Modeling and Analysis of LEAP, a Key Management Protocol for Wireless Sensor Networks *

Rakesh M. Verma † and Bailey E. Basile ‡

Department of Computer Science
University of Houston
Houston, TX, 77204, USA
http://www.cs.uh.edu

Technical Report Number UH-CS-08-12

August 8, 2008

Keywords: Cryptographic protocols, Wireless sensor networks, model checking

Abstract

A formal analysis of a key management protocol, called LEAP (Localized Encryption and Authentication Protocol), intended for wireless sensor networks is presented in this paper. LEAP is modeled using the high level formal language HLSPL and checked using the AVISPA tool for attacks on the security and authenticity of the exchanges. We focus on the protocol's establishment of pairwise keys for nearest neighbors and for multi-hop neighbors. We then use this foundation to test the protocol's method of cluster key redistribution. Finally, we check LEAP's use of μTESLA, an authentication protocol utilizing a one-way key chain and delayed key disclosure, which LEAP uses for authentication of node revocation messages.

*This work was supported by the National Science Foundation.
†Rakesh Verma is with the University of Houston.
‡Bailey Basile is with Rice University.

Modeling and Analysis of LEAP, a Key Management Protocol for Wireless Sensor Networks

*

Rakesh M. Verma † and Bailey E. Basile ‡

Abstract

A formal analysis of a key management protocol, called LEAP (Localized Encryption and Authentication Protocol), intended for wireless sensor networks is presented in this paper. LEAP is modeled using the high level formal language HLSPL and checked using the AVISPA tool for attacks on the security and authenticity of the exchanges. We focus on the protocol's establishment of pairwise keys for nearest neighbors and for multi-hop neighbors. We then use this foundation to test the protocol's method of cluster key redistribution. Finally, we check LEAP's use of μTESLA, an authentication protocol utilizing a one-way key chain and delayed key disclosure, which LEAP uses for authentication of node revocation messages.

Index Terms

Cryptographic protocols, Wireless sensor networks, model checking

I. Introduction

Wireless sensor networks have wide applications and are predicted to become ubiquitous in the future [1]. A significant advantage of these networks is that they perform in-network processing and data aggregation. On the other hand, these networks pose some unique challenges. For economic reasons they are somewhat limited in their energy, computation and communication power. Furthermore, they also interact with their environments and with people, which means that security is an important issue.

Among other desirable features, a secure network layer for sensor networks should provide data secrecy, authentication, and protection against replay attacks [1]. Traditional solutions for data secrecy and authentication are based on cryptographic protocols, which could typically use either public key or shared secret key cryptographic algorithms. The limitations on computation power and energy consumption make some traditional security protocols, such as public key cryptography, difficult to deploy. Hence new protocol solutions are needed for such networks. These protocols must address several aspects of security: key establishment and trust setup, secrecy, authentication, and privacy. Several such protocols have been proposed in the literature, for example, see [2] and references cited in [1] for key distribution protocols and [3], [4] for authentication and privacy protocols.

We focus on the detailed modeling and formal analysis of key establishment protocol, called LEAP [5], [2], in this paper using the AVISPA tool. One of the LEAP scenarios involved the modeling and analysis of μTESLA [3]. Several methods have been proposed for analyzing protocols in the literature and quite a few have been implemented as well [6], [7], [8], [9], [10]. Our choice of AVISPA is based on its availability on the web, its flexibility and convenience of specification. AVISPA is a model checking toolbox that has four different backends for analysis. The modeling of LEAP in the language of AVISPA poses some interesting questions and provides insights into the protocols as well as the tool itself. We found some problems with LEAP, such as replay attacks, lack of authentication on HELLO messages, and misuse of the muTESLA protocol. and We propose fixes, which we have also reanalyzed using AVISPA, for these problems.

The rest of this paper is organized as follows. Section II describes AVISPA in more detail. In Section III presents the details of LEAP and Section IV presents μTESLA. Section V describes the modeling and analysis. In Section VI we discuss related work and Section VII presents some conclusions.

*This work was supported by the National Science Foundation.
†Rakesh Verma is with the University of Houston.
‡Bailey Basile is with Rice University.

A. Notation

We try to use consistent notation throughout this paper. We specifically note any place where we deviate from the following meanings.

$$
\begin{array}{rcl}
A, B, C, S & = & \text{communicating nodes in the network} \\
N_A & = & \text{a nonce generated by node A} \\
M_1.M_2 & = & \text{message } M_1 \text{ concatenated with message } M_2 \\
\{M\}_K & = & \text{message } M \text{ encrypted with key } K \\
K_{AB} & = & \text{symmetric key between nodes A and B} \\
\{f_K\} & = & \text{a family of pseudo-random functions} \\
MAC(K, M) & = & \text{a message authentication code (MAC)} \\
 & & \text{on message } M \text{ using key } K \\
A \rightarrow B : M & = & \text{message M sent from A to B} \\
A \rightarrow * : M & = & \text{message M broadcast by A}
\end{array}
$$

II. AVISPA

The Automated Validation of Internet Security Protocols and Applications (AVISPA) Tool [10] is a set of applications intended for building formal models of security protocols and for formally analyzing their security properties. AVIPSA provides the High Level Protocol Specification Language (HLPSL) for specifying protocols and their security properties. AVISPA also provides tools for formally verifying these protocols. After the protocol has been modeled using HLPSL, AVISPA translates this high-level model into an intermediate format (IF). The models can then be input to any of four back-ends provided by AVISPA to determine whether security goals are achieved or violated.

