

# SECURITY ANALYSIS OF THE BOOTSTRAP PROTOCOL FOR DENY-BY-DEFAULT MOBILE AD-HOC NETWORKS

Honggang Zhang  
Suffolk University  
hzhang@suffolk.edu

O. Patrick Kreidl and Brian DeCleene,  
BAE Systems  
{pat.kreidl, brian.decleene}@baesystems.com

Jim Kurose,  
UMass Amherst  
kurose@cs.umass.edu

Xiaoyu Ni  
Suffolk University  
xni@suffolk.edu

## ABSTRACT

*In previous work, we proposed a “Bootstrap” protocol for establishing neighbor relationships, between two mobile nodes in a mission critical deny-by-default Mobile Ad-hoc Network. In this paper, we formally characterize the security properties of this Bootstrap protocol, striving to answer the following questions: 1) To what extent can an adversary undermine the correctness and performance of the Bootstrap protocol? 2) To what extent can the Bootstrap protocol be improved in anticipation of an adversary? Our analyses employ a combination of formal logic and two standard automated model checkers, SPIN and PRISM. Two types of threats are considered, which we call the subverted node and the subverted link. In the subverted link analysis, we further categorize the adversary into two variants, which we call dark-red or light-red in correspondence with having detailed Bootstrap-protocol-specific knowledge or only generic neighbor setup knowledge, respectively. The subverted node analysis shows that the adversary cannot TCP-SYN-flood-like attack nor deadlock the good node within the Bootstrap protocol. The subverted link analysis shows that the adversary cannot undermine the correctness of the protocol, in the sense that the protocol’s performance is only degraded in a bounded manner by the dark-red adversary or in a benign manner by the light-red adversary.*

## I. INTRODUCTION

Deny-by-default computer systems [3][6][13] promise enhanced information security (in comparison to their allow-by-default counterparts) by relying on policy rules to explicitly define the various actions that system components are allowed to take. For example, policy in a deny-by-default network [14][2][1][16] would specify which nodes can forward data or control traffic to which other nodes—datagrams for which no policy is provided are dropped. In contrast, policy in today’s Internet [12] typically specifies only what traffic should be blocked (via firewalls or ingress filters at routers)—datagrams outside of policy are forwarded. The promise of

enhanced security at the expense of open functionality is particularly appropriate in the context of mission-critical applications with the potential of an adversarial presence e.g., military Mobile Ad-hoc NETWORKS (MANETs).

A crucial challenge in a deny-by-default mission-critical MANET, in which network topology and information exchange requirements can be time-varying, is the manner in which the nodes maintain up-to-date policy [19]. Our previous work [20] addressed this challenge by defining (i) an *axiomatic* set of policies from which nodes can obtain additional policies or update outdated policies and (ii) a protocol (referred to as the Bootstrap protocol) by which nodes form, or *bootstrap*, the neighbor relationships among themselves in a manner consistent with current policy<sup>1</sup>. Along with the definition of the Bootstrap protocol as a Finite State Machine (FSM) [5][20], we also proved its correctness (safety and liveness); however, this initial analysis neglected the possibility of nodes or wireless channel being subverted by a knowledgeable adversary.

This paper extends the work in [20], seeking to answer the following questions: 1) To what extent can an adversary undermine the correctness and the performance of the Bootstrap protocol? 2) To what extent can the Bootstrap protocol be improved in anticipation of an adversary? To this end, we augment the Bootstrap protocol to leverage additional security features (e.g., strong identity) and formally characterize its security properties when under attack by an adversary. Our analyses employ a combination of formal logic and two standard automated model checkers, SPIN [9] and PRISM [22]. Two types of threats are considered, both illustrated in Figure 1 assuming two MANET nodes (labeled A and B) meet and begin executing the Bootstrap protocol. The threat of a subverted node assumes one of the two nodes is controlled by the adversary, whereas the threat of a subverted link assumes the channel between the two nodes is controlled by the adversary (similar to a “man-in-the-middle” attack). In the subverted link threat, we categorize the adversary into two variants, which we call dark-red or light-red in correspondence with having detailed Bootstrap-protocol-specific knowledge or only generic neighbor link setup knowledge, respectively.

