Practical Broadcast Authentication in Sensor Networks [*]

Donggang Liu Peng Ning
Cyber Defense Laboratory
Department of Computer Science
North Carolina State University
{dliu,pning}@ncsu.edu

Sencun Zhu
Department of Computer
Science and Engineering
The Pennsylvania State University
szhu@cse.psu.edu

Sushil Jajodia
Center for Secure
Information Systems
George Mason University
jajodia@gmu.edu

Abstract

Broadcast authentication is a critical security service in sensor networks; it allows a sender to broadcast messages to multiple nodes in an authenticated way. μTESLA and multi-level μTESLA have been proposed to provide such services for sensor networks. However, none of these techniques are scalable in terms of the number of senders. Though multi-level μTESLA schemes can scale up to large sensor networks (in terms of receivers), they either use substantial bandwidth and storage at sensor nodes, or require significant resources at senders to deal with DOS attacks. This paper presents efficient techniques to support a potentially large number of broadcast senders using μTESLA instances as building blocks. The proposed techniques are immune to the DOS attacks. This paper also provides two approaches, a revocation tree based scheme and a proactive distribution based scheme, to revoke the broadcast authentication capability from compromised senders. The proposed techniques are implemented, and evaluated through simulation on TinyOS. The analysis and experiment show that these techniques are efficient and practical, and can achieve better performance than the previous approaches.

1. Introduction

A wireless sensor network (WSN) typically consists of a large number of resource constrained sensor nodes and possibly a few powerful control nodes (called *base stations*). A sensor node usually has one or a few sensing components, which sense physical phenomenon (e.g., temperature) from its immediate surroundings, and a processing and communication component, which performs simple computation on the sensed data and communicates with base stations as well as other nodes through its immediate neighbor nodes. The control nodes may further process the data collected from sensor nodes, disseminate control commands to sensor nodes, and connect the network to a traditional wired network. Sensor nodes are expected to be deployed densely in a large scale and communicate with each other through wireless links without any infrastructure support [1]. These features lead to many attractive applications in military and civilian operations, such as target tracking and battlefield surveillance.

A sensor network may be deployed in hostile environments where there are malicious attacks. In such a situation, security becomes one of the major concerns. As a fundamental security service, broadcast authentication enables a sender to broadcast critical data and/or commands to sensor nodes in an authenticated way such that an attacker is unable to forge any message from the sender. However, due to the resource constraints on sensor nodes, traditional broadcast authentication techniques such as public key based digital signatures are not desirable.

Perrig et al. developed μTESLA for broadcast authentication in sensor networks based on symmetric cryptography [2], which removes the dependence on public key cryptography. Several multi-level μTESLA schemes have been proposed to extend the capability of the original μTESLA protocol [3, 4]. Despite these recent advances, several issues are still not properly addressed.

- *Scalability (in terms of the number of senders)*: In addition to base stations, many sensor network applications have a large number of other potential senders (e.g., mobile sinks, soldiers). For example, in battlefield surveillance where a sensor network is deployed to monitor the activities in an area, there may be a large number of soldiers or tanks, and each of them entering this area may broadcast queries to collect critical information from the network.

  One way to deal with multiple senders is to let the base station replay the message for each sender. However, this introduces a lot of communication overhead in the network, especially for those sensor nodes that are close to the base station. An alternative way is to

[*]This work is supported by the National Science Foundation under grants CAREER-0447761 and CNS-0430223.

let each sensor node store the initial μTESLA parameters (e.g., the key chain commitments) for all possible senders. In addition, sensor nodes usually use EEPROM to store a large amount of sensed data. Thus, it is not practical for resource constrained sensor nodes to store all parameters when there are a large number of senders.

- *DOS attacks*: The multi-level μTESLA schemes scale broadcast authentication up to large networks by constructing multi-level key chains and distributing initial parameters of lower-level μTESLA instances with higher-level ones. However, multi-level μTESLA schemes magnify the threat of DOS attacks. An attacker may launch DOS attacks on the messages carrying the initial μTESLA parameters [3, 4]. Though several solutions have been proposed in [4], they either use substantial bandwidth or require significant resources at senders.

- *Revocation*: Some senders may be captured and compromised by adversaries in hostile environments. As a result, an adversary may exploit the broadcast authentication capabilities of the compromised nodes to attack the network (e.g., consume sensors' battery power by instructing them to do unnecessary operations). Thus, it is necessary to revoke the broadcast authentication capabilities of the compromised senders once they are detected.

In this paper, we develop an efficient method for μTESLA to support a large number of senders over a long period of time. Our method has the following advantages over the multi-level μTESLA schemes [3, 4]: (1) Our scheme allows broadcast authentication in large sensor networks with a large number of senders, while multi-level μTESLA schemes (as well as the original μTESLA protocol) is not scalable in terms of the number of senders. (2) Our method is not subject to the DOS attacks against the distribution of μTESLA parameters. In contrast, multi-level μTESLA schemes either consume substantial bandwidth or require significant resources at senders in order to defeat such DOS attacks.