The four back-ends, OFMC, CL-AtSe, SATMC, and TA4SP, analyze the model in different ways so each may not provide the same results. In addition, each of the back-ends has a different set of options. For example, the "short" option in CL-AtSe will return the simplest attack on the protocol as specified instead of returning the first attack found. We found OFMC and CL-AtSe most useful as they reliably returned results, allowed us to specify untyped models, and included the xor function. OFMC also provided the "sessco" option which first searches the specification with a passive intruder ensuring that the honest agents can execute the protocol and then runs a second session with an active intruder with all the knowledge of the actions of the first session. This option will detect replay attacks when normal authentication, which provides replay protection, is specified as a security goal. Weak authentication, which does not provide replay protection, can be specified in cases where the protocol is not meant to prevent replay attacks.

In testing our models, we found that AVISPA requires extensive experimentation with the many parameters such as which goals to declare, how many sessions to use, and which options to use. While most secrecy properties are apparent, authentication properties to be checked are often not as obvious, and in our modeling, we erred on the side of authenticating too many values. Also, because the authentication property as AVISPA defines it is necessarily in one direction, we also tried to check all authenticated values in both directions. Secondly, in order to find some of the attacks we found, we had to experiment with the number and members of sessions of the protocol run concurrently by AVISPA. For example, we did not find the advanced replay attacks in which the intruder re-establishes an old key until we had at least three sessions, two with the honest agents and one with the intruder playing one role. We did not expect to find these precise attacks; however, they became clear as we played with the sessions. Although the computing power needed is extensive, we recommend having two sessions with honest agents and then at least one each of the sessions where the intruder plays an honest agent's role. However, we do not make the claim that no more attacks can be found after testing sessions in this manner. In addition, we frequently found that by testing this extensively we found so-called attacks in which the intruder merely acted as the channel, passing messages back and forth. On this note, one of the challenges of modeling in AVISPA is deciphering the returned attacks and determining whether they are truly attacks. Finally, the extensive experimentation also included use of the options available in each back-end, such as the "sessco" and "untyped" options available in OFMC.

III. LEAP

In this section, we first give an overview of Localized Encryption and Authentication Protocol (LEAP) [5] and then we detail the key establishment protocols.

A. Overview

LEAP is a key management protocol intended for large-scale wireless sensor networks where the nodes have limited power, processing, and memory resources. In order to support the in-network processing necessary for most applications of these networks while at the same time providing security properties, such as security and authentication, similar to those of pairwise symmetric keys, LEAP specifies four types of keys: individual keys, pairwise shared keys, cluster keys and group keys.

Individual keys are symmetric keys shared between the base station and each of the nodes. For example, a node might use the individual key to notify the base station of a suspicious neighbor. Pairwise shared keys are symmetric keys shared between a node and each of its neighbors. While pairwise shared keys are used to establish cluster keys, they prevent passive participation which is desirable for in-network processing. Cluster keys are symmetric keys shared between a node and all of its neighbors. These cluster keys can be used for locally broadcast messages such as a routing protocol might use and are also used for updating the group key. The group key, a symmetric key shared between the base station and all of the nodes, allows encrypted and authenticated messages to broadcast through the whole network.

B. Properties and Goals

LEAP's goal is to satisfy the security properties of authentication (which they do not define) and confidentiality in a wireless environment where the intruder may eavesdrop, inject packets, and replay messages. The authors of LEAP also desire that the protocol will be robust and will survive in the face of security attacks and that the effects of any attacks be minimized (to a node's neighbors only, for example). LEAP makes no claims as to defending against replay or denial of service attacks.

C. Key Establishment

LEAP details how each of these keys are established. The key establishment protocols are meant to be lightweight and scalable. While keys may be preloaded prior to deployment in cases where the network architecture is known beforehand, many wireless sensor networks are deployed in environments where such prior knowledge is not possible and keys must be established either during an initial setup phase or on-the-fly as the network architecture changes. In addition, preloaded keys must be updated to prevent cryptanalysis attacks.

1) Individual Keys.: Because the base station and each of the nodes is generally known before deployment, the protocol specifies that the individual keys should be preloaded into the nodes. To save the base station's memory, each of these individual keys is generated using a master key, K_m, and the node's unique ID. For example the key K_{as} shared between the base station (S) and a node (A) would be $K_{as} = f_{K_m}(A)$. Thus, when the base station receives a message from or wishes to send a message to A, the base station can generate the individual key using the stored master key.