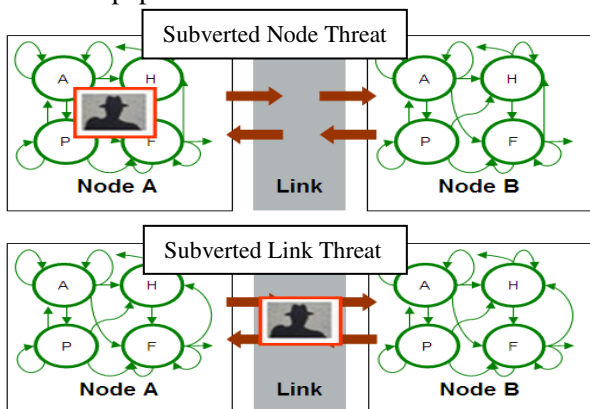
This material is based upon work under a subcontract #069153 issued by BAE Systems National Security Solutions, Inc. and supported by the Defense Advanced Research Projects Agency (DARPA) and the Space and Naval Warfare System Center (SPAWARSYSCEN), San Diego under Contract No. N66001-08-C-2013.

This research was supported in part by a research assistance grant through the College of Arts and Sciences at Suffolk University.

<sup>1</sup>The recognition here is that routing and other higher-level network functions can be put in place only once neighbor relationships are established.

The main contributions of our paper are as follows. We show that the subverted node cannot TCP-SYN-flood-like attack or deadlock the good node that is running the Bootstrap protocol with the subverted node. We show that the subverted link cannot undermine the correctness of the Bootstrap protocol. The subverted link’s impact on the protocol’s performance is shown to depend upon the adversary’s knowledge of the protocol specifics: in particular, even against the dark-red adversary, the performance degradation of the protocol is bounded; against the light-red adversary, the performance degradation is negligible.

The rest of this paper is structured as follows. Section II presents the preliminary for our analysis. In Section III, we discuss the threat from a subverted node. In Section IV, we discuss our analysis of the dark-red and light-red variants of the subverted link. Finally we conclude the paper in Section V.



**Figure 1.** Two considered threats to the Bootstrap protocol. Each node is represented as a FSM that has four internal states: Active Beacon (A); Passive (P); Half-Open (H); Full-Open (F). State transitions within a node are triggered by message exchange (with the other node) and policy evaluation [20].

## II. PRELIMINARY

We give a brief review of the Bootstrap protocol [20], and then enhance it with a number of security features.

### A. The Bootstrap Protocol

The Bootstrap protocol is designed for a scenario where two nodes meet in a deny-by-default MANET and both nodes can hear the other’s transmissions. The protocol allows the two nodes to establish a neighbor relationship when that relationship, and the process for bootstrapping that relationship, is consistent with policy. Recall that in a deny-by-default system, any action should be explicitly allowed by policy. By two nodes “establishing a neighbor relationship”, we mean that the final state of both nodes is that they have each agreed that they can send/receive messages to/from each other.

The Bootstrap protocol relies on a set of *axiomatic* policies, which is the initial set of primitive policies that are installed *a priori* and allow a deny-by-default system

to establish neighbor relationships. This set consists of four policies among which, *MayBeacon* policy specifies whether a node is allowed to advertise its existence, and *MayShare* policy of a node with respect to another node specifies whether this node may share a policy with another node. Since axiomatic policies do not define whether a particular neighbor relationship is allowed, for performance reason a node will typically also be initialized with non-axiomatic *OkNeighbor* policy that defines a set of nodes with which this node can be neighbor. *OkNeighbor* policy can be obtained by running the Bootstrap protocol.

When two nodes run the protocol with each other, each node’s internal state transitions can be described by a Finite State Machine (FSM) [20]. As shown in Figure 1, each node has four internal states: *active beacon* (A), *passive* (P), *half-open* (H), and *full-open* (F). Note that for this protocol, all state transitions should be allowed by policy. For example, if both nodes’ *OkNeighbor* policies (with respect to each other) have the value TRUE and at least one node is initially in *active beacon* state (i.e., *MayBeacon* policy has the value TRUE), then the two nodes can follow the transitions specified in the FSM to establish their neighbor relationship (i.e., both are internally in *full-open* state). If a node’s *OkNeighbor* policy (with respect to the other node) is undefined, it still can acquire this policy via message exchanges (if allowed by the two nodes’ axiomatic policies *MayShare* with respect to each other) with the other node, even though they have yet to establish a neighbor relationship. We have formally verified (in [20]) the correctness (safety and liveness) of the protocol by SPIN [9], a powerful model checker for formal verification of distributed software systems.

### B. Security Features of the Bootstrap Protocol

We propose to enhance the protocol with the following features. *First*, all messages (exchanged between two nodes) are privacy- and integrity- protected via encryption and cryptographic hash. Strong identity (e.g., provided by signature) is in place, so no node can impersonate any other node. *Second*, each message has a timestamp, which serves as a sequence number for defending against message reordering/replaying attack. *Third*, a node never simultaneously keeps two or more bootstrap sessions with any other node. To rate-limit requests, a node ignores repeated requests within a certain time interval from the first such request.

