A Formal Security Analysis of Secure AODV (SAODV) using Model Checking

A. Burak Gürdağ, M. Ufuk Çağlayan
Network Research Laboratory (NETLAB)
Department of Computer Engineering
Boğaziçi University, Istanbul, Turkey
Email: {gurdag,caglayan}@boun.edu.tr

Abstract—**In this paper, we present current state of our research on using model checking to analyze security properties of secure routing protocols for mobile ad hoc networks (MANETs). We provide a formal security analysis of Secure AODV (SAODV) using SPIN, a well-known model checker. First, we formally specify two security properties in the presence of an external attacker and model the protocol using PROMELA, the specification and modeling language of SPIN. Then, we present and discuss two attacks automatically found by SPIN and propose solutions to these attacks for some specific cases.**

I. Introduction

Mobile Ad Hoc Networks (MANETs) provide many challenges to the research community. Secure routing is one of them. There are many proposals in the literature that claim to provide certain levels of security in routing functionality [1], [2]. However, there are few works that formally analyze the security of these proposals.

When security is crucial for a protocol, the design should be proven to be secure. The most reliable way to do this is to deploy formal methods. Formal methods help us either to prove or to refute the correctness of a design. In this work, we use model checking approach. Specifically, we use PROMELA (PROcess MEta-LAnguage) as the specification and modeling language and SPIN (Simple PROMELA INterpreter) as the model checker [3]. Our primary aim is to demonstrate design flaws that lead to violations of security requirements using model checking. Model checkers are good at finding design errors and they provide error traces (i.e., counter-examples). In our case, an error trace demonstrates a successful attack.

In this study, we chose SAODV as the secure routing protocol to analyze [4]. SAODV is already known to be vulnerable to two attacks. We provide a formal model of SAODV and specify two security properties in PROMELA. We then use the SPIN model checker to demonstrate some attacks violating these properties and propose some solutions to thwart these attacks under certain conditions.

The rest of this paper is organized as follows. We provide a brief overview of SAODV in Sect. II. We introduce the SPIN model checker in Sect. III. In Sect. IV, we give formal and informal definitions of our security properties. In Sect. V, we explain our formal SAODV model. We give the verification results in Sect. VI. We give an overview of the related work from the literature in Sect. VII. Finally, in Sect. VIII, we discuss our work and present our future plans on this subject.

This work is supported by the Turkish State Planning Organization (DPT) under the project number 2007K120610.

II. Secure AODV (SAODV): An Overview

Secure AODV (SAODV) [4], [5] is an extended version of AODV protocol. It defines a set of message extensions to route request (RREQ), route reply (RREP) and route error (RERR) messages in AODV. There are also some new messages related to the detection of duplicate network addresses. In this work, we are only interested in route discovery. Therefore, we consider only mechanisms related to RREQ and RREP messages and omit those parts that involve RERR message, which is related to route maintenance.

In this work, we assume the links to be bi-directional. Therefore, we omit the use of RREP-ACK messages that are normally used to ensure the delivery of RREP messages in the existence of uni-directional links.

SAODV addresses integrity and authentication through digital signatures and hash chains. End-to-end digital signatures are used to sign non-mutable data in routing messages. This means only the initiators of RREQ and RREP messages (called *originator* and *destination*, respectively) sign the packets and intermediate nodes verify the signatures before forwarding them. This mechanism provides the authentication of originator and destination nodes and the integrity of non-mutable information in routing messages.

SAODV utilizes hash chains to keep the integrity of distance information, namely the hop count field, which is supposed to be incremented at each hop. This mechanism basically works as follows. When a node generates a RREQ or RREP message, it performs the following actions:

- Generate a random number called *seed*
- Set the maximum hop count field, d_m, to TTL value from the IP header. This is practically the expected diameter of the network.
- Initialize the hop count field, d, to zero.
- Initialize the hash field, h, to *seed*
- Calculate top hash field, h_{top}, by hashing *seed* d_{max} times, i.e.,

$$h_{top} \leftarrow H^{d_{max}}(seed)$$

where $H^n(x) = \overbrace{H(H(\cdots H}^{n \text{ times}}(x)\cdots))$ and H is the hash function.