2) Pairwise Shared Key with Single-Hop Neighbor.: Establishing a pairwise shared key with a single-hop neighbor requires four phases. First, each node calculates a master key from an initial key, K_i, preloaded into all the nodes and the nodes unique name. Second, each node broadcasts a HELLO message with its name and a nonce. Then each neighbor replies to the message with a message authenticated using the message authentication code (MAC) with its master key, which the initial node can check using the neighbor's identity and the initial key. Third, the two nodes calculate the pairwise shared key. Finally, once the initial setup timer has expired, the nodes erase both the initial key and their neighbors' master keys. For a requestor node A and its neighbor B, the protocol, thus, requires the following two messages and calculations.

$$\begin{aligned} K_A &= f_{K_i}(A) \\ K_B &= f_{K_i}(B) \end{aligned} \tag{1}$$

$$\begin{aligned} A \rightarrow * &: A.N_A \\ B \rightarrow A &: B.MAC(K_B, N_A.B) \end{aligned} \tag{2}$$

$$K_{AB} = f_{K_B}(A) \tag{3}$$

3) Pairwise Shared Key with Multi-Hop Neighbor.: To establish a pairwise shared key with a multi-hop neighbor requires the use of shared neighbors, called proxies. If a node, A, wishes to establish a pairwise shared key with its multi-hop neighbor, C, A would first broadcast a QUERY message to all its neighbors with its ID and the ID of the desired node. The n shared neighbors, B_i, which are single-hop neighbors of A and singe-hop neighbors of C, send a REPLY message authenticated with the pairwise shared key. Node A splits the new key K_{AC} into n randomly generated shares K_i such that $K_{AC} = K_1 \oplus K_2 \oplus \ldots \oplus K_n$. A sends these shares to the proxies encrypted with a pairwise shared key and a verification key $f_{K_i}(0)$. The proxies then re-encrypt the share with the pairwise key shared between the proxy and C, and the proxies forward the key and the verification key to C. Finally when C receives the shares from the proxies, C verifies the shares, recreates the new key by $K_{AC} = K_1 \oplus K_2 \oplus \ldots \oplus K_n$, and sends A a DONE message encrypted with the new key.

4) Cluster Key.: The cluster key is established by a node by randomly generating a cluster key and then sending the key to its neighbors encrypted with pairwise shared keys.

5) Group Key.: Group keys are preloaded into all the nodes prior to deployment. However, because this key is shared among all the nodes in the network, when one of the nodes is compromised, the group key must be redistributed so that the intruder cannot send and read encrypted messages for the network. In addition, this key should be updated regardless of node revocation to prevent cryptanalysis.

6) Node Revocation and Key Redistribution.: Once a node is identified as compromised, the base station generates a new group key $K_g^{'}$ and sends a message authenticated using the μTESLA protocol (see Section IV) identifying the revoked node (C) and providing a verification key $f_{K_g^{'}}(0)$ for the new key. The μTESLA key is released one μTESLA interval later allowing the nodes to authenticate the node revocation message. In order to remove the revoked node from the network, each node deletes any pairwise keys shared with C and, using the same process as the first establishment, establishes new keys for cluster keys compromised by C. The base station sends out the new group key encrypted with a cluster key to its children who in turn verify the new group key and then send it to their children encrypted with a cluster key and so on until all nodes have the new group key. Here LEAP assumes a breadth-first spanning tree established by a routing protocol.

IV. μTESLA

μTESLA [3] is an authentication protocol intended for wireless sensor networks. μTESLA assumes that the base station and the nodes are loosely synchronized and know the upper bound of the time discrepancy. First, the base station derives a key chain from a randomly generated key K_n using a one-way hash function f; namely, $K_i = f(K_{i+1})$. The base station then uses these keys in the reverse order from which it calculated them. The base station authenticates each message sent during a time period T_i with the key K_i. When the nodes receive these messages, they buffer them until the μTESLA key is released. The base station then waits δ μTESLA time intervals until $T_{i+\delta}$ to release the key to all the nodes. The nodes can then check the key using a previous key, $K_i = f(K_{i+1})$, and the authentication of buffered messages.

μTESLA also provides a protocol for bootstrapping a receiver node, providing the node an initial key and information about synchronization. Namely,

$$\begin{aligned} A \rightarrow S: &\quad N_A \\ S \rightarrow A: &\quad T_S.K_i.T_i.T_{int}.\delta. \\ &\quad MAC(K_{AS}, T_S.K_i.T_i.T_{int}.\delta) \end{aligned}$$

where T_S is the base station's current time, K_i is the most recently released key, T_i is the beginning of the time interval in which K_i was used, T_{int} is the length of the μTESLA time interval, δ is the number of intervals before each key is released, and K_{AS} is a symmetric key shared by A and S.

V. Modeling and Analysis

We created four different models in our formal analysis of LEAP: establishing pairwise shared keys with single-hop neighbors, establishing pairwise shared keys with a multi-hop neighbor, establishing a cluster key, and μTESLA. Each of these protocols must be secure in order for the node revocation and key redistribution protocol to be secure. We focused on the LEAP's key establishment protocols rather than the use of the keys because with proper use of the symmetric keys communications will remain secure.