## III. SUBVERTED NODE THREAT

In this threat model, we suppose that exactly one of the two nodes engaged in the Bootstrap protocol is subverted, meaning that the adversary has full control over the behavior of that node. Clearly, the adversary may always choose to neglect taking its steps in the

protocol and, in turn, prevent the neighbor relationship from being established. This is certainly a desirable outcome from the security perspective, so we assume instead that the adversary’s intent is to establish the neighbor relationship. Note that, by our up-to-date policy assumption and the verified correctness of the protocol, we know this neighbor relationship will only be established if permitted by policy<sup>2</sup>. And, given the adversary has yet to be detected and operates within the protocol, the good node has no pretext for denying the neighbor relationship. Altogether, then the real interesting question to ask is the extent to which the adversary may engage with the protocol to confound the operation of the good node. The following claim verifies that the adversary is highly limited along these lines.

**Claim:** *The subverted node cannot TCP-syn-flood-like attack or deadlock the good node within the Bootstrap protocol.*

The logic to support this claim is as follows. **First**, by strong identity, the good node will never store more than one state relative to the subverted node. Recall that in TCP-SYN-flood attack [21], an honest TCP server cannot defend against a SYN-flood attack from a malicious TCP client (unless SYN cookie is used), because the server does not verify the identity of each client and it is possible for a malicious client to trick the server to store a large number of states by repeatedly sending a large number of TCP connection requests in a short time interval. But due to the strong identity assumption in our case, the identity of any node can be verified, so a subverted node cannot trick a good node to store multiple states for itself. **Second**, the Bootstrap protocol specifies that a node ignores repeated requests within a certain time interval from the first such request. This rate-limit feature essentially slows down repeated requests. **Third**, the resource overhead to process and ignore messages is negligible. **Finally**, it is impossible for a node to deadlock the other node that is running the Bootstrap protocol with itself. If deadlock were possible, our previous SPIN-based safety verification in [20] would have failed.

Thus, in order to prevent a good node from establishing a neighbor relationship with a subverted node, we need some countermeasures such as detecting the malicious node and updating policy, but they are beyond scope of the Bootstrap protocol.

#### IV. SUBVERTED LINK THREAT

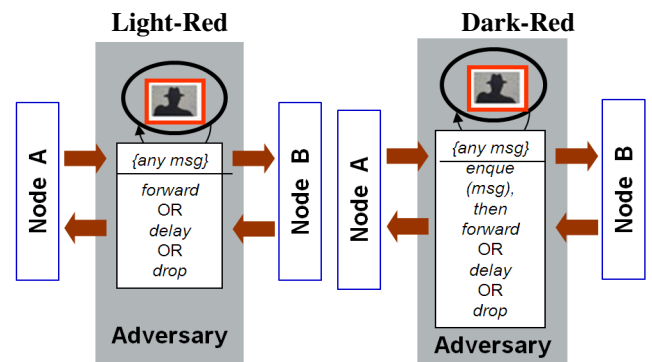
We now weaken the adversary, in the sense that both nodes in the Bootstrap protocol are non-subverted and instead the adversary is tapped into the channel between

<sup>2</sup> Or even if not permitted by policy but the subverted node also has *PolicySource* privileges, which is a security problem of much broader scope than that of the Bootstrap protocol. This is beyond the scope of this paper.

them. Neither node is aware of the existence of the adversary, each perceiving the channel like any other unreliable channel. Any message from one node may be processed (i.e., forwarded, delayed or dropped) by the adversary before it may be received by the other. Such a scenario is possible if the adversary uses a directional antenna [27]. Our assumed strong identity services prevent the adversary from altering/spoofing messages.

Our analysis proceeds differently depending on different assumptions about the adversary’s knowledge of the protocol. We categorize the adversary into the following two variants. **Dark-red adversary**, having a complete knowledge of the Bootstrap protocol (e.g., transition triggers, key timers), can always deduce from the message sequence the internal states of the two nodes and strategically choose its sequence of actions. It follows that the dark-red adversary is well modeled as an unreliable link with worst-case delay or loss events, which is amenable to analysis via SPIN [9] upon augmenting the FSM description of the protocol to include the adversary’s action space by allowing adversary to non-deterministically process messages. **Light-red adversary**, having no specific knowledge of the protocol, simply acts to confound a generic neighbor setup process. The light-red adversary is well modeled as an unreliable link with random message drops (since a random delay amounts to a dropped message if the delay exceeds a timer or a forwarded message otherwise), which is amenable to analysis via PRISM [22] upon augmenting the FSM description of the protocol with probabilistic message loss.

