain MPR Selector Set of the sender, the set of neighbor nodes selecting the sender as MPR node. In our approach, besides MPR Selector Set, the sender also declares its MPR Set, the set of nodes being selected by the sender as MPR nodes, in TC messages. The sender then adds the timestamp in the TC messages and signs them. In our approach all nodes generate TC messages.

Each node stores in its cache the TC messages received from all other nodes in last TC emission interval. Upon receiving a new TC message, the receiver checks the validity of the signature and the freshness of the timestamp. If both checks are successful, the message is accepted and the link state information in the message is further processed as follows (suppose node A receives a TC message from node B):

1 For all nodes I in the MPR Selector Set of B, A picks the received TC message or messages from I in the cache, checks if B is in the MPR Set, if so, the link BI is validated and added into the Topology Set of A.
2 For all nodes J in the MPR Set of B, A picks the received TC message or messages from J in the cache, checks if B is in the MPR Selector Set, if so, the link JB is validated and added into the Topology Set of A.
3 Node A then updates the cache.

Fig.1 illustrated the detailed procedure.

We compare the link state update procedure in our approach with that in ADVSIG, which use signed proofs to counter the declaration of forged MPR links. In ADVSIG, one signature verification operation is performed to verify the proof of each advertised MPR neighbor and one verification is performed to check the global signature of the message when processing a TC message, while our approach only needs to verify the global signature, hence significantly reduces the processing overhead of the link state update, especially in dense networks. However, our approach may cause a validation delay when network topology changes. In these cases, the MRP relation between A and B is validated after the processing of the TC messages from A and B. The impact of this delay on the performance of OLSR is further studied via simulation in Sec.5.

3.2 Secure Neighbor Establishment Procedure

This procedure secures the neighbor establishment of OLSR. Unlike other classic link-state routing protocols in which each node maintains a list of its 1-hop neighbors, OLSR needs the topology information of the 1-hop and the symmetric 2-hop neighbors. The symmetric two-hop neighbor list is used to select MPR nodes. Inspired by ADVSIG, we also use the mechanism based on proofs to prevent the declaration of forged links, especially forged symmetric links. However, two problems in ADVSIG should be addressed.

Firstly, we identify a security flaw in ADVSIG that may be exploited by attackers to forge symmetric links. In ADVSIG, if node A wants to declare a link of type λ with node B in its HELLO message, it should include in the HELLO message a certificate signed by B as a proof. The proof, obtained from the HELLO message previously broadcasted by B, is composed of the address of A, the type λ_p of the declared link AB and the timestamp, as shown in the following scheme.

- For λ = ASYM_LINK, no proof needed because B does not hear A.
- For λ = SYM_LINK, λ_p = ASYM_LINK or SYM_LINK.
- For λ = SYM_NEIGH or MPR_NEIGH, λ_p = SYM_LINK or SYM_NEIGH.

The following example[1] illustrates the establishment of a symmetric link between node A and B in ADVSIG by secured HELLO messages (Fig.2)[2].

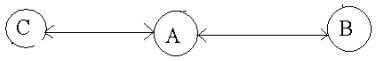

Figure 2: ADVSIG standard dialogue

1 $A \rightarrow *$ (B): $\{\phi,\ \phi,\ T_A(t_0)\}_{SA}$
2 $B \rightarrow *$ (A): $\{\{\text{“}A : ASYM\_LINK\text{”},\ T_B(t_1)\}_{SB},\ \phi,\ T_B(t_1)\}_{SB}$
3 $A \rightarrow *$ $(B,\ C)$: $\{\{\text{“}B : SYM\_LINK\text{”},\ T_A(t_2)\}_{SA},\ \{\text{“}A : ASYM\_LINK\text{”},\ T_B(t_1)\}_{SB},\ T_A(t_2)\}_{SA}$

[1]The example presented here differs from that described in [2]. However, our analysis in following sections holds in both cases.

[2]Throughout the paper, A → * (B): M indicates A broadcasts message M which is received by B. The format of HELLO messages in ADVSIG is {link state (certificate with signature), proof with signature, timestamp}, ϕ indicates no proof or certificate possible)

Secure Link State Update Procedure

Each node in the network buffers the received TC messages from every other node in last TC interval in RvdTcSet, where each TC message is stocked as RvdTcTuple (Src, MPR set, MPR Selector Set, V_Time) where src is the generator of the TC message, V_Time is the validity time when the tuple should be removed.