To deal with the limited packet payload size in sensor networks, we adopt the idea in [5] and develop a simple method to distribute large messages required for authenticating μTESLA parameters over multiple packets. A nice property of this method is that it allows immediate authentication of the segments of such messages, and thus is immune to DOS attacks. We also develop two complementary techniques to revoke broadcast authentication capability from compromised senders: a revocation tree based scheme and a proactive distribution scheme. The former constructs a Merkle hash tree to revoke compromised senders, while the latter proactively controls the distribution of broadcast authentication capability of each sender to allow the revocation of compromised senders. To gain further understanding of the proposed techniques, we evaluate them through simulation using TinyOS [6], an operating system for networked sensors. The results indicate that the proposed techniques are efficient and practical, and can achieve better performance than the previous methods.

The remainder of this paper is organized as follows. Section 2 presents the proposed techniques for broadcast authentication in sensor networks. Section 3 discusses the simulation evaluation of the proposed techniques. Section 4 reviews related work on sensor network security. Section 5 concludes this paper and points out several future research directions.

2. Practical Broadcast Authentication in WSN

In this section, we develop a series of techniques to support a large number of senders and to revoke broadcast authentication capabilities from compromised senders. The proposed techniques use the μTESLA broadcast authentication protocol [2] as a building block. In other words, these techniques use multiple μTESLA instances with different parameters to provide additional capabilities related to broadcast authentication.

We assume there is an off-line *central server* with computation power and storage equivalent to a regular PC, which is used to pre-compute and store certain parameters. We assume the central server is well protected. We assume a sender has enough storage to save, or enough computation power to generate one or several μTESLA key chains. We assume the clocks on sensor nodes are *loosely* synchronized, as required by the μTESLA protocol [2].

2.1. Overview of μTESLA

An asymmetric mechanism such as public key cryptography is generally required for broadcast authentication. Otherwise, a malicious receiver can easily forge any packet from the sender. μTESLA introduces asymmetry by delaying the disclosure of symmetric keys [2]. A sender broadcasts a message with a Message Authentication Code (MAC) generated with a secret key K, which is disclosed after a certain period of time. When a receiver gets this message, if it can ensure that the packet was sent before the key was disclosed, the receiver buffers this packet and authenticates the packet when it later receives the disclosed key. To continuously authenticate broadcast packets, μTESLA divides the time period for broadcast into multiple intervals, assigning different keys to different time intervals. All packets broadcast in a particular time interval are authenticated with the same key assigned to that time interval.

To authenticate the broadcast messages, a receiver first authenticates the disclosed keys. μTESLA uses a one-way key chain for this purpose. The sender selects a random value K_n as the last key in the key chain and repeatedly performs a pseudo random function F to compute all the other keys: $K_i = F(K_{i+1}), 0 \leq i \leq n-1$, where the secret key K_i (except for K_0) is assigned to the i-th time interval. Because of the one-way property of the pseudo random function, given K_j in the key chain, anybody can compute all the previous keys $K_i, 0 \leq i \leq j$, but nobody can compute any of the later ones $K_i, j+1 \leq i \leq n$. Thus, with the knowledge of the initial key K_0, which is called the *commitment* of the key chain, a receiver can authenticate any key in the key chain by merely performing pseudo random function operations. When a broadcast message is available in the i-th time interval, the sender generates a MAC for this message with a key derived from K_i, broadcasts this message along with its MAC, and discloses the key K_{i-d} for time interval I_{i-d} in the broadcast message (where d is the disclosure lag of the authentication keys).

Each key in the key chain will be disclosed after some delay. As a result, the attacker can forge a broadcast packet by using the disclosed key. μTESLA uses a security condition to prevent such situations. When a receiver receives an incoming broadcast packet in time interval I_i, it checks the security condition $\lfloor (T_c + \Delta - T_1)/T_{int} \rfloor < i + d - 1$, where T_c is the local time when the packet is received, T_1 is the start time of the time interval 1, T_{int} is the duration of each time interval, and Δ is the maximum clock difference between the sender and itself. If the security condition is satisfied, i.e., the sender has not disclosed the key K_i yet, the receiver accepts this packet. Otherwise, the receiver simply drops it. When the receiver receives the disclosed key K_i, it can authenticate it with a previously received key K_j by checking whether $K_j = F^{i-j}(K_i)$, and then authenticate the buffered packets that were sent during time interval I_i.

A multi-level μTESLA technique is proposed to extend the capabilities of μTESLA [3, 4]. The basic idea is to construct a multi-level μTESLA structure, where any higher-level μTESLA instance is only used to authenticate the commitments of its immediate lower-level ones and the lowest level μTESLA instances are actually used to authenticate the data packets. This extension enables the original μTESA to cover a long time period and support a large number of receivers. For more detailed discussion on the multi-level μTESLA technique, please refer to [3, 4].