The augmented FSM descriptions of the system (including two good nodes and one adversary) are shown in Figure 2 (where internal states of nodes A and B are not shown). The bootstrapping process between A and B can be described by a sequence of *system states*, each



**Figure 2.** FSMs of ligh-red and dark-red adversary. Any message from one node is processed by the adversary before it is received by the other. The adversary-controlled channel is modeled as a process with a single transition (upon receiving a message) which probabilistically processes messages if the adversary is light-red or non-deterministically processes messages (including worst-case action sequences against the protocol execution between the two nodes) if the adversary is dark-red.

being a pair of the internal states at the two nodes, i.e. (*A's state*, *B's state*). Since there are four internal states per node in the protocol, there are sixteen system states in total. Initially the system can only be in one of the following four states, (*active beacon*, *active beacon*), (*active beacon*, *passive*), (*passive*, *active beacon*), or (*passive*, *passive*), and all others may be entered only after the protocol is initiated. The initial condition of the system can be specified by the values of the following three policies of the two nodes: axiomatic policies *MayBeacon* and *MayShare* (each has value either TRUE or FALSE) and non-axiomatic policy *OKNeighbor* (with value either UNDEFINED, TRUE, or FALSE). Note that a particular initial system condition defines a particular initial system state. Our security analyses of the protocol (based on SPIN and PRISM) are repeated for all possible initial conditions.

#### A. DARK-RED ADVERSARY

The goal of the dark-red adversary is to lead the bootstrapping process between the two good nodes into an undesirable system state, through manipulating the message exchanges between the two nodes. A system state is *undesirable* if it is either *inconsistent* or normally *unreachable*. Formally, **inconsistent state** is a state where one node believes the neighbor relationship is established (i.e., this node is internally in the *full-open* state) while the other node does not believe so. **Unreachable state** is a state that, holding all initial conditions the same and removing the adversary from the picture, cannot be reached by the protocol.

For example, states (*full-open*, *half-open*) and (*full-open*, *active beacon*) are each inconsistent state. If the system starts with (*passive*, *passive*) (i.e., both nodes' *MayBeacon* policies have the value FALSE), then (*active beacon*, *passive*) and (*full-open*, *half-open*) are each normally unreachable; of course, (*full-open*, *half-open*) system state is *not* normally unreachable if both nodes' *MayBeacon* have the value TRUE and both nodes' pre-installed *OkNeighbor* policies have the value TRUE. An inconsistent state can be reachable from some initial conditions and unreachable from others. Whenever a state is both inconsistent and unreachable, we classify it as unreachable. Inconsistent states can occur normally, but they are transient in the absence of an adversary.

##### 1) SPIN-Based Analysis

We use SPIN to identify possible attacking strategies by the adversary. We first build a SPIN model for the bootstrapping process between the two good nodes via an adversary-controlled wireless channel. A SPIN model represents a number of interacting state machines (or processes) communicating via "message channels" (a built-in data type). As shown in the right sub-figure of

Figure 2, the adversary has four message channels (two incoming and two outgoing) to communicate with the two nodes. There is no direct message channel between the two nodes. Each node's internal state transitions are defined in [20].

The adversary process is specified as an infinite loop waiting for messages from two incoming channels. The adversary first buffers a received message into a queue, then may non-deterministically (i.e., every option is always possible) choose one of three actions: immediately forward, drop or delay the message. Due to signed sequence numbers, message re-ordering by the adversary will not cause any security problem.

Since SPIN is built upon discrete-event assumptions and has no mechanism for representing absolute time, thus we model the adversary's action of delaying the forwarding of an actual message (until the message-receiving node times out) as a *timeout\_message*, triggering the same transition on the receiving node as if its timer expires before the actual message is received.

The non-deterministic infinite loop of the adversary process effectively tells SPIN that the adversary is free to choose from all possible action sequences against any particular valid execution of message exchanges between the two good nodes. Note that an attacking sequence or strategy by the adversary is just a particular action sequence. Thus, defining the adversary process in this way allows us to use SPIN's verifier (performing exhaustive search) to perform worst-case protocol analysis which reflects dark-red adversary assumptions. Specifically, it allows us to query SPIN for what sequence of adversary actions can drive the system into any particular undesirable state. That is, if we input into SPIN a claim that a particular undesirable state cannot be reached from a given initial condition, SPIN will output either that the claim is TRUE or provide a counter-example, in the latter case SPIN also outputs the adversary's sequence of actions that bring about the particular undesirable state.

Our analysis proceeds as follows. First, for each initial system condition, we classify all 16 system states into three categories: *normal or OK state*, *unreachable state*, and *inconsistent state*. Then, we use SPIN to check if the adversary process can choose an action sequence to lead the system into any undesirable state. To do that, we input a claim in SPIN for each undesirable state, and run SPIN verifier to check if that claim is TRUE or not. The following is an example of our analysis procedure.