When a node receives a RREQ or RREP, it performs the following actions:

- Verify the hop count field, d, by applying hash function to the current hash field $(d_t - d)$ times to see if the result equals h_{top}, i.e. ,

  $$d \text{ is VERIFIED if } H^{(d_{max}-d)}(h) = h_{top}$$

- If the message is to be re-broadcast, increment d and apply the hash function to h and store the result back into h, i.e.,

  $$d \leftarrow d + 1 \text{ and } h \leftarrow H(h)$$

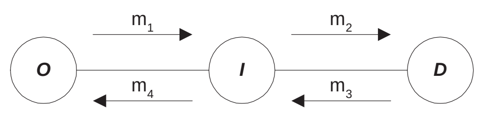
Note that the top hash field (h_{top}) is immutable and therefore it is included in the signature generated by the message initiator. This ensures the integrity of the field.

We use the following tuples to represent RREQ and RREP messages in SAODV:

{ *RREQ*, S, *reqId*, D, *SEQ*$_D$, O, *SEQ*$_O$, d_O, d_{max}, h_{top}, K_O, *SIG*$_O$, h }
{ *RREP*, S, D, *SEQ*$_D$, O, *SEQ*$_O$, d_D, *lifetime*, d_{max}, h_{top}, K_D, *SIG*$_D$, h }

where,

- *RREQ* and *RREP* are the message constants.
- $reqId$ is a number that uniquely identifies a RREQ message.
- S is the address of the neighbor node that sends the message. Although it is normally placed in IP header, we include it in the SAODV messages for the sake of clarity.
- D is the address of the destination node.
- *SEQ*$_D$ is the sequence number of the destination node as known by the RREQ originator.
- O is the address of the originator node.
- *SEQ*$_O$ is the sequence number of the originator node.
- d_i is the hop count that shows the distance that the RREQ or RREP message traveled from the message initiator so far. i can be O for RREQ or D for RREP.
- *lifetime* is the duration for which the RREP is valid.
- d_{max} is a constant that shows the maximum hop count expected in the network.
- h_{top} is a constant that shows the top hash value.
- K_O is the public key of the originator that is used to verify the signature field.
- *SIG*$_i$ is the signature of all fields except S, d_i, and h; i can be O for RREQ or D for RREP.
- h is the hash value to verify the hop count field.



m_1 (bcast) : { *RREQ, O, reqId, D, SEQ*$_D$*, O, SEQ*$_O$*,* d_{max}, h_{top}, K_O, *SIG*$_O$, h }
m_2 (bcast) : { *RREQ, I, reqId, D, SEQ*$_D$*, O, SEQ*$_O$*,* d_{max}, h_{top}, K_O, *SIG*$_O$, h }
m_3 (ucast) : { *RREP, D, SEQ'*$_D$*, O, lifetime,* d_{max}, h_{top}, K_D, *SIG*$_D$, h' }
m_4 (ucast) : { *RREP, I, SEQ'*$_D$*, O, lifetime,* d_{max}, h_{top}, K_D, *SIG*$_D$, h' }

Fig. 1. A Sample SAODV Messaging

A sample SAODV route discovery process is demonstrated in Fig. 1. Here, the originator node O initiates a route request for the destination node D by sending a RREQ message. Upon receiving RREQ message, the destination node initiates a route reply by sending a RREP message. Each node is supposed to check the signature and hash fields before processing an incoming message. If the message is to be forwarded, the hash field is updated after incrementing the hop count field.

SAODV has some weaknesses. First of all, there is nothing to prevent a node from increasing a hop count arbitrarily or from leaving it unchanged. The latter is already defined as a weakness in [5] since it helps a malicious node to attract traffic. We also think that the former can also lead to a situation that is desirable for an attacker. If there are at least one alternative to a route on which the attacker resides, it may consistently declare high hop counts so that this route is not selected. This can lead to a concentration of traffic on certain routes and to an excessive power consumption on certain nodes.