2.2. The Basic Approach

The essential problem in scaling up μTESLA is how to distribute and authenticate the parameters of μTESLA instances, including the key chain commitments, starting time, duration of each time interval, etc. The multi-level μTESLA technique uses higher-level μTESLA instances to authenticate the parameters of lower-level ones, and thus inherits the authentication delay introduced by μTESLA during the distribution of those parameters [3, 4]. The consequence of such authentication delay is that an attacker can launch DOS attacks to disrupt the distribution of initial μTESLA parameters. Moreover, they cannot handle a large number of senders.

Note that in the μTESLA technique, a receiver only needs to buffer the data packets received during d time intervals, since they can authenticate any packet in I_i if K_{i+d} is disclosed. Due to the low bandwidth communication in sensor networks, the number of data packets buffered during d time intervals is usually small. Thus, in this paper, we only focus on the DOS attacks that target at disrupting the distribution of initial μTESLA parameters.

In this section, we propose to authenticate and distribute these μTESLA parameters using a Merkle hash tree [7]. This method removes the authentication delay as well as the vulnerability to DOS attacks during the distribution of μTESLA parameters, and at the same time allows a large number of senders.

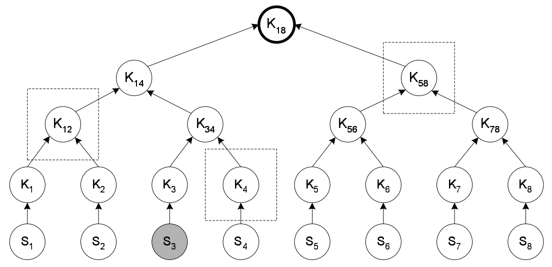


Figure 1. Example of a parameter distribution tree

Assume a sensor network application requires m μTESLA instances, which may be used by different senders during different periods of time. For convenience, assume $m = 2^k$, where k is an integer. Before deployment, the central server pre-computes m μTESLA instances, each of which is assigned a unique, integer-valued ID between 1 and m. For the sake of presentation, denote the parameters (i.e., the key chain commitment, starting time, duration of each μTESLA interval, etc.) of the i-th μTESLA instance as S_i. Suppose the central server has a hash function H. The central server then computes $K_i = H(S_i)$ for all $i \in \{1, ..., m\}$, and constructs a Merkle tree [7] using $\{K_1, ..., K_m\}$ as leaf nodes. Specifically, $K_1, ..., K_m$ are arranged as leaf nodes of a full binary tree, and each non-leaf node is computed by applying H to the concatenation of its two children nodes. We refer to such a Merkle tree as a *parameter distribution tree* of parameters $\{S_1, ..., S_m\}$. Figure 1 shows a parameter distribution tree for eight μTESLA instances, where $K_1 = H(S_1)$, $K_{12} = H(K_1||K_2)$, $K_{14} = H(K_{12}||K_{34})$, etc.

The central server also constructs a *parameter certificate* for each μTESLA instance. The certificate for the i-th μTESLA instance consists of the set S_i of parameters and the values corresponding to the siblings of the nodes on the path from the i-th leaf node to the root in the parameter distribution tree. For example, the parameter certificate for the 3rd μTESLA instance in Figure 1 is $ParaCert_3 = \{S_3, K_4, K_{12}, K_{58}\}$. For each sender that will use a given μTESLA instance, the central server distributes the μTESLA key chain (or equivalently, the random number used to generate the key chain) and the corresponding parameter certificate to the node. The central server also pre-distributes the root of the parameter distribution tree (e.g., K_{18} in Figure 1) to regular sensor nodes, which are potentially receivers of broadcast messages.

When a sender needs to establish an authenticated broadcast channel using the i-th μTESLA instance (during a predetermined period of time), it broadcasts a message containing the parameter certificate $ParaCert_i$. Each receiver can immediately authenticate it with the pre-distributed root of the parameter distribution tree. For example, if $ParaCert_3 = \{S_3, K_4, K_{12}, K_{58}\}$ is used, a receiver can immediately authenticate it by verifying whether $H(H(K_{12}||H(H(S_3)||K_4))||K_{58})$ equals the pre-distributed root value K_{18}. As a result, all the receivers can get the authenticated parameters of this μTESLA instance, and the sender may use it for broadcast authentication.

Security: According to the analysis in [8, 2], an attacker is not able to forge any message from any sender without compromising the sender itself. However, the attacker may launch DOS attacks against the distribution of parameters for μTESLA instances. Fortunately, the parameter certificates in our technique can be authenticated immediately and are immune to the DOS attacks. When a few senders are compromised, additional techniques are required to remove these compromised senders. This will be addressed in Section 2.5.

Overhead: In this approach, each sensor node (as a receiver) only needs to store one hash value, and remember the parameters for those senders that it may communicate with. This is particularly helpful for those applications where a node only needs to communicate with a few senders or there are only a few senders staying in the network at one time.