A. Change in Notation

Because HLPSL does not have a way to define either f or MAC as previously noted, we use the following changed notation for this section.

$$\begin{aligned} &\text{For Creating New Keys} \\ f(K.A) &= \text{a one-way hash function using key } K \text{ on A} \\ &\text{For Verification Keys} \\ f(0)_K &= \text{a one-way hash function using key } K \text{ on A} \\ &\text{For Message Authentication} \\ \{MAC(M)\}_K &= \text{a CBC-MAC using key } K \text{ on message } M \end{aligned}$$

B. Definitions

Throughout the following sections we refer to authentication and secrecy properties. We use these terms as they are defined by the AVISPA creators[10], namely, that the secrecy property is violated when the intruder discovers a value declared as a secret from the intruder, and that the authentication property is violated when an agent's request to verify the value of a variable is witnessed by another agent with different value for this variable. Any authentication or secrecy attack was found by AVISPA and is based on these definitions.

C. Single-Hop Pairwise Shared Key

We considered two different cases of this model. First, we looked at the case where node A establishes a key with only one of its neighbors, B. Then we looked at the case where A,B, and C all establish pairwise keys with each other. We consider them separately here.

1) Establishing One Single-Hop Pairwise Shared Key.: With the first model LEAP specifies exactly the message, so the model first enacts this sequence of messages. Then, B and A exchange a test message using the same format as Dolev and Yao [11] showed to be safe using public keys:

$$\begin{aligned} &\text{HELLO} \\ &\qquad A \rightarrow * : \quad A.N_A \\ &\text{REPLY} \\ &\qquad B \rightarrow A : \quad B.\{MAC(N_A.B)\}_{K_B} \\ &\text{Test Messages} \\ &\qquad A \rightarrow B : \quad A.\{M.A\}_{K_{AB}}.B \\ &\qquad B \rightarrow A : \quad B.\{M.B\}_{K_{AB}}.A \end{aligned}$$

$$\begin{aligned} \text{where} \quad K_A &= f(K_i.A) \\ K_B &= f(K_i.B) \\ K_{AB} &= f(K_B.A) \end{aligned}$$

We evaluated the model based on the secrecy of the new keys and the secrecy of the data in the test messages. We also defined as goals the one-way authentication of the HELLO messages and the two-way authentication of the test messages.

We found several attacks on this protocol. Using the "sessco" option of OFMC, we found simple replay attacks at first. The intruder (I) replays A's earlier messages and B responds as before.

$$\begin{aligned} A \rightarrow * &: \quad A.N_A \\ B \rightarrow A &: \quad B.\{MAC(N_A.B)\}_{K_B} \\ A \rightarrow B &: \quad A.\{M.A\}_{K_{AB}}.B \\ B \rightarrow A &: \quad B.\{M.B\}_{K_{AB}}.A \\ &\text{Replay} \\ I \rightarrow * &: \quad A.N_A \\ B \rightarrow I &: \quad B.\{MAC(N_A.B)\}_{K_B} \\ I \rightarrow B &: \quad A.\{M.A\}_{K_{AB}}.B \\ B \rightarrow I &: \quad B.\{M.B\}_{K_{AB}}.A \end{aligned}$$

Not only does LEAP not protect against replay attacks, but also this attack produces only wasted resources. We found replay attacks in all of the following models but we will not mention them.

Secondly, we found a secrecy attack in which the intruder is a member of the network and is aware of the initial key K_i and the hash function f. Here the intruder is successfully able to establish a pairwise key with A while pretending to be B.

$$\begin{aligned} A \rightarrow * :&\quad A.N_A \\ I \rightarrow A :&\quad B.\{MAC(N_A.B)\}_{K_B} \\ A \rightarrow I :&\quad A.\{M.A\}_{K_{AB}}.B \end{aligned}$$

LEAP, however, assumes that the initial setup of pairwise keys is finished, including the erasing of setup keys, in an amount of time much smaller than the amount of time an intruder would take to compromise a node and discover the initial key.

The third attack we found was an authentication attack on the initial HELLO message. The intruder changes A's initial HELLO message and is able to cause A to create a pairwise key with B while B creates a pairwise key with the intruder.

$$\begin{aligned} A \rightarrow * :&\quad A.N_A \\ I \rightarrow B :&\quad I.N_A \\ B \rightarrow I :&\quad B.\{MAC(N_A.B)\}_{K_B} \\ I \rightarrow A :&\quad B.\{MAC(N_A.B)\}_{K_B} \\ A :&\quad K_{AB} = f(K_B.A) \\ B :&\quad K_{IB} = f(K_B.I) \end{aligned}$$

Although the intruder does not discover either A's new key or B's new key, he has violated the authentication goal as defined by AVISPA. In addition, if B were to request a key from A because B does not yet have a shared key with A, A would likely mark B as compromised because A believes B already has a shared key. Thus, extensive attacks of this nature could cripple a wireless sensor network. Based on this attack, we suggest a fix on the original protocol in which the HELLO messages are authenticated using the sender's master key.

$$\begin{aligned} A \rightarrow * :&\quad A.N_A.\{MAC(A.N_A)\}_{K_A} \\ B \rightarrow A :&\quad B.MAC(K_B, N_A.B) \end{aligned}$$

$$\begin{aligned} \text{where}\quad K_A &= f(K_i.A) \\ K_B &= f(K_i.B) \\ K_{AB} &= f(K_B.A) \end{aligned}$$

In the more recent paper on LEAP+ [2], the authors suggest that the lack of authentication on the HELLO message would produce only resource attacks; however, we have discovered an attack in which the lack of authentication produces more than just a resource attack. They have suggested two solutions to this problem: using the current group key to authenticate the packet or adding randomness to added nodes' ids.