Besides the inability to detect incorrect hop count manipulation, SAODV has also another weakness. It does not do anything to protect the sender IP address field, S, which is used as next hop information in routing tables. This weakness can be used to disrupt routing operation. For example, a malicious node can impersonate (i.e. spoof) another node while forwarding a RREP, which causes incorrect routing information stored in the network. Such attacks potentially cause the nodes to consume excessive time and energy. The designers of SAODV deliberately left such denial of service (DoS) attacks out of scope [5].

III. The SPIN Model Checker

SPIN [3] is an automata-based temporal logic model checker. Its specification and modeling language is called PROMELA.

In PROMELA, a system is modeled as the composition of asynchronous processes that can interact with buffered or unbuffered message channels. Since there is no notion of time or clock, models with real-time aspects are very hard, if not impossible, to express in PROMELA. The language is especially designed to describe systems such as asynchronous communication protocols.

Correctness properties are specified in several ways in PROMELA. Assertions and *never* claims are the most frequently used constructs for this purpose. An assertion has a similar semantic as in the C Language. When the expression to be asserted is false, the assertion fails, and SPIN gives "assertion violated" error.

Never claims are used to specify the finite or infinite behavior that should never happen during the execution of a system. When we want to specify a property to be satisfied by the system, we formalize it in a logic formula and produce a *never* claim that corresponds to the negation of this formula. SPIN then tries to find a violation for this *never* claim. If it finds one, this means there is a case that the opposite of our property can occur in the system, which means our property can not be satisfied by the system.

A *never* claim can be written by hand or can be translated from a linear temporal logic (LTL) formula. SPIN also includes a timeline property editor that helps users visually specify properties that are otherwise hard to formalize.

SPIN has two modes of operation: simulation and verification. In simulation mode, it runs the model and helps users get an impression on how their model behaves and debug their model. In verification mode, SPIN analyzes the model against the properties considering all possible executions performing an exhaustive search on the state space. It can also perform partial search on the state space, which is quite useful in case of very large models or insufficient computational resources.

If SPIN finds a violation, it produces an error trace. Using this error trace, a user can run a simulation of the execution that leads to the violation.

Our primary aim in this work is to find security flaws in the design of SAODV protocol. We employ SPIN to check our protocol model against the security properties that we formally specify as never claims in PROMELA and to list any security flaws, if any, as violations.

We can make great use of SPIN by making proper abstractions and keep the complexity of our model low enough to be analyzed by computers.

IV. Formal Specification of the Security Requirements for SAODV

We use PROMELA as the formal specification language. We define our requirements as LTL (Linear Temporal Logic) formulae and convert these to *never* claims in PROMELA. In the following sections, all the formulae and related definitions are coded in PROMELA.

We consider two important properties (i.e. requirements) for the routing operation. The first one is the maintenance of correct distance information in the routing tables. The other one is the avoidance of routing loops. We want to show if these two properties are violated for SAODV in the presence of an external attacker. In case of a violation, SPIN will produce an error trace that gives the details of the attack.

A. Correctness of Distance Information

The requirement for the maintenance of correct distance information property can be informally stated as follows:

"It is always true that if there is a route from an originator node O to a destination node D, the length of this route known to node O is actually the shortest path from O to D."

The corresponding LTL formula for this property is as follows:

$$\Box(p \rightarrow q) \tag{1}$$

where p stands for the proposition *"There is a route from O to D"* and q stands for the proposition *"The length of this route known to O is actually the shortest path from O to D."* Here, $\Box$ is the "Always" (or "Globally") operator of LTL. We need to convert this formula to a *never* claim in PROMELA, which represents the behavior that should never happen. Therefore, we negate our original formula and convert it to a corresponding *never* claim by using the internal converter supplied with SPIN. The corresponding *never* claim for the correctness of distance property is as follows:

```
never {    /* !([] (p->q)) */
  T0_init:
    if
    :: (! ((q)) && (p))
           -> goto accept_all
    :: (1)
           -> goto T0_init
    fi;
  accept_all:
  skip
}
```

We formalize the propositions p and q as follows:

$$p \equiv V_D(O) \tag{2}$$
$$q \equiv (H_D(O) = D_D(O)) \tag{3}$$

where $V_j(i)$ is true if there is a route entry for destination node j in the routing table of node i; $H_j(i)$ is the minimum distance in hops to node j stored in the routing table of node i; $D_j(i)$ is the actual distance in hops from node i to node j.