Each sender needs to store a parameter certificate, the key chain, and other parameters (e.g., starting time) for each instance it has. To establish an authenticated broadcast channel with nodes using an instance j, a sender only needs to broadcast the corresponding pre-distributed parameter certificate, which consists of $\lceil \log m \rceil$ hash values and the parameter set S_j. This is practical, since such distribution only needs to be done once for each instance. After receiving this parameter certificate, a sensor node only needs $1 + \lceil \log m \rceil$ hash functions to verify the related parameters.

Comparison: Compared with the multi-level μTESLA schemes [3, 4], the most significant gain of the proposed approach is the removal of the authentication delay in distributing the μTESLA parameters. The multi-level μTESLA schemes are subject to DOS attacks against the distribution of μTESLA parameters because of the authentication delay. Specifically, receivers cannot authenticate parameter distribution messages immediately after receiving them, and thus have to buffer such messages. An attacker may send a large amount of bogus messages to consume receivers' buffers and thus prevent the receiver from saving the authentic message. To mitigate or defeat such DOS attacks, the multi-level μTESLA schemes either use duplicated copies of distribution messages along with a multi-buffer, random selection strategy, or require substantial pre-computation at the sender. In contrast, the proposed approach does not have these problems. With the proposed approach, senders may still duplicate parameter distribution messages to deal with communication failures. However, unlike multi-level μTESLA schemes, a sender does not have to compete with malicious attackers, since it can immediately authenticate the parameter distribution message instead of keeping it in the buffer for future authentication. In other words, with the proposed approach, it is sufficient for a receiver to receive one copy of each parameter distribution message.

In general, our approach allows late binding of μTESLA instances with senders. For example, the central server may reserve some μTESLA instances during deployment time and distribute them to mobile sinks as needed during the operation of the sensor networks. This allows us to add new senders dynamically by simply generating enough number of instances at the central server for later joined senders. Thus, in our later discussion, we will not discuss how to add new senders.

There are multiple ways to arrange senders, μTESLA instances, and their parameters in a parameter distribution tree. Different ways may have different properties. Next we investigate one specific scheme, which has some additional attractive properties. We consider other options as future work.

2.3. A Scheme for Long-Lived Senders

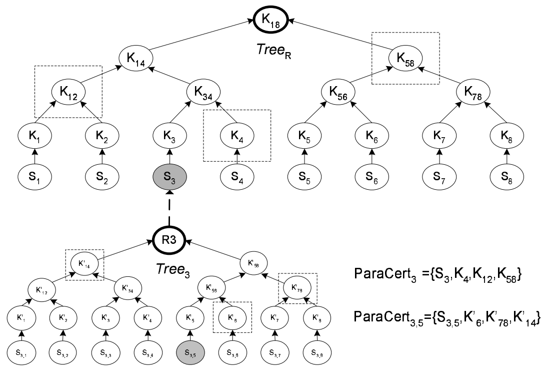
In the following, we present a special instantiation of the basic approach when there are up to m senders in the network and n_j μTESLA instances for each sender j. The purpose is to improve the parameter distribution for those senders that may stay in the network for a long period of time. The protocol can be divided into two phases: *pre-distribution* and *establishment of authenticated broadcast channel*.

Pre-Distribution: The central server first divides the (long) lifetime of each sender into n_j time intervals such that the duration of each time interval (e.g., 1 hour) is suitable for running a μTESLA instance on a sender and sensor nodes efficiently. For convenience, we denote such a time interval as a *(μTESLA) instance interval*, or simply an *instance interval*. When $n_j = 1$ for all $j \in \{1, ..., m\}$, the long-lived version becomes the basic scheme. (Note that each instance interval should be partitioned into smaller time intervals (e.g., 500ms intervals) to run the μTESLA protocol (See Section 2.1).)

For sender j, the central server generates one μTESLA instance for each instance interval. The corresponding key chains are linked together by pseudo random functions. Specifically, the central server generates the last key of the n_j-th μTESLA key chain randomly; for the i-th μTESLA key chain, the central server generates the last key by performing a pseudo random function F' on the first key (the key next to the commitment) of the $(i+1)$-th μTESLA key chain. Let $S_{j,i}$ denote the parameters (e.g., key chain commitment, starting time) of the i-th μTESLA instance for sender j. The parameters (such as the duration of each μTESLA interval) that can be pre-determined do not need to be included in $S_{j,i}$.

For each sender j, the central server generates a parameter distribution tree $Tree_j$ from $\{S_{j,1}, \cdots, S_{j,n_j}\}$. This tree is used to distribute the parameters of different μTESLA instances for sender j. Let $ParaCert_{j,i}$ denote the parameter certificate for $S_{j,i}$ in $Tree_j$. Assume R_j is the root of $Tree_j$. The central server then generates the parameter distribution tree $Tree_R$ for all senders from $\{S_1, \cdots, S_m\}$, where S_j consists of R_j and parameters not included in $\{S_{j,1}, \cdots, S_{j,n_j}\}$ for sender j. If a parameter (e.g., n_j) is the same for all senders, it can be pre-distributed to all sensor nodes before the deployment of sensor networks. Let $ParaCert_j$ denote the parameter certificate for S_j in $Tree_R$. The central server pre-distributes to each sender j the parameter certificate $ParaCert_j$, the n_j μTESLA instances, and the parameter distribution tree $Tree_j$. The central server also pre-distributes the root value of tree $Tree_R$ to each sensor node. Figure 2 shows an example of the above construction.