2) Establishing Single-Hop Pairwise Shared Keys among Three Neighbors.: We also examined the case where three neighbors must establish pairwise keys between them. LEAP specifies that if two nodes both send HELLO's, the first HELLO is given precedence and takes the role indicated by A in the description of LEAP (Section III-C.2); therefore, in this model, A establishes keys with both B and C because its HELLO is first, and then C establishes a key with B because its HELLO is second. We used the following sequence of messages.

$$
\begin{array}{rrl}
\text{HELLO} & & \\
& A \rightarrow * : & A.N_A \\
& C \rightarrow B : & C.N_C \\
\text{REPLY} & & \\
& B \rightarrow A : & B.\{MAC(N_A.B)\}_{K_B} \\
& C \rightarrow A : & C.\{MAC(N_A.C)\}_{K_C} \\
& B \rightarrow C : & B.\{MAC(N_C.B)\}_{K_B} \\
\text{Test Messages} & & \\
& A \rightarrow B : & A.\{M_1.A\}_{K_{AB}}.B \\
& A \rightarrow C : & A.\{M_1.A\}_{K_{AC}}.C \\
& C \rightarrow B : & C.\{M_2.C\}_{K_{BC}}.B \\
& B \rightarrow A : & B.\{M_1.B\}_{K_{AB}}.A \\
& B \rightarrow C : & B.\{M_2.B\}_{K_{BC}}.C \\
& C \rightarrow A : & C.\{M_1.C\}_{K_{AC}}.A
\end{array}
$$

$$
\begin{aligned}
\text{where} \quad K_A &= f(K_i.A) \\
K_B &= f(K_i.B) \\
K_C &= f(K_i.C) \\
K_{AB} &= f(K_B.A) \\
K_{AC} &= f(K_C.A) \\
K_{BC} &= f(K_B.C)
\end{aligned}
$$

Here again we used a test message format from Dolev and Yao [11]. We evaluated the model based on the secrecy of the new keys and the secrecy of the data in the test messages. We also tested the one-way authentication of the HELLO messages and the two-way authentication of the test messages.

In this case the most significant attack found in AVISPA was again on the authentication of the HELLO messages. By exploiting the unauthenticated HELLO messages, the intruder causes B to not respond to C while convincing B that a reply B means for C goes to A. Thus, B and A and B and C do not agree on the value of the HELLO message.

$$
\begin{array}{rl}
A \rightarrow I : & A.N_A \\
C \rightarrow I : & C.N_C \\
& \\
I \rightarrow B : & A.N_I \\
B \rightarrow I : & B.\{MAC(N_I.B)\}_{K_B} \\
& \\
I \rightarrow B : & C.N_A \\
B \rightarrow I : & B.\{MAC(N_A.B)\}_{K_B} \\
I \rightarrow A : & B.\{MAC(N_A.B)\}_{K_B} \\
& \\
I \rightarrow C : & A.N_A \\
C \rightarrow I : & C.\{MAC(N_A.C)\}_{K_C} \\
I \rightarrow A : & C.\{MAC(N_A.C)\}_{K_C}
\end{array}
$$

Again we suggest authenticating the HELLO message in order to prevent this attack.

D. Multi-Hop Pairwise Shared Key

We modeled the establishment of the multi-hop pairwise shared key using only two proxies. In this case, node A wishes to establish a pairwise key with S through proxies B and C. The exchange of messages follows.

$$
\begin{array}{rl}
\text{QUERY} & \\
A \rightarrow * : & A.S \\
\text{REPLY} & \\
B \rightarrow A : & B.A.S.\{MAC(B.A.S)\}_{K_{AB}} \\
C \rightarrow A : & C.A.S.\{MAC(C.A.S)\}_{K_{AC}} \\
\text{Shares to Proxies} & \\
A \rightarrow B : & \{K_1\}_{K_{AB}}.\{f(0)\}_{K_1} \\
A \rightarrow C : & \{K_2\}_{K_{AC}}.\{f(0)\}_{K_2} \\
\text{Shares to Destination} & \\
B \rightarrow S : & \{K_1\}_{K_{BS}}.\{f(0)\}_{K_1} \\
C \rightarrow S : & \{K_2\}_{K_{CS}}.\{f(0)\}_{K_2} \\
\text{Done Message} & \\
S \rightarrow A : & \{0\}_{K_{AS}}
\end{array}
$$

where $K_{AS} = K_1 \oplus K_2$

Here we have used zero as the DONE message rather than using a freshly generated nonce, because we assume that the nodes have a pre-established message that means “Done”. Because A and S must already know zero and agree on it to check the verification keys, A can check this done message unlike a done message modeled with a nonce. Also, LEAP, saying only that the REPLY is authenticated using a pairwise shared key does not expressly give the REPLY message format for B and C, so we have modeled it as shown. Because none of the attacks found involved exploitation of this message, we did not try any alternate packet formats. Our model tests the secrecy of the new key and the authentication of the key in both directions.

The only secrecy attack that AVISPA found was one in which the intruder has compromised all proxies. In this case the intruder will be able to read each of the shares and form the new key while at the same time sending the key on to S. LEAP, however, assumes that at most $n-1$ of the n proxies are compromised.