B. Loop Freedom

A routing protocol is loop-free if its operation does not allow the formation of cycles on a route between two nodes. This definition implies that the length of a route can be at most $n-1$ where n is the number of nodes in the network. Based on this definition, we define the loop freedom property as follows:

"It is always true that if node O has a route entry for a destination node D then node D must be at most $n-1$ hops away from node O, where n is the number of nodes in the network."

Note that this particular definition of loop freedom is independent of the network topology. The corresponding LTL formula for the loop freedom property is of the following form:

$$\Box(p \rightarrow q) \tag{4}$$

where p stands for the "if" part and q stands for the "then" part of the property. Note that our *never* claim for this property is the same as the one in our distance property except the definitions of p and q. We formalize the propositions p and q as follows:

$$\begin{aligned} p &\equiv (X_D(O) \neq null) \quad (5) \\ q &\equiv (X_D(O) = D) \\ &\quad \vee (X_D^2(O) = D) \\ &\quad \vee \ldots \\ &\quad \vee (X_D^{(n-1)}(O) = D) \quad (6) \end{aligned}$$

where $X_D(O)$ is the next node from node O to node D and $X_D^2(O) = X_D(X_D(O))$ and so on. If $X_D(O)$ is $null$, this means there is no route entry for node D defined in the routing table of node O.

V. Modeling the SAODV Protocol

A. Assumptions and Simplifications on SAODV

Abstraction is the key process of a model checking attempt. We need to make certain assumptions and simplification on SAODV to produce a model that is both relevant to the properties we specify and simple enough to allow a feasible analysis by the model checker.

SAODV, by design, makes the following assumptions about a mobile ad hoc network [4]:

1) All nodes have a public-private key pair to produce digital signatures.
2) Nodes and their cryptographic belongings can not be compromised. This means that only external attacks are considered.
3) Attackers do not collude.

We made the following additional specific assumptions and simplifications to reduce the complexity of our SAODV model:

1) In this work, we used at most five nodes in a network. Therefore, it is enough to allocate three bits for the address field.
2) Each node has only one network interface and a unique IP address, which eliminates the use of interface descriptors and mechanisms to detect duplicate addresses in the model.
3) The network does not contain subnets and it is not connected to external networks.
4) Intermediate nodes are not allowed to reply route request. Normally, this is an option in SAODV to enhance protocol performance. Since we are interested in functionality, we can disregard this option. By not allowing intermediate replies, we eliminate the use of double signature extension messages in SAODV.
5) There is a single external attacker in the network with the same networking capabilities as a regular node has. Being an external attacker, it does not have and also can not obtain any cryptographic properties that the honest nodes have.
6) Topology does not change during the operation. This means, mobility, link breakages, and node failures do not occur and therefore route maintenance operations are not performed.
7) The cryptographic functions and algorithms used in the protocol are assumed to be secure.
8) Links are symmetric (i.e. bidirectional). This makes representation of node connectivity simpler. Since links are symmetric, routes are also symmetric. This assumption eliminates the use of RREP-ACK messages in our model.

B. SAODV Nodes

There are four different types of nodes in our model:

Originator (O)	:	initiates route discovery process to the destination node.
Destination (D)	:	creates and sends route reply (RREP) messages for route requests (RREQ) generated by the originator node.
Intermediate (I)	:	re-broadcasts route requests and forwards route replies to the other nodes.
Attacker (A)	:	behaves much like an intermediate node except that it does not increment hop count and it nondeterministically changes sender address field in route reply (RREP) and route request (RREQ) messages.

Each node type is defined as a PROMELA process type. In our model, there is only one originator, one destination, and one attacker node and there can be at most two intermediate nodes depending on the size of the network. The number of nodes does not change during the network lifetime.