Establishment of Authenticated Broadcast Channel: To establish an authenticated broadcast channel with sensor nodes, a sender j first broadcasts $ParaCert_j$ to authenticate the root S_j. The authenticity of S_j can be verified as discussed in the basic approach. To distribute the parameters of the i-th μTESLA instance, sender j only needs to $ParaCert_{j,i}$, which can be verified by re-computing the root R_j from $ParaCert_{j,i}$ if S_j, which includes R_j, is already verified. After this, the sender j can authenticate the messages using the i-th μTESLA instance, and the sensor nodes having the authentic parameters can verify the broadcast messages from sender j. To deal with message loss, a sender may broadcast $ParaCert_j$ and $ParaCert_{j,i}$ multiple times.



K_{18}
$Tree_R$
K_{14}
K_{58}
K_{12}
K_{34}
K_{56}
K_{78}
K_1 K_2 K_3 K_4 K_5 K_6 K_7 K_8
S_1 S_2 S_3 S_4 S_5 S_6 S_7 S_8
R3
$Tree_3$
K'_{14} K'_{58}
K'_{12} K'_{34} K'_{56} K'_{78}
K'_1 K'_2 K'_3 K'_4 K'_5 K'_6 K'_7 K'_8
$S_{3,1}$ $S_{3,2}$ $S_{3,3}$ $S_{3,4}$ $S_{3,5}$ $S_{3,6}$ $S_{3,7}$ $S_{3,8}$
$ParaCert_3 = \{S_3, K_4, K_{12}, K_{58}\}$
$ParaCert_{3,5} = \{S_{3,5}, K'_6, K'_{78}, K'_{14}\}$

Figure 2. Example of a parameter distribution tree for long-lived schemes

Security: In the long-lived scheme, different key chains are linked together. This does not sacrifice security. The knowledge of the commitment of later key chain cannot be used to recover the first key in the later key chain and thus cannot be used to recover any key in earlier key chains, since it is computational infeasible to revert the pseudo random function. Moreover, this instantiation is resistant to DOS attacks if each parameter certificate can be delivered in one packet, similar to the basic approach. When it is necessary to send a parameter certificate in multiple packets, there may be DOS attacks. We will investigate this problem in Section 2.4.

Overhead: The above scheme requires each sender j to store n_j key chains, a parameter certificate $ParaCert_j$, and a parameter distribution tree $Tree_j$. This storage overhead is usually affordable at senders, since they may be much more resourceful than the sensor nodes. Similar to the basic scheme, each sensor node only needs to store one hash value, the root of $Tree_R$. To establish an authenticated broadcast channel with sensor nodes, sender j needs to broadcast $ParaCert_j$, which includes $\lceil \log m \rceil$ hash values and parameters in S_j, and $ParaCert_{j,i}$, which includes $\lceil \log n_j \rceil$ hash values and parameters in $S_{j,i}$. A sensor node needs to perform $1 + \lceil \log m \rceil$ hash functions to verify the root R_j of tree $Tree_j$, and $1 + \lceil \log n_j \rceil$ hash functions to verify the corresponding parameters.

Comparison: Compared with the basic approach, this scheme has several benefits. First, the parameters of the i-th μTESLA instance for each sender j is divided into the distribution of the parameters common to all μTESLA instances (i.e., S_j) for the same sender and those specific to each μTESLA instance (i.e., $S_{j,i}$). Thus, the communication overhead can be reduced. Second, this scheme connects different μTESLA key chains together through pseudo random functions, and thus provides two options to verify a disclosed μTESLA key. A sensor node can always verify disclosed keys with an earlier key. This is suitable when there are no long term communication failures or channel jamming attacks, since a sensor node can usually authenticate a disclosed key with a few pseudo random functions. Alternatively, a sensor node can authenticate any disclosed key using the commitment derived from the most recently verified parameter certificate. This is suitable when there are long term communication failures or channel jamming attacks, or for newly deployed nodes. Third, when all μTESLA instances are linked together, a sensor node may use a later key to derive an earlier key to authenticate a buffered message. Such a capability is not available for independent μTESLA instances.

2.4. Distributing Parameter Certificates

As we mentioned earlier, the proposed technique is resistant to the DOS attacks if each parameter certificate is delivered in one packet, since a receiver can authenticate such a certificate immediately upon receiving it. However, due to the low bandwidth and small packet size in sensor networks, a certificate may be too large to be transmitted in a single packet. As a result, it is often necessary to fragment each certificate and deliver it in multiple packets.