TC message generation: each node A includes its MPR Set, MPR Selector Set and the timestamp t in the TC message and signed the message with its private key S_A.

$$A \rightarrow *: < MPR\ Set,\ MPR\ SelectorSet,\ t >_{SA}$$

TC message processing: upon receiving a TC message from B, node A:

- Checks the validity of the signature and the freshness of the timestamp
- For each node I in MPR Selector Set of B:
  - Searches in RvdTcSet of A, find RvdTcTuple with Src = I
    - If B is in MPR set of I, update TopologySet with TopoTuple (I, B)
- For each node J in MPR Set of B:
  - Searches in RvdTcSet of A, find RvdTcTuple with Src = J
    - If B is in MPR Selector Set of J, update TopologySet with TopoTuple (B, J)
- Updates TopologySet of A
- Updates RvdTcSet with the received TC message

Figure 1: Secure Link State Update Procedure

At $T_A(t_0)$, node A broadcasts a HELLO message heard by node B. B, in its next HELLO message, indicates that it has heard A with the status "ASYM_LINK". Upon receiving HELLO message 2, A has node B's signature as proof that at time $T_B(t_1)$ there exists a link from A to B. In HELLO message 3 A declares its symmetric link with B with the certificate signed by B, $\{\text{“}A : ASYM\_LINK'', T_B(t_1)\}_{SB}$, received in message 2 as proof that both A and B claim to be able to receive HELLO messages from each other. Upon receiving the HELLO message 3 containing such proof, the receivers can be sure that node A is declaring valid and correct link information and select its MPR nodes according to the information.

However, since the HELLO messages are broadcasted to all neighbor nodes, the standard ADVSIG dialogue does not guarantee that there exists a symmetric link between A and B. Figure 3 shows a scenario where there is an asymmetric link $B \rightarrow A$ between A and B, and both A and B have an established symmetric link with D. In the scenario, B can forge the link $B \rightarrow A$ to be symmetric by performing the following attack to ADVSIG:

1 $B \rightarrow *\ (A) : \{\phi,\ \phi,\ T_B(t_0)\}_{SB}$

1' $D \rightarrow *(A, B)$: $\{\{''A : SYM\_LINK'',\ T_D(t_1)\}_{SD}$, $\{''D : SYM\_LINK'',\ T_A(t)\}_{SA}, T_D(t_1)\ \}_{SD}$

- On receiving message 1', B knows the existence of A and creates message 2.

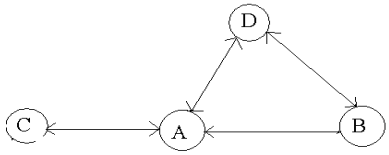

Figure 3: ADVSIG security flaw

2 $B \rightarrow *\ (A)$: $\{\{\text{“}A : ASYM\_LINK'',\ T_B(t_2)\}_{SB},\ \phi,\ T_B(t_2)\}_{SB}$

3 $A \rightarrow *\ (C, D)$: $\{\{\text{“}B : SYM\_LINK'',\ T_A(t_3)\}_{SA}$, $\{\text{“}A : ASYM\_LINK'',\ T_B(t_2)\}_{SB},\ T_A(t_3)\}_{SA}$

This scenario shows the security flaw in ADVSIG. The problem is that in HELLO message 2, no proof is required in the declaration of an asymmetric link, thus B can forge the message to make A believe that B can hear A by declaring the asymmetric link $A \rightarrow B$ although it does not exists.

In our approach, we counter the above security flaw by adding a proof in the declaration of asymmetric link. When B receives a HELLO message from A and wants to declare an asymmetric link $A \rightarrow B$, it should include the hashing value of the combination of the previously received HELLO message from A and the identity of B, $h(h(HELLO_A),\ B)$, as a proof. Upon receiving the HELLO message containing such proof, A can be sure that B can really hear it. Note that the identity of B is needed, otherwise the proof may be reused by an attacker to declare an asymmetric link with A. Other nodes receiving the

HELLO message containing such proof need not verify it because they only process symmetric link declaration between their one-hop neighbors and two-hop neighbors to update their MPR nodes. The following example illustrates the secure neighbor establishment in our solution (Fig.2):