#### Example Analysis.

- *Step 1: Define initial conditions.* Let both nodes' *MayBeacon* and *MayShare* (with respect to each other) policies have the value TRUE, which implies that the initial system state is (*active beacon*, *active beacon*), and we let both nodes' *OKNeighbor* (with respect to each other) have

the value UNDEFINED. Suppose that each node initially carries the other node’s *OKNeighbor* policy.

- *Step 2: Classify system states.* We classify all 16 possible system states based on the initial condition (in Step 1) as follows: 7 unreachable states; 4 inconsistent state; 5 ok or normal states. See our technical report [26] for details.
- *Step 3: Present claims to SPIN.* For each undesirable state (either inconsistent or unreachable), we input a claim into our SPIN model, which claims that the undesirable state can not be reached if the system starts with the initial condition given in Step 1. We query SPIN to check those claims. Figure 3 presents the counter-example output by SPIN, in which we’ve highlighted (by bold curves) the implied adversary action sequence, against the claim “inconsistent system state (*full-open, active beacon*) cannot be reached.”

Let us elaborate on SPIN output in Step 3. In our discussion, we assume that readers are familiar with the Bootstrap protocol [20]. In this counter-example (shown in Figure 3), we see that the adversary immediately forwards all messages until the system enters (*full-open, full-open*) state. Then, the adversary continues to forward the minimum required number of upkeep messages from Node B to Node A, storing others in a queue or buffer. That is, the adversary makes sure an upkeep messages arrives at A just before A is about to timeout (in order to keep A in that state as long as possible). In the meantime, the adversary delays the *NeighborAck* message from A to B until B times out. At this point, the system is in inconsistent (*full-open, active beacon*) state. In order to sustain this state, the adversary drops all beacon messages from B and pulls from its previously stored upkeep messages to keep A in its *full-open* state. However, by virtue of the signed sequence numbers, the supply of upkeep messages is eventually exhausted at which point A times out and the system returns to the initial state. Thus, the time interval during which the adversary can keep the system in the undesirable state is bounded.

Note that the inconsistent system state (*full-open, active beacon*) can actually be reached even without adversary, but it is only a transient state. In the absence of adversary, if for some reason the *NeighborAck* message from A to B is lost (or delayed long enough), then B will timeout and return back to *active beacon* state. Then, both nodes will re-start running the protocol with each other. But, the adversary can intentionally lead the system into this inconsistent state by following the attacking sequence identified in Figure 3.

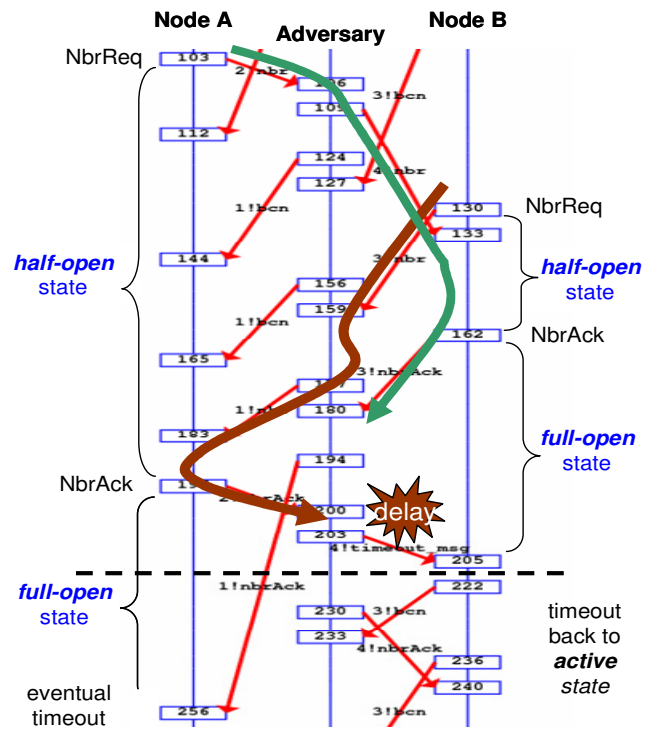
To further understand the attacking strategy of the adversary, we give in Figure 4 a FSM internal to the adversary, and the state-transition path (shown as bold dashed curve) followed by the adversary in the counter-example (shown in Figure 3). Note that each state in the FSM in Figure 4 represents the adversary’s knowledge of the system state, which is true system state, since the

adversary has full knowledge of the protocol and it handles all message exchanges. This FSM specifies multiple paths or action sequences by the adversary to lead the system into the intended inconsistent state.

## 2) Summary of analysis results

In our analysis, both nodes’ *MayShare* (with respect to the other node) can be different, and we assume policies are consistent in the sense that both nodes’ *OkNeighbor* policies with respect to each other are symmetric. We assume if one node’s *OkNeighbor* policy with respect to the other node is undefined, then the other node always carries an up-to-date copy of that policy. Based on the above assumptions, we have indentified 39 initial system conditions (see our technical report [26] for details) after merging redundant states and considering all possible values of *MayShare*, *MayBeacon*, and *OkNeighbor*.