A straightforward approach is to simply split those values in a certificate into multiple packets. However, this simple idea suffers from DOS attacks, where an attacker sends a large number of forged certificates and forces a sensor node to perform a lot of computations to identify the right one from those fragments. To deal with this problem, we adopt the idea in [5]. Intuitively, we fragment a parameter certificate in such a way that a sensor node can authenticate each fragment independently instead of trying every combination.

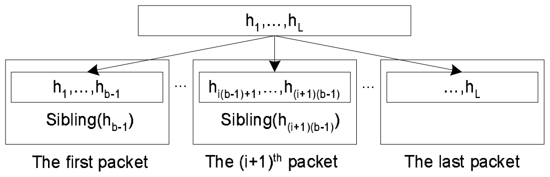


Figure 3. Example of fragmentation

Assume a parameter certificate then consists of L values $\{h_1,\, h_2,\, \cdots, h_L\}$, and each packet can carry b values. As shown in Figure 3, in the first step of fragmentation, we put the first $b-1$ values in the first packet, the second $b-1$ values in the second packet, and so on, until there are no more values left. If the last packet only includes one value, we move it to the previous packet and remove the last packet. The previous packet then becomes the last packet, containing b values. In the second step, we append in every packet other than the last one the sibling (in the parameter distribution tree) of the last value in this packet. By doing this, the first fragment can be authenticated immediately once the sensor node receives an authentic fragment. After authenticating the first fragment, the second fragment can be also authenticated immediately using the values in the first fragment. This process will continue until the sensor node receives all authentic fragments.

For example, in Figure 1, $ParaCert_3$ consists of 4 values, $\{K_{58}, K_{12}, K_4, S_3\}$. Assume each fragment can carry 3 hash values and S_3 consists of 1 key chain commitment. Using the above technique, the first packet includes $\{K_{58}, K_{12}, K_{34}\}$, and the second packet includes K_4, S_3. If a sensor node receives the first fragment, it can authenticate the fragment by verifying whether $H(H(K_{12}|K_{34})|K_{58})$ equals the pre-distributed root value. Once the first fragment is authenticated successfully, the second fragment can be authenticated by verifying if $H(H(S_3)|K_4)$ equals the hash value K_{34}, which is contained in the first fragment.

The above technique can reduce the computation overhead required to authenticate the certificate fragments significantly when there are DOS attacks. In addition, if most of fragments are received in order, there is no need to allocate a large buffer to store these fragments, since most of fragments can be authenticated immediately.

2.5. Revoking μTESLA Instances

In hostile environments, not only sensor nodes but also broadcast senders may be captured and compromised by adversaries. Once a sender is compromised, the attacker can forge any broadcast message using the secrets stored on this sender and convince other sensor nodes to perform unnecessary or malicious operations. Thus, it is necessary to revoke the broadcast authentication capability from compromised senders.

In this paper, we do not consider the process or techniques to detect compromised or captured broadcast senders, but assume such results are given. The detection of compromised or captured senders is in general difficult, but feasible at least in certain scenarios. For example, in battlefields, broadcast senders may be carried by, for example, soldiers or unmanned vehicles. If a soldier or an unmanned vehicle is captured, we need to revoke its broadcast authentication capability.

We propose two approaches to revoke compromised senders. The first one uses a revocation tree to take back the broadcast authentication capability from compromised senders, while the second one employs proactive refreshment to control the broadcast authentication capability of each sender. Revocation of compromised senders requires the central server to be on-line when it broadcasts revocation messages; however, the central server can still remain off-line in other situations.

Revocation Tree: When a sender is detected to have been compromised, the central server broadcasts a revocation message with the IDs of the sender. This message has to be authenticated; otherwise, an attacker may forge such messages to revoke non-compromised senders. We may use another μTESLA instance maintained by the central server to authenticate such message. However, this instance has special functions and may become an attractive target for DOS attacks due to the authentication delay. The following discussion provide an alternative method that does not suffer from DOS attacks or authentication delay.

The main idea of this method is to construct a Merkle tree similar to parameter distribution trees, which is called a *revocation tree*, since its purpose is to revoke broadcast authentication capabilities from compromised senders. The revocation tree is built from sender IDs and random numbers. If the sender ID j and the corresponding random number is disclosed in an authenticated way, sender j is revoked.

Assume there are potentially m senders. For simplicity, we assume $m = 2^k$ for an integer k. The central server generates a random number r_j for each sender with ID j, where $1 \leq j \leq m$. The central server then constructs a Merkle tree where the j-th leaf node is the concatenation of ID j and r_j. We refer to this Merkle tree as the *revocation tree*. The central server finally distributes the root of the revocation tree to all sensor nodes. We assume the central server is physically secure. Protection of the central server is an important but separate issue; we do not address it in this paper.