AVISPA found two authentication attacks on this protocol both of which are advanced replay attacks. The first attack involved the establishment of the original key and a reestablishment; by replaying a message from the original session of the protocol, the intruder is able to give S the wrong key. The agents go through the first session of the protocol as usual with the intruder copying and saving the messages. When the protocol is started again with the second session, the following attack occurs. New key shares are noted with primes, and old keys shares from the previous session are unprimed.

$$
\begin{array}{rrl}
\text{Unauthenticated QUERY} & & \\
I & \rightarrow * : & A.S \\
B & \rightarrow I : & B.A.S.\{MAC(B.A.S)\}_{K_{AB}} \\
C & \rightarrow I : & C.A.S.\{MAC(C.A.S)\}_{K_{AC}} \\
\text{Actual QUERY} & & \\
A & \rightarrow I : & A.S \\
I & \rightarrow A : & B.A.S.\{MAC(B.A.S)\}_{K_{AB}} \\
I & \rightarrow A : & C.A.S.\{MAC(C.A.S)\}_{K_{AC}} \\
\text{Shares to Proxies} & & \\
A & \rightarrow B : & \{K_1'\}_{K_{AB}}.\{f(0)\}_{K_1'} \\
A & \rightarrow I : & \{K_2'\}_{K_{AC}}.\{f(0)\}_{K_2'} \\
I & \rightarrow C : & \{K_2\}_{K_{AC}}.\{f(0)\}_{K_2} \\
\text{Shares to Destination} & & \\
B & \rightarrow S : & \{K_1'\}_{K_{BS}}.\{f(0)\}_{K_1'} \\
C & \rightarrow S : & \{K_2\}_{K_{CS}}.\{f(0)\}_{K_2} \\
\text{Done Message} & & \\
 & K_{AS}' = & K_1' \oplus K_2 \\
S & \rightarrow A : & \{0\}_{K_{AS}'}
\end{array}
$$

While the message interception aspect of this attack may not be possible because of the broadcast nature of a wireless communications environment, we can see that S will not receive the correct key if even one of the proxies is compromised, contrary to LEAP's assumption that the protocol is secure up to $n - 1$ of the n proxies being compromised.

The second attack is similar but more damaging. This attack occurs after two sessions of the protocol have occurred. The intruder saves the messages from the first instance of the protocol, and then causes S to revert to the previous key. Because nodes sleep between communication in order to save on power consumption, the intruder utilizes the time that A is asleep to instantiate the protocol again with B and C, sending the first key to S again. While A and S will not be able to communicate, the intruder may have had time to perform cryptanalysis on the original K_{AS} and read the encrypted messages sent by S to A. Thus, this attack is very serious.

While AVISPA does not directly model that agents can "sleep", by using a Dolev-Yao intruder model [11] in which the intruder can intercept messages it effectively models such a situation. In addition, in our modeling we had the intruder play the role of A in the third session of the composition, which successfully modeled node A being asleep.

We suggest that the QUERY message be authenticated with a cluster key in order to improve the security of this protocol. While this fix does not solve the problem of these advanced replay attacks, it does prevent an element of the attacks that AVISPA found in which the intruder begins the protocol with the proxies before the honest sensor begins the protocol.

E. Cluster Key

In this model, node A establishes a cluster key K_C with its two neighbors, B and C. After establishing the key, each node broadcasts a test encryption message with randomly generated nonce and its ID encrypted with the cluster key. The exchange follows.

$$\begin{array}{rrl}
\text{Distribute Cluster Key} & & \\
& A \rightarrow B: & \{K_C\}_{K_{AB}} \\
& A \rightarrow C: & \{K_C\}_{K_{AC}} \\
\text{Test Cluster Key} & & \\
& A \rightarrow *: & \{N_A.A\}_{K_C} \\
& B \rightarrow *: & \{N_B.B\}_{K_C} \\
& C \rightarrow *: & \{N_C.C\}_{K_C}
\end{array}$$

While LEAP does not indicate DONE messages or the content of any messages, we chose to test the use of the key using the message content format shown by Dolev and Yao [11] to be safely exchanged using public encryption keys. Our model's secrecy goals were on the new key and the three nonces. We also tested the authentication on the key and the three nonces.

We found two authentication attacks on this model. The first attack AVISPA found was one in which the intruder prevents one of the neighbor nodes from receiving the new cluster key but allows that same neighbor to receive the test messages. While such an attack is difficult in a wireless environment, it would be possible for the intruder to jam a message by sending a high-energy signal [1]. LEAP is not meant to prevent such denial-of-service attacks, and, in fact, they do not assume that the intruder can remove messages from the network, but this attack has significant implications on LEAP's key exchanges.

The second attack was akin to the second attack on the multi-hop pairwise shared key. After two sessions of the protocol (one keying and a second re-keying), the intruder can send the original key to the neighbors while A is asleep. If this key has already been compromised, the intruder will be able to read any messages the neighbors send using this old key. Also, because the cluster key is used to update the group key, this attack could prevent nodes from receiving the new group key.