The originator process nondeterministically checks whether it needs to send route request for the destination or not. If it has a valid route, it does nothing. If it doesn't have one, it initiates a route discovery. A route to the destination expires nondeterministically in the originator process. Route entries do not expire in the intermediate and destination nodes.

The behavior of the nodes is determined by a set of model parameters. These parameters are preprocessor directives that determines which parts of PROMELA code to be enabled. The list of our model parameters are given in Table I. These parameters are discussed in Sect. VI.

TABLE I
Model Parameters for Node Behaviors

Parameter Name	Description
RREP_HOP_COUNT_ATTACK	Attacker leaves the hop count field unchanged in RREP messages.
SPOOF_RREP	Attacker nondeterministically changes the sender field of RREP messages.
CHECK_RREP_HOP_COUNT	Honest nodes check the hop count and the sender fields in RREP messages to detect incorrect hop count.
CHECK_ROUTING_LOOP	Intermediate honest nodes applies a check to prevent routing loop formation

C. The Communication Model

In PROMELA, communication channels between processes are point-to-point. Therefore, we model broadcast communication by using arrays of unidirectional channels. In a network with N nodes, each node is associated with $N-1$ unidirectional channels, and only the channels to its neighbors are used to send and receive messages. Separation of channels allows us to use channel assertions to reduce the size of the state space. The capacity of each channel is limited to two messages, which is enough for our modeling purposes.

The neighborhood information is stored in a global connectivity matrix. Since the links are assumed to be bidirectional, the connectivity matrix is symmetric. When broadcasting, nodes refer this matrix to decide whether to send a message to a node or not. Since there is no mobility, link failure or node failure in our models, the connectivity matrix remains constant during the verification process.

D. The PROMELA Implementation and the Complexity of the Model

Our PROMELA model for SAODV has approximately 1200 lines of code (LOC) excluding comments and blank lines. The code is instrumented with preprocessor directives some of which are listed in Table III. These directives are model parameters, and they define the ultimate complexity of the model. The most important directive is the number of nodes in the network, N that determines the definitions related to channels and topology. For example, the preprocessed code with $N = 4$ has 576 LOC whereas it has 669 LOC if $N = 5$.

The complexity of a PROMELA model is usually measured by the length of its state vector and the size of its full state space. We measure the size of the full state space by letting SPIN search the state space without giving it any claim. Searching for nothing, SPIN eventually covers all the state space if time and memory permit.

Table II demonstrates how the model complexity is affected by changing the attacker's behavior and by adding an intermediate node. Enabling SPOOF_RREP, the attacker can nondeterministically impersonate any node when it forwards a RREP message, which causes an exponential increase in the model complexity.

Similarly, when we introduce an additional intermediate node, the size of the state space increases exponentially. Due to time and resource limitations, we could not obtain the number of states and transitions in the full state space for $N = 5$ with SPOOF_RREP enabled.

VI. Verification Results and Security Analysis

A. Verification Environment

Our verification environment is given in Table III.

For each property that we defined in Sect. IV, SPIN found a counter-example (i.e. violation) in milliseconds for networks with four and five nodes arranged in linear topologies. We examine these violations in the following two sections.

TABLE III
OUR VERIFICATION ENVIRONMENT

Processor	Pentium 4 Dual Core 3.2 Ghz
Memory	2 GB
Operating System	Fedora Linux Core 4
C Compiler	gcc 4.0.2
SPIN Version	4.2.9
PROMELA Preprocessor	GNU M4 1.4.3

B. An Attack on the Correctness of Distance Property

We check the correctness of distance property on a network of four nodes that are linearly arranged as shown in Fig. 2. In this scenario, the attacker increments hop count field in RREQ messages correctly but leaves it unchanged in RREP messages. It also does not spoof any other node.

Since SAODV does not have a mechanism to prevent or detect unchanged hop count values, SPIN flags this as a violation to the correctness of distance property. The shortest execution trace produced by SPIN corresponds to the scenario in Fig. 2. The violation is found at depth 436 after visiting 215 states and making 395 state transitions in 12 ms.