1 $A \rightarrow *\ (B)$: $\{\phi,\ \phi,\ T_A(t_0)\}_{SA}$
2 $B \rightarrow *\ (A)$: $\{\{\text{“}A : ASYM\_LINK''$, $T_B(t_1)\}_{SB}$, $h(h(HELLO_A),\ B),\ T_B(t_1)\}_{SB}$
3 $A \rightarrow *\ (B,\ C)$: $\{\{\text{“}B : SYM\_LINK''$, $T_A(t_2)\}_{SA}$, $\{\text{“}A : ASYM\_LINK''$, $T_B(t_1)\}_{SB}$, $T_A(t_2)\}_{SA}$

The second problem we address is that in ADVSIG, a node should perform a large number of asymmetric signatures verifications to check the received proofs in the HELLO messages. Such operations cost substantial computation and power resources and may be too expensive or even prohibited in the networks where the processing power is limited. To address this problem, we use one-way hash chain elements as certificates and proofs in stead of digital signatures.

Now we are ready to present our secure neighbor establishment procedure.

Assumption

As ADVSIG, we assume that all nodes in the network are loosely synchronized. The time synchronization protocol proposed for ADVSIG in [3] can be applied in our approach. ADVSIG needs a PKI in which a server diffuses periodically the public keys of the nodes in the network. In our approach, besides the public keys, the server also distributes the authenticated Hash chain elements. The server estimates the upper bound of the number of nodes in the network N and the upper bound of the life cycle of the network T_{max}, which is further divided into n time intervals. T_{max} and the the network initiation time T_0 are published in the network. Since ad hoc networks are usually temporary local networks, it is not difficult to estimate the upper bound of the participants and life cycle of the network. In case where either of the upper bound is reached, the server needs to re-estimate N or T_{max} and redistribute the Hash chain elements.

Server initiation

At initialization, the server generates N^2 hash chains using N^2 random seeds. Each hash chain is labeled by a triplet $C(i,\ j,\ s)$ $(1 \leq i \leq N,\ 1 \leq j \leq N,\ 1 \leq s \leq 4)$. The kth element in $C(i,\ j,\ s)$ indicates that node i has a link with node j of status s at time interval $n-k$, where the value 1, 2, 3, 4 of s represent respectively the status ASYM_LINK, SYM_LINK, SYM_NEIGH and MPR_NEIGH.

Node bootstrap

Before entering the network, each node A needs to contact the server. The sever maps it to a set of Hash chains $C(i_A,\ j,\ s)$ $(1 \leq j \leq N,\ 1 \leq s \leq 4)$ and secretly communicates the seeds $s(i_A,\ j,\ s)$ of this set of Hash chains to A. The server then publishes the mappings between each node and the correspondent Hash chain set[3] as well as the last element of all the Hash chains $h^n(s(i,\ j,\ s))$ in the network.

HELLO message generation

In our approach, we replace the signed certificates and proofs in ADVSIG by Hash chain elements. For example, if A declares a symmetric link with B in time interval k, A includes the $(n-k)th$ element $h^{n-k}(s(i_A,\ i_B,\ 3))$ of Hash chain $C(i_A,\ i_B,\ 3)$ and k as a certificate in the HELLO message, required proof previously generated by B is also included. The global signature and timestamp are needed to protect the whole HELLO message and to prevent the replay attack.

HELLO message processing

On receiving a HELLO message, after the verification of the global signature and the freshness check of the timestamp, Hashing operations are performed to verify the proofs. To verify a proof $h^{n-k}(s(i_A,\ i_B,\ s))$, node A firstly checks k to see whether the proof is out of date, only proofs generated in recent time intervals (can be configured by the receiver) are accepted. If the proof is accepted, A then checks if $h^k(h^{n-k}(s(i_A,\ i_B,\ s)))$ equals to the published element $h^n(s(i_A,\ i_B,\ s))$, if so, the proof is valid. If the proof $h^{n-k+1}(s(i_A,\ i_B,\ s))$ corresponding to the same state of the same link in previous time interval has recently been received and stored, only one Hashing operation (to check whether $h(h^{n-k}(s(i_A,\ i_B,\ s))) = h^{n-k+1}(s(i_A,\ i_B,\ s)))$ is needed. Finally A updates its 1-hop and 2-hop neighbor lists.