F. μTESLA

We model μTESLA using a base station, S, and a node A. First, node A is bootstrapped, and then S sends three test messages to A along with a message disclosing the last key in the chain.

$$\begin{array}{rrl} \text{Bootstrap Request} & & \\ & A \rightarrow S: & N_A \\ \text{Bootstrap} & & \\ & S \rightarrow A: & K_a.M_1.\{MAC(K_a.M_1)\}_{K_{AS}} \\ \text{Messages} & & \\ & S \rightarrow I: & K_a.M_2.\{MAC(M_2)\}_{K_b} \\ & S \rightarrow I: & K_b.M_3.\{MAC(M_3)\}_{K_c} \\ & S \rightarrow A: & K_c.M_4.\{MAC(M_4)\}_{K_d} \\ & S \rightarrow A: & K_d \end{array}$$

where $K_a = f(K_b) = f(f(K_c)) = f(f(f(K_d)))$

Because AVISPA does not model time accurately, we chose to model the synchronization part of the bootstrapping message as randomly generated data. For this same reason, we also chose to model the time intervals with a series of separate intervals. Because μTESLA does not specify how the keys are released, we chose to release the keys with the next message (and therefore in the next time interval) that needs authentication. μTESLA specifies that the keys are released after δ time intervals; although the implementation of the authors of μTESLA uses two, LEAP specifies that the keys are released after only one interval; because our main goal is testing LEAP, we wait only one interval. We tested the model for the authentication of the request message, the data in the bootstrap message, and the three test messages.

AVISPA found the following attack on the authentication of the request message and the authentication of one of the test messages.

$$\begin{array}{rrl} \text{Bootstrap Request} & & \\ & A \rightarrow I: & N_A \\ & I \rightarrow S: & N_I \\ \text{Bootstrap} & & \\ & S \rightarrow A: & K_a.M_1.\{MAC(K_a.M_1)\}_{K_{AS}} \\ \text{Delayed Messages} & & \\ & S \rightarrow I: & K_a.M_2.\{MAC(M_2)\}_{K_b} \\ & S \rightarrow I: & K_b.M_3.\{MAC(M_3)\}_{K_c} \\ \text{False Message} & & \\ & I \rightarrow A: & K_a.M_I.\{MAC(M_I)\}_{K_b} \end{array}$$

where M_I is a message from the intruder.

First, this attack assumes that the intruder can intercept and delay messages which contradicts the broadcast nature of wireless networks. In addition, this attack takes advantage of the lack of proper time synchronization with the AVISPA tool; however, because the node and the base station may not be perfectly synchronized, it is possible for the intruder to use the time discrepancy to send the first message after the server's interval has lapsed and the server has released the old key but before the node's interval has lapsed. Such an attack is possible only when keys are released in the next interval. If the release delay is more than one interval, then such an attack is not possible as long as the maximum synchronization error as defined by μTESLA is less than the duration of one interval. LEAP, however, releases the previous key in the next interval and thus, is susceptible to this serious attack.

Based on this attack, we suggest that LEAP utilize the μTESLA protocol with the two-interval wait instead of the single-interval wait. While this does extend the time required to revoke a node and redistribute a new group key, it provides additional security.

Also note that the intruder has used the unauthenticated request message in his attack. While the lack of authentication in the request message does no damage in this instance nor in the LEAP protocol, in another use of μTESLA such an attack could be fatal. Hence, we recommend that the protocol authenticate the request with a MAC using a pairwise shared symmetric key, such as the individual key from LEAP.

VI. Related Work

Security of wireless sensor networks is an emerging topic of interest. However, to the best of our knowledge, there are only a handful of very recent papers on formal analysis of wireless sensor network protocols [12], [13], [14]. In [12], the authors analyze (also with AVISPA) a combination of the TinySec authentication protocol with three scenarios of the LEAP protocol (pairwise-key, cluster key and multi-hop). Two of their scenarios deal only with the TinySec protocol, which we do not study here. As for the LEAP scenarios we note the following.

In [12] the authors report a man-in-the-middle attack for pairwise-key establishment, whereas we also report a replay attack and a secrecy attack with an insider attacker besides a similar man-in-the-middle attack. For the cluster key case [12] report a type confusion flaw with a DONE message, whereas we found two authentication attacks. Let us also note that the LEAP paper does not specify a DONE message for the cluster key scenario (even if there was a DONE message it would presumably be something whose type is known to the sensors). For the multi-hop case [12] report a replay attack, whereas we discovered two authentication attacks. As far as replay attacks are concerned, as mentioned above, we do not report them since they are known to exist for all scenarios and their prevention is a separate issue that must be addressed with other techniques not included in the TinySec and LEAP papers.

In [13] the authors analyze a different protocol (again with AVISPA) called Minisec together with two LEAP scenarios: establishing and testing a single-hop pairwise shared key and establishing and testing a cluster key. They use an authenticated HELLO in the first case and do not find any attacks. They also find no attacks in the second case because they do not use the done message as they did in [12]. We believe that the two attacks we report were missed since the session definitions of AVISPA require extensive experimentation. In fact, the second authentication attack we found required three sessions, the last of which required the intruder playing the role of an honest agent in the session. Note that the absence of an attack with AVISPA gives a measure of confidence, but it is not a *proof* of protocol security since AVISPA (and other similar tools) must impose a bound on the role instances to get decidability. We have not been able to access the most recent paper [14] so far.