We have repeated SPIN-based analysis for all possible initial conditions, and have demonstrated the following security properties of the Bootstrap protocol: 1) *No sequence of actions by the adversary can cause the system to enter a normally unreachable system state;* 2) *Adversary can lead system into inconsistent states, but*



**Figure 3.** SPIN’s Counter-Example for the Claim in our Example Analysis. This diagram shows the adversary’s action sequence to lead the system into inconsistent state (*full-open, active beacon*). The three vertical lines (from left to right) represent respectively node A, adversary, and node B. The boxes represent action or transition events, each with an id. Short straight lines with arrows represent message transmissions. Initially, the adversary relays messages immediately, till the system enters (*full-open, full-open*). In event 203, the adversary delays the forwarding of *NeighborAck* until B’s timer (for *full-open* state) expires. This delay till timeout is shown as the adversary sending a *timeout\_msg*. Once B times out, the system enters (*full-open, active beacon*) state.

not for arbitrarily long time.

**Choosing timer value.** We learn from our analysis that there is an important security and performance tradeoff when choosing values for the two timers (one for *half-open* and one for *full-open*) in the protocol. For the attacking sequence identified in the example analysis, a smaller timer value for *full-open* state implies good security, i.e., a smaller interval of inconsistent state. But a smaller timer value implies that a node more likely times out prematurely, then the performance suffers. See our technical report [26] for more discussions.

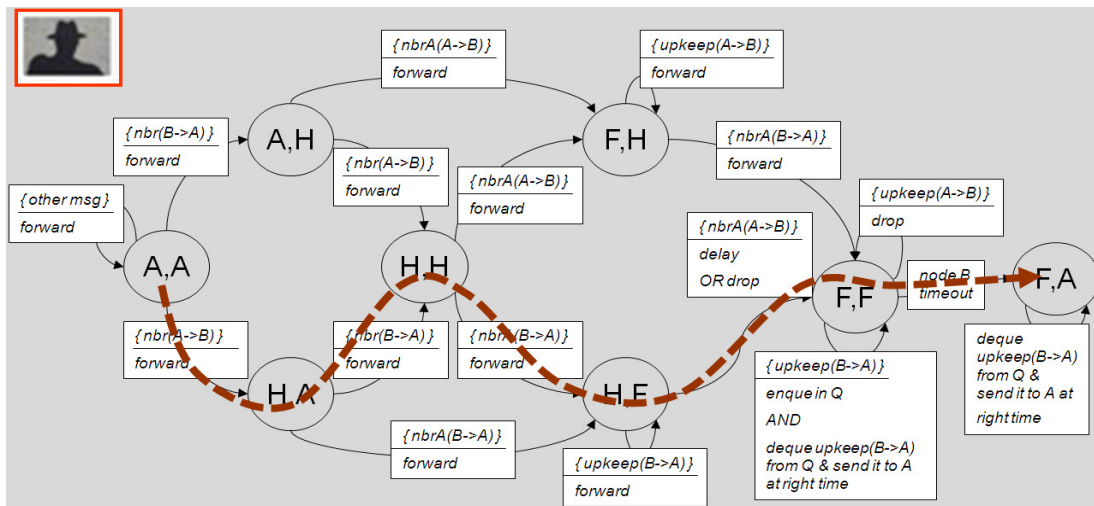
### B. Light-red Adversary

The goal of the light-red adversary is to prevent the neighbor relationship (of two good nodes) from being established successfully. Note that this goal can be achieved with probability one by the dark-red adversary, since it has complete knowledge of the protocol and can track the system state throughout. However, the light-red adversary lacks both the knowledge to track the system state and the full control over the link to act against the protocol precisely. Nevertheless the light-red adversary can *randomly* drop messages to disrupt the operation of the protocol. Thus, the light-red adversary's impact on the protocol is measured in terms of the extent to which a lossy channel increases the *expected* time needed for successfully establishing the neighbor relationship.

Starting from some initial condition, the system (of two good nodes) experiences a sequence of transitions which may end in (*full-open, full-open*) state or may go back to the initial state (due to timeout). We refer to the former sequence as a *successful trial*, and the latter one as an *unsuccessful trial*. If no message loss or delay, a protocol run consists of only one successful trial; otherwise, it may take more than one trial for the two

nodes to establish their neighbor relationship. Thus, the performance of the protocol under the attack by light-red adversary can be captured by a trial's success probability  $q$  as a function of the channel's message loss probability (which captures the adversary's capability). Or the performance can be captured by the number of trials needed (denoted as  $X$ ) to successfully establish the neighbor relationship. Clearly  $X$  is a geometric random variable with mean  $EX=1/q$ . A smaller  $EX$  indicates that the protocol is more robust to the lossy channel or to the attack from the adversary.