The following example (Fig.2) illustrates the establishment of a symmetric link between A and B in our approach[4] (suppose the dialogue takes place within the time interval k):

1 $A \rightarrow *\ (B)$: $\{\phi,\ \phi,\ \phi,\ T_A(t_0)\}_{SA}$
2 $B \rightarrow *\ (A)$: $\{\{\text{“}A : ASYM\_LINK''\}$, $h^{n-k}(i_B,\ i_A,\ 1),\ k,\ h(h(msg\ 1),\ B),\ T_B(t_1)\}_{SB}$
3 $A \rightarrow *\ (B,\ C)$: $\{\{\text{“}B : SYM\_LINK''\}$, $h^{n-k}(i_A,\ i_B,\ 2),\ k,\ h^{n-k}(i_B,\ i_A,\ 1),\ k,\ T_A(t_2)\}_{SA}$

[3]The mapping can be a duet consisting of the identity of the node and the correspondent Hash chain index e.g. $(ID_A,\ i_A)$.

[4]The format of HELLO messages in the example is {link state, certificate, proof, timestamp}

Since a Hashing operation is 10^3 to 10^4 times lighter than a signature or verification operation, our solution significantly reduces the processing overhead compared with ADVSIG.

However, each node should store $4 * N$ Hash chains. We suggest using the mechanism proposed in [5]. Each node then only needs to store $O(N * 4 * log_2(n))$ Hash chain elements. In most cases $10 - 30$KB memory can meet the requirement. We argue that the requirement is reasonable and overpaid by the improvement in performance.

4 Security Analysis

Ideally, a secure routing protocol should be resistant against all potential attacks. In reality, given the highly dynamic nature of ad hoc networks and the different scenarios of their application, it is difficult even impossible to design a solution that can provide protection against all kinds of attacks in all possible application scenarios. In this section, we discuss the security properties of our approach under different attacks.

Our approach uses the signature and the timestamp in the routing messages to ensure authentication, integrity and non-repudiation of the routing messages, providing a defense line to the external attacks to forge or alter routing messages. The timestamp prevents the out-of-date routing messages from being replayed and injected into the network.

As for internal attacks, pure cryptographic operations cannot counter them. We adopt more sophisticated methods: we apply the coherence check and the proof check to ensure that internal attackers cannot advertise forged topology in TC and HELLO messages under condition that they do not collaborate. Any uncoordinated internal attacks are detected by the failures in coherence check or proof check. An attacker may refuse to declare the MPR links in the TC messages to isolate the victim nodes. We establish two mechanisms to counter this kind of "MPR link denial" attacks: (1) we take the advantage of the redundancy, the intrinsic nature of ad hoc networks to provide alternate routes to prevent the victim nodes from being isolated by the attack; (2) the victim nodes may change their MPR nodes when detecting the incoherence of MPR links in the received TC messages (e.g., the victim node is not in the MPR selector list of the attacker's TC messages though it selects the attacker as MPR node and declares so in the previous HELLO message).

Besides colluding attacks, another kind of attack called repeater attack is not addressed by our approach either, in which an attacker M behaves as a repeater by relaying all routing messages between two normal nodes A and B but dropping all or some of data packets. In fact this attack is not an attack to the routing protocol because from the topology's point of view, there does exist a link between A and B via a repeater M. Therefore the attack should be countered by other mechanisms such as watchdog or packet leash other than a secure routing protocol.

Attacks	Our approach	ADVSIG
Routing control message forging by external attackers	Yes	Yes
Routing control message forging by internal attackers	Yes	Not all
Routing control message alteration	Yes	Yes
Routing control message replay	Yes	Yes
Spoofing	Yes	Yes
Colluding Attack	No	No

Table 1: Security analysis (in the table, yes means the attack can be countered)

5 Performance Evaluation

In this section we implement our approach and evaluate its performance via simulation.

5.1 Simulation Setup

In order to evaluate the performance of the proposed mechanism, we implement it in NS-2.28 [1] and UM-OLSR [11] and carry out a set of simulations. The simulation field is $1500m * 300m$, where 30 nodes move around according to the random way-point model. The speed of the nodes is set uniformly distributed in [0, 2m/s] for low mobility case, [0, 5m/s] for medium mobility case and [0, 20m/s] for high mobility case. The traffic pattern is 25 random sessions (the source and destination are randomly chosen for each session) of constant bit rate (CBR) flow at a rate of 10 packets per second, and 512 bytes per packet in size. The hash function and digital signing function are MD5 (128 bits) and RSA (1024 bits). The TC and HELLO interval of OLSR are set to 5s and 2s.