VII. Conclusions

We performed a formal analysis of LEAP using the AVISPA tool. We specifically looked at LEAP's key establishment protocols for pairwise shared keys between single-hop neighbors, for pairwise shared keys between multi-hop neighbors, and for cluster keys. We also modeled μTESLA in order to determine whether LEAP's use of μTESLA for the authentication of node revocation messages is secure. We modeled these cases because they are all necessary foundations to LEAP's group key redistribution plan.

We found that the protocol for establishing pairwise shared keys for single-hop neighbors is vulnerable to replay attacks, secrecy attacks from compromised nodes, and authentication attack on the HELLO message. LEAP does not protect against replay attacks, and the secrecy attack requires that one of the protocol's assumptions is violated. We recommend that LEAP utilize a HELLO message authenticated with the node's master key to overcome the authentication attack.

We found that the protocol for establishing pairwise shared keys for multi-hop neighbors is vulnerable to a secrecy attack when all the proxies are compromised, an authentication attack on the key by a single compromised proxy, and an advanced replay attack that re-establishes an old key. While none of these attacks hinge on the lack of authentication in the QUERY message, we suggest adding authentication to this message as the intruder QUERYs before the honest agent does in the attacks we found.

We found that the protocol for establishing cluster keys is vulnerable to a denial-of-service attack on the receiving neighbors and an advanced replay attack that re-establishes an old key. LEAP, however, protects against neither denial-of-services attacks nor replay attacks.

Finally, we found that LEAP's implementation of μTESLA is flawed because it releases the previous key after only one μTESLA interval, allowing an intruder to take advantage of the synchronization error between the base station and the receiving nodes. Although we found no attacks which hinged on the lack of authentication in the bootstrap request message, the attack we found did utilize this void.

Proposed fixes were effective in our experiments in some cases as mentioned above. We also found that modeling time with AVISPA is nontrivial and that one has to experiment with many different session scenarios in AVISPA so as not to miss any attacks.

REFERENCES

[1] A. Perrig, J. A. Stankovic, and D. Wagner, "Security in wireless sensor networks," *Commun. ACM*, vol. 47, no. 6, pp. 53–57, 2004.

[2] S. Zhu, S. Setia, and S. Jajodia, "Leap+: Efficient security mechanisms for large-scale distributed sensor networks," *TOSN*, vol. 2, no. 4, pp. 500–528, 2006.

[3] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler, "Spins: Security protocols for sensor networks," *Wireless Networks*, vol. 8, no. 5, pp. 521–534, 2002.

[4] C. Karlof, N. Sastry, and D. Wagner, "Tinysec: a link layer security architecture for wireless sensor networks," in *SenSys*, 2004, pp. 162–175.

[5] S. Zhu, S. Setia, and S. Jajodia, "Leap: efficient security mechanisms for large-scale distributed sensor networks," in *ACM Conference on Computer and Communications Security*, 2003, pp. 62–72.

[6] C. Meadows, "The NRL protocol analyzer: An overview," *J. Log. Program.*, vol. 26, no. 2, pp. 113–131, 1996.

[7] J. K. Millen and V. Shmatikov, "Constraint solving for bounded-process cryptographic protocol analysis," in *ACM Conference on Computer and Communications Security*, 2001, pp. 166–175.

[8] R. Corin, S. Etalle, and A. Saptawijaya, "A logic for constraint-based security protocol analysis," in *IEEE Symposium on Security and Privacy*, 2006, pp. 155–168.

[9] J. C. Mitchell, M. Mitchell, and U. Stern, "Automated analysis of cryptographic protocols using mur-phi," in *IEEE Symposium on Security and Privacy*, 1997, pp. 141–151.

[10] A. Armando, D. A. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuéllar, P. H. Drielsma, P.-C. Héam, O. Kouchnarenko, J. Mantovani, S. Mödersheim, D. von Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Viganò, and L. Vigneron, "The AVISPA tool for the automated validation of internet security protocols and applications," in *CAV*, 2005, pp. 281–285.

[11] D. Dolev and A. C.-C. Yao, "On the security of public key protocols," *IEEE Transactions on Information Theory*, vol. 29, no. 2, pp. 198–207, 1983.

[12] L. Tobarra, D. Cazorla, F. Cuartero, G. Daz, and E. Cambronero, "Model Checking Wireless Sensor Network Security Protocols: TinySec + LEAP," in *Proc. of the First IFIP International Conference on Wireless Sensor and Actor Networks (WSAN'07).* IFIP Main Series, Springer, September 2007, pp. 95–106.

[13] L. Tobarra, D. Cazorla, F. Cuartero, and G. Diaz, "Analysis of security protocol MiniSec for Wireless Sensor Networks," in *Proc. of the IV Congreso Iberoamericano de Seguridad Informatica (CIBSI'07)*, November 2007, pp. 1–13.

[14] M. Katelman, J. Meseguer, and J. C. Hou, "Redesign of the lmst wireless sensor protocol through formal modeling and statistical model checking," in *FMOODS*, 2008, pp. 150–169.