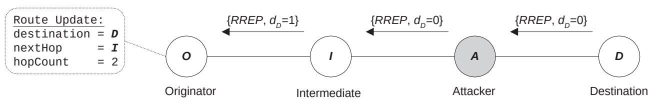


Fig. 2. Incorrect hop count propagation

1) Defending Against Hop Count Attack: For such cases where the attacker is a neighbor of the destination node, the hop count attack can be detected. Consider again the network in Fig. 2. Here, destination D sends a RREP to attacker A with hop count zero. If node A forwards RREP to intermediate node I without incrementing the hop count, node I can understand that node A is lying because the only node that can send a RREP with hop count zero is the destination itself. Detection of hop count attack is enabled by the model parameter CHECK_RREP_HOP_COUNT and it is demonstrated in Fig. 3. In our model, when node I detects the attack, it simply discards the message.

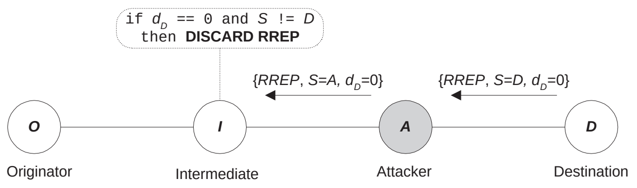


Fig. 3. Detecting incorrect hop count for RREP messages

When we incorporate such a detection into the RREP processing function in our model, SPIN reports no error. However, if we also allow the attacker to spoof other nodes, by enabling SPOOF_RREP option, SPIN finds another violation in which the attacker spoofs the destination node when forwarding the RREP message to node I. In this case, node I is misled into

TABLE II
CHANGE IN MODEL COMPLEXITY FOR DIFFERENT DIRECTIVES

Directive(s)	Depth	State Vector Size	# States	# Transitions
$N = 4$	54,224	564 bytes	767,166	1.7×10^6
$N = 4$, SPOOF_RREP	266,679	564 bytes	3.56×10^7	8.21×10^7
$N = 5$	95,461	868 bytes	4.56×10^6	1.02×10^7
$N = 5$, SPOOF_RREP	597,094	868 bytes	Unknown	Unknown

thinking that the RREP message comes from the destination itself and can not detect the hop count attack.

Note that, a similar hop count attack can be performed when forwarding RREQ messages. In this case, detection of hop count attack is possible if the attacker is a neighbor of the originator node.

C. An Attack on the Loop Freedom Property

In order to analyze the loop freedom property, we use a network with five nodes that are linearly arranged. In this analysis, the attacker is allowed to spoof the other nodes when forwarding RREP messages (i.e. SPOOF_RREP option is enabled).

SPIN finds a counter-example that is demonstrated in Fig. 4 at depth 797 after visiting 3611 states and making 9142 state transitions in 95 ms. Here, attacker A sets the sender address field of the RREP message to node I_1 and sends it to node I_2. Node I_2 looks at the sender address field of the incoming RREP message and updates the next hop address for the destination to node I_1. Then, it forwards the RREP message to node I_1 which in turn updates the next hop address for the destination to the sender of this RREP message, namely node I_2. As a result, a routing loop is formed between nodes I_1 and I_2 as shown in Fig. 4.

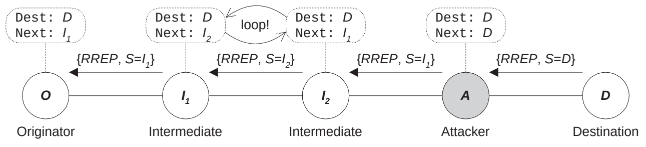


Fig. 4. Loop Formation Between Two Intermediate Nodes

1) Defending Against Loop Formation Attack: We propose a little check, which is enabled by CHECK_ROUTING_LOOP parameter, to prevent loop formation in such cases. If node I_2 compares the sender address of incoming RREP with the address of the next hop that it will forward the RREP message and sees that they are the same, it can detect the danger of loop formation and take necessary actions such as blacklisting the sender of the incoming RREP and discarding the message. In our model, when the attack is detected, the message from the attacker is discarded and no further action is taken.