In order to simulate the impact of signature and verification operations on the performance of OLSR, we add the correspondent delay when processing and sending messages. [6] provides a suite of benchmarks

of different cryptographic operations. We also take into consideration the heterogeneity of ad hoc networks in which nodes has different processing capacity. We thus attribute to every node a random processing time in [0.2ms, 150ms] for signing operation and in [0.1ms, 100ms] for verification operation.

We simulate the following three metrics to measure the performance of our approach and ADVSIG: control traffic overhead is the total overhead of TC messages and HELLO messages generated and relayed in the network including the correspondent security overhead; data packet delivery ratio is the ratio of the data packets generated by the CBR sources that are delivered to the destination; average end-to-end delay of data packets is the average delay between the emission of the data packet by the CBR source and its arrival at its destination.

5.2 Simulation Result

Control traffic overhead

Fig.4 compares the control message overhead[5] of original OLSR, ADVSIG and our approach. We can see that the overhead is significantly higher in ADVSIG and our approach compared with standard OLSR. In ADVSIG, this is mainly due to the signed proofs and certificates of the information in TC and HELLO messages. In our approach, the increase of the overhead is mainly due to the Hash elements in HELLO messages, the additional information of MPR Set in TC message and the additional generated TC messages. Compared with ADVSIG, our approach generates significantly less control traffic overhead.

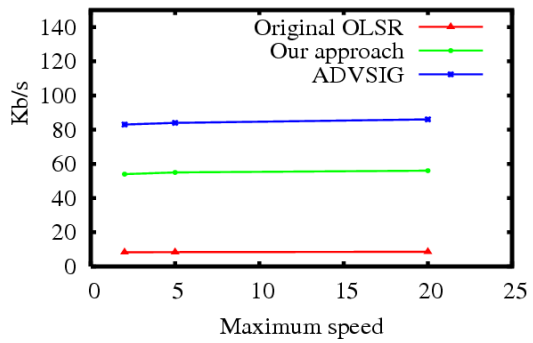

Figure 4: control message overhead

Packet delivery ratio

We also compare the delivery ratio of our approach

[5]In our simulation, MID and HNA messages are not taken into account since their overhead is negligible compared with the overhead of TC and HELLO messages

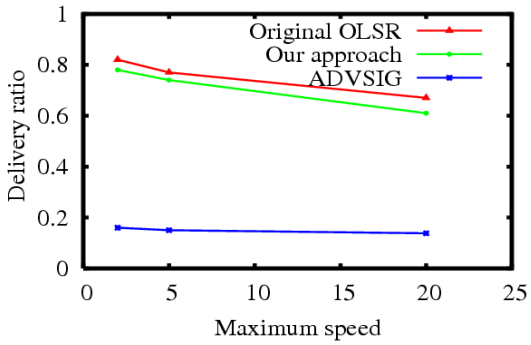

Figure 5: Packet delivery ratio

and ADVSIG. Compared with standard OLSR, our approach only degrades 5%-8% in term of delivery ratio, while ADVSIG degrades 70% of the performance of OLSR. A brief analysis on the processing of TC and HELLO messages explains the result:

TC message processing: in ADVSIG, one verification operation is needed to verify a proof, thus n verification operations of the proofs (n is the average number of MPR neighbor nodes, the typical value of n ranges from 5 to 7) and one verification of the global signature are performed when receiving a TC message. Therefore, in the network of M nodes, in a TC interval, a node should perform $(n+1)*M/2$ verification operations under the condition that on average half of the nodes generate TC messages. In our approach, only one verification operation is performed when receiving a TC message. Since all the nodes generate TC messages, a node performs M verifications in a TC interval. As a result, when the verification delay or M increases, the performance of ADVSIG drops much more rapidly than that of our approach.