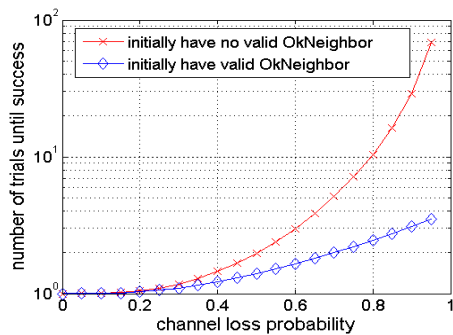
**PRISM-based Analysis.** A trial's success probability can be found by relying on PRISM [22], a widely used probabilistic model checker for analyzing protocols [24][25][23]. For a certain message loss probability, PRISM can perform a variety of probabilistic computations, including the probabilities of protocol execution paths starting from any initial system condition. We cast the two node processes interacting with the lossy channel process as an instance of a discrete-time Markov model in PRISM. Then for a given channel loss probability, we query PRISM to compute the probability of any successful or unsuccessful trial starting from any initial system condition. We now give an example result, shown in Figure 5, for two different assumptions on the nodes' initial *OkNeighbor* policies (all other things equal). The horizontal axis varies the channel loss probability, while the vertical axis is the expected number of trials till the full neighbor relationship is established. For message loss probabilities below 0.2, every trial is successful with overwhelming probability (or the expected number of trials to success is one). Note that the case of high message loss probability ( $>0.8$ ) is not a security concern,



**Figure 4.** FSM specification of the adversary's attacking strategy for leading the system into inconsistent state (*full-open, active beacon*). Each state in this FSM represents the system state (known to the adversary). The dashed curve corresponds to the sequence of attacking actions shown in Figure 3. Note that the adversary keeps a buffer (Q) of received upkeep messages, and sends each message out at the time that guarantees it arrives right before the receiving node's timer is about to expire.

because it makes more sense not to establish a neighbor relationship at all if the channel loss rate is that high. The adversary's impact (for a given channel loss probability) is greater when the nodes do not initially possess their *OkNeighbor* policies, reflecting the additional messages required to first exchange these policies before continuing with the rest of the protocol.

In sum, for all initial system conditions, our PRISM-based analysis has shown that the protocol is very robust under the attack from a light-red adversary. The average number of trials needed till successfully establishing a neighbor relationship is small if the channel loss rate is low. The light-red adversary's attack only degrades the performance of the protocol in a benign manner.



**Figure 5.** Bootstrap Performance Degradation Against the Light-Red Adversary. Top curve represents the case where initially neither node carries its own valid *OkNeighbor* policy but both nodes carry the other node's *OkNeighbor* policy.

## V. CONCLUSION AND FUTURE WORK

We have performed formal security analysis of the Bootstrap protocol by employing a suite of formal methods including deductive logic based on explicit assumptions and adversary's intention, SPIN model checker for verifying correctness or finding counter examples to identify possible attacking strategies by the dark-red adversary, and PRISM model checker for characterizing performance degradation due to light-red adversary. We have demonstrated that a subverted node cannot TCP-SYN-flood-like attack or deadlock a good node, and a subverted link cannot undermine the correctness of the protocol in the sense that the dark-red adversary only degrades the protocol's performance in a bounded manner and the light-red adversary degrades the protocol's performance in a benign manner.

Our analysis has exposed the values of some functionality that are beyond the scope of bootstrap. For example, it is important to have up-to-date policy, strong identity, hop-by-hop encryption, and sequence numbers. In order to prevent the link setup with subverted node we would require some additional mechanisms for detecting and policy updating. As future work, we will perform a more comprehensive analysis (covering more variants of system policy configurations) of the protocol. We will

try to apply SPIN/ PRISM-based analysis techniques to other protocols for deny-by-default MANETs.