When we enable CHECK_ROUTING_LOOP directive, SPIN finds the violation shown in Fig. 5. Here, the attacker spoofs the originator O when forwarding the RREP to I_2, causing a routing loop that consists of nodes O, I_1, and I_2. However, in this case, the loop is not physical but logical because node O and I_2 are not neighbors and I_2 can recover from this case soon by deploying one of the local connectivity maintenance techniques suggested in AODV RFC [6].

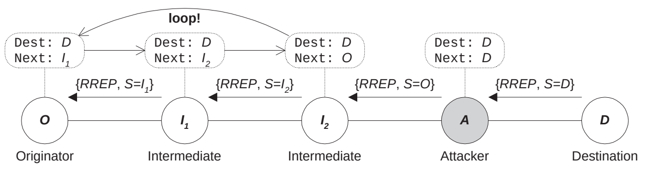


Fig. 5. Routing Loop Formation Among Three Nodes

VII. RELATED WORK

There are very few works in the literature that deploy formal methods to analyze security of secure routing protocols for MANETs.

In [7], Ács *et al.* developed a formal framework based on simulation paradigm to analyze the security of distance vector routing protocols for MANETs such as SAODV [4], [5] and ARAN [8]. The authors defined two attacks for SAODV. One of the attacks is already defined in SAODV paper, and it involves forwarding a RREQ message without changing its hop count value, deceiving the destination to think the route is one hop shorter, which makes this route more attractive. In the second attack, the attacker forwards RREP that comes from the destination to the source as if it comes from another node (i.e. spoofing attack) that is not a true next hop for the destination. In this case, the data packets from the source to the destination are probably not get delivered or routing loops can be formed that causes energy consumption. SAODV draft already tells that it does not do much about such DoS attacks [4].

In a recent work, Nanz and Hankin applied their calculus and static analysis technique to model and analyze a basic version of SAODV [9]. They demonstrate their technique to show the same spoofing attack as in [10].

There are two recent works that are the most relevant to ours. One is [11] in which Önem uses PROMELA and SPIN to make a formal security analysis of ARIADNE [12], a secure on-demand source routing protocol for MANETs. The other is [13] in which Andel proposes a formal security analysis framework using the SPIN model checker. Andel also incorporates an adaptive threat model into his framework.

There are also several studies involving the modeling and verification of some routing properties of MANET routing protocols. In one of these studies, Bhargavan *et al.* [14] model AODV with PROMELA and analyze it with SPIN to demonstrate that the protocol is not loop-free. Their network model is quite simple. It consists of three network nodes with no mobility and predetermined roles such as destination. The model also includes node failures, link breakages, and node restarts to show the loop conditions. In this study, the authors suggest an improvement to the AODV protocol and provide a formal proof showing that the improved version is loop-free. They use SPIN and HOL for this purpose.

In another study, Wibling *et al.* [15] analyze functional correctness of a hybrid MANET routing protocol called LUNAR with SPIN. There are five to seven nodes in their network models and their verification scenarios involves couple of topology changes.

Renesse and Aghvami [16] used SPIN to verify certain properties of WARP, which is a hybrid MANET routing protocol. Their verification model includes five nodes with predetermined roles for sender and receiver. The model also includes a limited degree of mobility.

VIII. Conclusion and Future Work

We have shown by using model checking the two important design deficiencies in SAODV which allows an external attacker to manipulate routing information in the network. Although these deficiencies are already known, we think it is important to show them by using a well-established formal method such as model checking. We have also proposed some improvements that help honest nodes detect the two attacks in some specific cases.

Our study has two important results. First, it shows a serious consequence, namely routing loops, as a result of not protecting the sender address field, which is used to establish routing information. Although, DoS attacks are out of scope for SAODV, we think such a trivial manipulation should not be allowed by any secure MANET routing protocol.