HELLO message processing: in ADVSIG, $p+1$ verification operations of the proofs (p is the average number of proofs to be verified in a HELLO message, the typical value of p ranges from 4 to 8) and one verification of the global signature are performed when processing a HELLO message. In a HELLO interval (2s in our simulation), m HELLO messages should be processed (m is the average number of neighbor nodes), thus $m*(p+1)$ verification operations. For a node with the verification processing time of 50ms, the verification of signatures of HELLO messages takes on average 3.5 seconds, not to mention the processing of TC messages. As a result, it is not surprising to see the severe degradation of the performance. In our approach, only one or more Hash operations and one verification of the global signature is performed when processing a HELLO message. Performing substantially less verification operations of the signature, our

approach shows better performance.

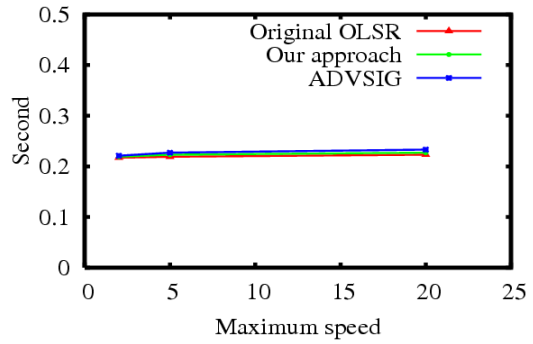

Figure 6: Packet delivery delay

Average end-to-end delay of data packets

Fig.6 compares the average end-to-end delay of data packets. Our approach slightly outperforms ADVSIG without interfering the performance of OLSR.

Performance under malicious attacks

We also evaluate our approach in hostile environment where 20% of the nodes in the network is compromised. They forge TC and HELLO messages to declare non-existent symmetric links with other nodes randomly selected from the normal nodes. Then they behave as black-holes by dropping all data packets. We can see from the simulation result (Fig.7) that the attacks cause detrimental effects on the performance of OLSR, which shows the necessity of building a secure routing protocol to guard against malicious attacks. The result of ADVSIG is not satisfactory either due to the large number of expensive cryptographic operations it performs. On the other hand, our approach is still able to achieve 50%-60% in terms of packet delivery ratio for different simulation configurations.

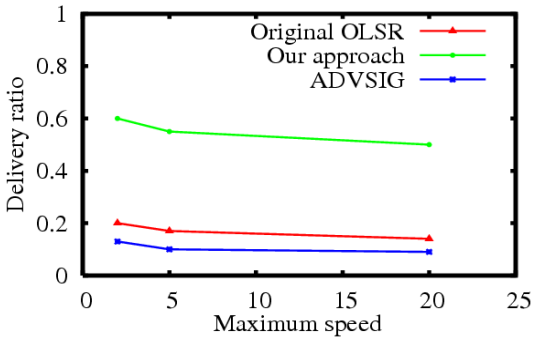

Figure 7: Packet delivery ratio under attacks

6 Discussion

6.1 Clock Synchronization

Clock synchronization is a common requirement in many solutions securing ad hoc networks such as ARAN and SEAD [7]. ADVSIG and our approach need a loose clock synchronization among nodes in the network. An ideal clock synchronization mechanism for mobile ad hoc network is distributed and does not depend on any specialized hardware. It is beyond the scope of this paper to address the clock synchronization in ad hoc networks. One point worth mentioning is that the synchronization mechanism itself should be secured in order to provide secured "real" time.

6.2 Collision Probability of Hash Chain Elements

In our approach, a collision of Hash chain elements in our approach may cause a security flaw that different states of different links correspond to one proof or certificate. Hereby we perform an analysis on the collision probability P_c: let A be the number of Hash chain elements in the element space, for m-bit Hash chains, $A = 2^m$; let N be the number of nodes in the network; let n be the length of a hash chain; let M be the number of Hash chain elements generated by the server, $M = 4 * N * N * n$. We thus have

$$\begin{aligned} P_c &= 1 - Prob(no\ collision) = 1 - \frac{C_A^M * M!}{A^M} \\ &= 1 - \frac{A!}{(A-M)!A^M} \\ &= 1 - \frac{A-1}{A}\frac{A-2}{A}...\frac{A-(M-1)}{A} \\ &= 1 - (1-\frac{1}{A})(1-\frac{2}{A})\cdots(1-\frac{M-1}{A}) \\ &< 1 - (1-\frac{M-1}{A})^{(M-1)} \\ &\sim 1 - (1-\frac{(M-1)^2}{A}) \sim \frac{M^2}{A}(M \ll A) \end{aligned}$$

With 128-bit Hash chains in an ad hoc network of 1000 nodes using Hash chain containing 10^6 elements, we have $Prob\ (collision) < 10^{-28}$, which can be regarded as negligible.