When a sender j is detected to have been compromised, the central server broadcasts the ID j and the random number r_j. To authenticate these values, the central server has to broadcast the sibling of each node on the path from "$j||r_j$" (i.e., the leaf node for j in the revocation tree) to the root. This is exactly the same as the parameter certificate technique used to authenticate μTESLA parameters. To distinguish from parameter certificate, we refer to the above set of values as a *revocation certificate*, denoted $RevoCert_j$. With $RevoCert_j$, any sensor node can recompute the root hash value, and verify it by checking if it leads to the predistributed root value. If a sensor node gets a positive result from this verification, it puts the corresponding sender into a revocation list, and stops accepting broadcast messages from the sender. To deal with message loss, the distribution of a revocation certificate may be repeated multiple times.

The revocation tree approach cannot guarantee the revocation of all compromised senders in presence of communication failures, though traditional fault tolerant techniques can provide high confidence. However, it guarantees that a non-compromised sender will not be revoked. This is because the revocation of a sender requires a revocation certificate, which is only known to the central server. An attacker cannot forge any revocation certificate without access to the random numbers kept in the leaves of the revocation tree, due to one-way function used to generated the revocation tree [7].

In this approach, each sensor node needs to store an additional hash value, the root of the revocation tree. To revoke a sender, the central server distributes a revocation certificate, which consists of $1 + \lceil \log m \rceil$ values. An overly long revocation certificate can be transmitted in the same way as discussed in the previous subsection. To authenticate the revocation certificate, a sensor node needs to perform $1 + \lceil \log m \rceil$ hash functions.

The revocation tree approach has several limitations. First, due to the unreliable wireless communication and possible malicious attacks (e.g., channel jamming), the revocation messages are not guaranteed to reach every sensor node. As a result, an attacker can convince those sensor nodes that missed the revocation messages to do unnecessary or malicious operations using the revoked μTESLA instances. Second, each sensor node needs to store a revocation list, which introduces additional storage overhead, especially when a large number of senders are revoked.

Note that the above approach can also be used to tell sensor nodes that the corresponding sender has stopped broadcast so that they can erase its parameters to save memory space for other senders.

Proactive Refreshment of Authentication Keys: To deal with the limitations of the revocation tree approach, we present an alternative method to revoke the authentication capability from compromised senders. The basic idea is to distribute a fraction of authentication keys to each sender and have the central server update the keys for each sender when it is necessary. A clear benefit is that if a sender is compromised, the central server only needs to stop distributing new authentication keys to this sender; There is no need to broadcast a revocation message and maintain a revocation list at each sensor node. In addition, this approach guarantees that once compromised senders are detected, they will be revoked from the network after a certain period of time. The authentication keys for each sender can be distributed in a proactive way, since we can predetermine the time when a key will be used.

Specifically, during the pre-distribution phase, the central server distributes the parameter certificates (but not the μTESLA instances) to each sender. For simplicity, we assume the central server gives a μTESLA instance to a sender each time. Before the current μTESLA instance expires, the central server distributes the key used to derive the next μTESLA key chain to the sender through a key distribution message encrypted with a key shared between the central server and the sender, provided that the sender has not been detected to have been compromised. The sender may then generate the next μTESLA key chain accordingly. To increase the probability of successful distribution of authentication keys in presence of communication failures, the central server may send each key distribution message multiple times.

As mentioned earlier, the revocation of a compromised sender is guaranteed (with certain delay) in the proactive refreshment approach when it is detected to have been compromised. However, the broadcast authentication capability of a sender is not guaranteed if there are message losses. A sender may miss all key distribution messages that carry new authentication keys due to unreliable wireless communication and malicious attacks. Thus, a sender may have no keys to authenticate new data packets. Moreover, there may be a long delay between the detection and the revocation of a compromised sender, and the compromised sender may still have keys that can be used to forge broadcast messages.

In the proactive refreshment approach, instead of storing n_j μTESLA instances, a sender j only needs to store a few of them. Thus, the storage overhead is reduced. However, the communication overhead between the central server and the senders is increased, since the central server has to distribute keys to each sender individually. Moreover, the central server has to be on-line more often. There are no additional communication and computation overheads for sensor nodes.

Both of the above approaches have advantages and disadvantages. In practice, these two options may be combined together to provide better performance and security. The revocation certificates from the central server can mitigate the problem of the delay between the detection and the revocation of a compromised sender, while the proactive refreshment technique guarantees the future revocation of a compromised sender if the compromise is detected.

3. Implementation and Evaluation

We have implemented the long-lived version of the proposed techniques on TinyOS [6], and used Nido, the TinyOS simulator, to evaluate the performance. Our evaluation is focused on the broadcast of data packets and the distribution of μTESLA parameters. Since the number of senders does not affect these two aspects, we only consider a single sender in the evaluation.