The second result is the promising usage of a well-known and powerful generic model checker to reveal security attacks in the presence of an attacker. Despite our assumptions and simplifications, our SAODV model is nevertheless complicated as a finite system to be model-checked. However, SPIN can manage to reveal the attacks in a couple of seconds or even in a sub-second time scale. This can be considered as the best case since the attacks are quite trivial to find, but there may be more complicated security flaws that are hard to find without automation and model checkers like SPIN seem like the most suitable tools to search for such flaws.

This work presents the initial results of our research on using model checking to analyze the security of routing protocols for MANETs. We are currently focused on the analysis of security properties involving mobility and multiple attackers whose presence makes secure routing a challenge. We are planning to incorporate detection mechanisms such as neighborhood watching into our models. We are also considering to model-check other secure routing protocols against the same properties using the same attacker types in order to make a formal comparison.

References

[1] Y.-C. Hu and A. Perrig, "A survey of secure wireless ad hoc routing," *IEEE Security and Privacy*, 2004.

[2] P. G. Argyroudis and D. O'Mahony, "Secure routing for mobile ad hoc networks," *IEEE Communications Surveys*, vol. 7, no. 3, pp. 2–21, 2005.

[3] G. J. Holzmann, *Spin Model Checker, The: Primer and Reference Manual*. Addison Wesley, September 2003.

[4] M. G. Zapata, "Secure ad hoc on-demand distance vector (SAODV) routing," Internet Draft, draft-guerrero-manet-saodv-06, work in progress, September 2006.

[5] M. G. Zapata and N. Asokan, "Securing ad hoc routing protocols," in *the ACM Workshop on Wireless Security (WiSe'02)*, 2002.

[6] C. E. Perkins, E. M. Belding-Royer, and S. R. Das, "Ad hoc on-demand distance vector (AODV) routing," IETF RFC 3561, July 2003.

[7] G. Ács, L. Buttyán, and I. Vajda, "Provable security of on-demand distance vector routing in wireless ad hoc networks," in *Proceedings of the Second European Workshop on Security and Privacy in Ad Hoc and Sensor Networks (ESAS 2005)*, ser. LNCS 3813. Springer-Verlag, July 2005, pp. 113–127.

[8] K. Sanzgiri, B. Dahill, B. N. Levine, C. Shields, and E. M. Belding-Royer, "A secure routing protocol for ad hoc networks," in *Proceedings of the 10th IEEE International Conference on Network Protocols (ICNP'02)*, Paris, France, November 2002.

[9] S. Nanz and C. Hankin, "A framework for security analysis of mobile wireless networks," *Theoretical Computer Science*, vol. 367, no. 1-2, pp. 203–227, 2006.

[10] G. Ács, L. Buttyán, and I. Vajda, "Provably secure on-demand source routing in mobile ad hoc networks," *IEEE Transactions on Mobile Computing*, vol. 5, no. 11, November 2006.

[11] E. Önem, "Formal security analysis of a secure on-demand routing protocol for ad hoc networks using model checking," MSc Thesis, Bogazici University, 2007.

[12] Y.-C. Hu, D. B. Johnson, and A. Perrig, "Ariadne: A secure on-demand routing protocol for ad hoc networks," in *MOBICOM'02*, September 2002.

[13] T. R. Andel, "Formal security evaluation of ad hoc routing protocols," PHD Thesis, The Florida State University, 2007.

[14] K. Bhargavan, D. Obradovic, and C. A. Gunter, "Formal verification of standards for distance vector routing protocols," *Journal of the ACM (JACM)*, vol. 49, no. 4, pp. 538–576, 2002.

[15] O. Wibling, J. Parrow, and A. Pears, "Automatized verification of ad hoc routing protocols," in *Proceedings of 24th IFIP WG 6.1 International Conference on Formal Techniques for Networked and Distributed Systems (FORTE 2004)*, ser. LNCS 3235. Madrid, Spain: Springer-Verlag, September 2004.

[16] R. de Renesse and A. H. Aghvami, "Formal verification of ad-hoc routing protocols using SPIN model checker," in *Proceedings of The 12th IEEE Mediterranean Electrotechnical Conference(MELECON 2004)*, Dubrovnik, Croatia, May 2004, pp. 1177–1182.