6.3 MPR Selection

In OLSR, MPR nodes are selected among neighbors according to their willingness. From the point of view of security, this may cause new vulnerabilities. A malicious node may show high value of willingness in order to be selected as MPR node but misbehave later

by not relaying data traffic. Therefore, it is necessary to take security into account in the MPR selection. One possible solution is to implement a watchdog in each node observing the behavior of its neighbor nodes. The result of the observation serves as a criterion in MPR selection. Mal-behaved nodes are excluded from being selected as MPR nodes

6.4 Redistribution of Hash chains

In our approach, the server needs to redistribute the Hash chains when the chains are used up and when the number of nodes in the network exceeds the estimated upper bound. In the formal case, all Hash chains should be redistributed while in the latter case, only the chains concerning new nodes need to be redistributed. We suggest overestimate the upper bound of the number of nodes and the life cycle of the ad hoc network to avoid frequently redistributing the Hash chains because the redistribution may lead to the reinitiation of the network.

7 Conclusion and Future Work

Today the security of ad hoc networks may become one of the bottlenecks of its potential applications especially in open environment. We argue that security and its impact on performance should be taken into consideration in the design of routing protocols for ad hoc networks. In this paper, we propose a lightweight and robust mechanism to secure OLSR.

The two key functions of OLSR is link state update and neighbor establishment. We propose to check the coherence of the declared MPR relation from both ends to secure link state update. Inspired by ADVSIG, we use proof-based mechanism to secure neighbor establishment. However, we address two problems in ADVSIG. Firstly, we identify a security flaw in ADVSIG which makes it vulnerable to link spoofing attacks. We then propose our solution to counter the security flaw. Secondly, we use one-way hash chain technique to reduce the processing overhead of HELLO messages. Compared with ADVSIG, our approach performs substantially less signature verification operations, thus shows better performance, especially in the networks with limited bandwidth and processing power. The only additional requirement is that each node needs to stock a number of hash chains. We argue that the price is reasonable and overpaid by the improvement in performance.

Our future work includes implementing different possible attacks and test our approach in a more hostile environment and improving our approach to counter colluding attacks.

References

[1] T. Clausen and P.Jacquet, "Optimized link state routing protocol (OLSR)," *RFC 3626, Experimental*, October 2003.

[2] D. Raffo, C. Adjih, T. Clausen, and P. Mühlethaler, "An advanced signature system for OLSR," *Proceedings of the 2004 ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN04)*, Washington DC, USA, October 2004

[3] C. Adjih, T. Clausen, P. Jacquet, A. Laouiti, P. Mhlethaler, and D. Raffo, "Securing the OLSR protocol," *IFIP Med-Hoc-Net*, Tunisia, 2003

[4] Fan Hong, Liang Hong, Cai Fu, "Secure OLSR", *In Advanced Information Networking and Applications*, 2005. AINA 2005. 19th International Conference on Volume 1, 28-30 March 2005.

[5] M. Jakobsson, "Fractal hash sequence representation and traversal," *IEEE International Symposium on Information Theory*, 2002

[6] D. Raffo, *Security schemes for the OLSR protocol for ad hoc networks*, Ph.D. dissertation, September 2005

[7] P. Papadimitratos, Z. Haas, "Secure routing for mobile ad hoc networks," *Proc. of 7th SCS Communication Networks and Distributed Systems Modeling and Simulation (CNDS)*, 2002

[8] Y.-C. Hu, D.B. Johnson, and A. Perrig. "SEAD: Secure Efficient Distance Vector Routing for Mobile Wireless Ad hoc Networks". *Proc. 4th IEEE Workshop on Mobile Computing Systems and Applications, Callicoon, NY*, June 2002.

[9] Y.-C. Hu, A. Perrig, and D.B. Johnson. "Ariadne: A Secure On-Demand Routing Protocol for Ad hoc Networks". *Proc. 8th ACM Conf. Mobile Computing and Networking (Mobicom02), Atlanta, USA*, 2002.

[10] NS2. http://www.isi.edu/nsnam/ns/

[11] UM-OLSR. http://masimum.dif.um.es/um-olsr/html/