## REFERENCES

- [1] K. Argyraki and D. Cheriton, "Network capabilities: The good, the bad and the ugly," in *HotNets-IV*, November 2005.
- [2] H. Ballani, Y. Chawathe, S. Ratnasamy, T. Roscoe, and S. Shenker. "Off by default!" in *HotNets-IV*, November 2005.
- [3] M. Bauer, "Paranoid penguin: Introduction to selinux, part ii", *Linux Journal*, vol. 155, 2007.
- [4] S. Bhatt, S. R. Rajagopalan, and P. Rao, "Federated security management for dynamic coalitions," in *DARPA Information Survivability Conference and Exposition*, 2003.
- [5] G. Bochmann and C. Sunshine, "Formal methods in communication protocol design," *IEEE Tran. on Comm.*, V28, no. 4, pp. 624-631, 1980.
- [6] S. Bratus, A. Ferguson, D. McIlroy, and S. Smith, "Pastures: Towards usable security policy engineering," in *2nd International Conference on Availability, Reliability and Security*, 2007.
- [7] C.-Y. J. Chiang, et al., "Performance analysis of drama: a distributed policy-based system for manet management," in *IEEE MILCOM*, 2006.
- [8] J. Clark, J. Murdoch, J. McDermid, S. Sen, H. Chivers, O. Worthington, and P. Rohatgi, "Threat modelling for mobile ad hoc and sensor networks," in *Annual Conference of ITA*, 2007.
- [9] G. J. Holzmann, *The Spin Model Checker: Primer and Reference Manual*. Addison-Wesley, September 2003.
- [10] S. Ioannidis, A. D. Keromytis, S. M. Bellovin, and J. M. Smith. "Implementing a distributed firewall," in *ACM CCS 2000*.
- [11] S. Kent and K. Seo, *IETF RFC 4301: Security architecture for the internet protocol*. 2005.
- [12] A. Keromytis, S. Ioannidis, M. Greenwald, and J. Smith, "The strongman architecture," in *3rd DARPA Information Survivability Conference and Exposition*, 2003.
- [13] H. Peine, "Rules of thumb for developing secure software: Analyzing and consolidating two proposed sets of rules," in *3rd Int. Conf. on Availability, Reliability and Security*, 2008.
- [14] T. Wolf, "Design of a network architecture with inherent data path security," in *3rd ACM/IEEE Symposium on Architecture for networking and communications systems*, 2007.
- [15] H. Wong, C.-K. Chau, J. Crowcroft, and K.-W. Lee, "How to enable policy-based interactions in dynamic wireless networks?" in *IEEE Workshop on Policies for Distributed Systems and Networks*, 2008.
- [16] X. Yang, D. Wetherall, and T. Anderson, "A DOS-limiting network architecture," in *ACM SIGCOMM*, 2005.
- [17] H. Zhang, B. DeCleene, J. Kurose, and D. Towsely, "Vulnerability analysis of policy bootstrapping," in *Tec. Report*, UMass. Amherst, 2008, [ftp://gaia.cs.umass.edu/pub/Zhang08\\_vulnerability\\_analysis.pdf](ftp://gaia.cs.umass.edu/pub/Zhang08_vulnerability_analysis.pdf).
- [18] R.B. Bobba, L. Eschenauer, V. Gligor, and W. Arbaugh, "Bootstrapping security associations for routing in mobile ad-hoc networks", in *IEEE Globecom*, 2003.
- [19] M. Srivatsa, D. Agrawal and S. Balfe, "Bootstrapping Coalition MANETs" in *ITA Technical Report*, February 2008.
- [20] H. Zhang, B. DeCleene, J. Kurose, and D. Towsely, "Bootstrapping Deny-By-Default Access Control For Mobile Ad-Hoc Networks" in *IEEE MILCOM 2008, San Diego, November 17-19, 2008*.
- [21] RFC4987. TCP SYN Flooding Attacks and Common Mitigations.
- [22] Marta Kwiatkowska, Gethin Norman, and David Parker. "PRISM: Probabilistic symbolic model checker". In *Proc. TOOLS 2002, V2324 of Lecture Notes in Computer Science*, pp. 200-204, Springer. April 2002.
- [23] H. Hansson and B. Jonsson. "A logic for reasoning about time and reliability". *Formal Aspects of Computing*, 1994, Vol. 6, pages 102-111.
- [24] M. Dufлот, L. Fribourg, T. Herault, R. Lassaigne, F. Magniette, S. Messika, S. Peyronnet, and C. Picaronny. "Probabilistic model checking of the CSMA/CD protocol using PRISM and APMC". In *4th Workshop on Automated Verification of Critical Systems 2004 (AVoCS'04)*.
- [25] Vitaly Shmatikov, "Probabilistic model checking of an anonymity system", *Journal of Computer Security*, 2004, Vol. 12, 2004.
- [26] H. Zhang, P. Kreidl, B. DeCleene, J. Kurose, and X. Ni, "Security Analysis of Bootstrapping Deny-by-default Mobile Ad-hoc Networks," 2009, [ftp://gaia.cs.umass.edu/pub/Zhang09\\_bootstrap\\_security.pdf](ftp://gaia.cs.umass.edu/pub/Zhang09_bootstrap_security.pdf).
- [27] C. He, J. Mitchell, "Security analysis and improvements for IEEE 802.11i" In *NDSS 2005*.