We compare our techniques with the multi-level μTESLA schemes in [3, 4], where two schemes were proposed: multi-level DOS-tolerant μTESLA and multi-level DOS-resistant μTESLA. Multi-level DOS-resistant μTESLA has as much communication overhead as multi-level DOS-tolerant μTESLA, and will fall back to multi-level DOS-tolerant μTESLA at a receiver when the receiver misses all copies of a parameter distribution message. Thus, we only compare our scheme with the multi-level DOS-tolerant μTESLA, which can be obtained from `http://discovery.csc.ncsu.edu/software/ML-microTESLA`. For convenience, we call the techniques proposed in this paper as the *tree-based* scheme.

We adopt a setting similar to [3, 4]: The μTESLA key disclosure delay is 2 μTESLA time intervals, the duration of each μTESLA time interval is 100 ms, and each μTESLA key chain consists of 600 keys. Thus, the duration of each μTESLA instance is 60 seconds. We assume there are 200 μTESLA instances, which cover up to 200 minutes in time. Each parameter set $S_{j,i}$ only contains a μTESLA key chain commitment. This means that each parameter certificate contains 9 hash values. Assume each hash value, cryptographic key or MAC value is 8 bytes long. The parameter certificate can be delivered with 4 packets, each of which contains a sender ID (2 bytes), a key chain index (2 bytes), a fragment index (1 byte), and three hash values (24 bytes). As a result, the packet payload size is 29 bytes, which is the default maximum payload size in TinyOS [6].

The multi-level μTESLA schemes use Commitment Distribution Messages (*CDM*) to distribute parameters of μTESLA instances. According to the implementation we obtained, each *CDM* message in multi-level DOS-tolerant scheme also contains 29 bytes payload. For convenience, we call a parameter certificate fragment or a *CDM* packet a *parameter distribution packet*. Thus, in our later comparison, both of our scheme and the multi-level μTESLA scheme introduce the same communication overhead when the frequency of parameter distribution packets is the same.

We study and compare the performances of the tree-based technique and the multi-level DOS-tolerant μTESLA scheme in terms of DOS attacks, channel loss rate, and storage and communication overheads. We set the data packet rate from the sender as 100 data packets per minute, and allocate 3 buffers for data packets at each sensor node. The metrics we are interested in here are the *authentication rate*, which is the fraction of authenticated data packets, the *distribution rate*, which is the fraction of successfully distributed parameters, and the *average failure recovery delay*, which is the average number of μTESLA time intervals needed to have the authenticated parameters for the next μTESLA instance after a sensor node loses every authentic parameter distribution message for a given μTESLA instance.

We use a simple strategy to rebroadcast a parameter certificate, where the rebroadcasts of any certificate are non-interleaving, and the fragments of each certificate are broadcast in order. Other strategies are also possible. However, we consider them as possible future work.

To investigate the authentication rate and the distribution rate under DOS attacks and communication failures, we assume the attacker sends 200 forged parameter distribution packets per minute. We also assume the channel loss rate is 0.2. Figure 4(a) illustrates the authentication rate for both schemes as the frequency of parameter distribution packets increases. We assume 20 *CDM* buffers at each receiver for the multi-level DOS-tolerant μTESLA scheme. We can see that the tree-based scheme always has a higher authentication rate the multi-level DOS-tolerant μTESLA scheme. The reason is that in the tree-based scheme, a sensor node is able to authenticate any buffered message once it receives a later disclosed key, since different key chains are linked together. Though in the multi-level DOS-tolerant μTESLA scheme, lower-level μTESLA key chains are also linked to the higher-level ones, a sensor node may have to wait for a long time to recover an authentication key from the higher-level key chain when the corresponding lower-level key chain commitment is lost due to severe DOS attacks or channel losses. During this time period, most of previous buffered data packets are already dropped.

Figure 4(b) shows the authentication rate for both schemes as the number of buffers for parameter distribution packets increases. We assume the sender distributes 20 parameter distribution packets per minute, which introduce the same communication overhead for both schemes. We can see that the multi-level DOS-tolerant μTESLA scheme has to allocate a large buffer to achieve certain authentication rate when there are severe DOS attacks, while the tree-based scheme can achieve higher authentication rate without any additional buffer. The reason is that in the tree-based scheme, a sensor node can verify a parameter certificate immediately and thus there is no need to buffer certificates, while in the multi-level DOS-tolerant μTESLA scheme, a sensor node has to wait for a while before authenticating *CDM* messages.

Figure 5(a) focuses on the communication and storage overhead introduced by both schemes to achieve high distribution rate. It shows that to achieve a desirable distribution rate under severe DOS attacks, the multi-level DOS-tolerant μTESLA scheme requires a large buffer and a high rebroadcast frequency, while the tree-based scheme is more communication and storage efficient. Note that the number of μTESLA key chains supported by the tree-based scheme affects the number of fragments for each certificate and thus affects the distribution rate. Figure 5(b) shows that to achieve high distribution rate, the number of key chains supported by the tree-based scheme increases dramatically when the frequency of parameter distribution packets increases. This means that the tree-based scheme can cover a very long time period by increasing a little communication overhead.

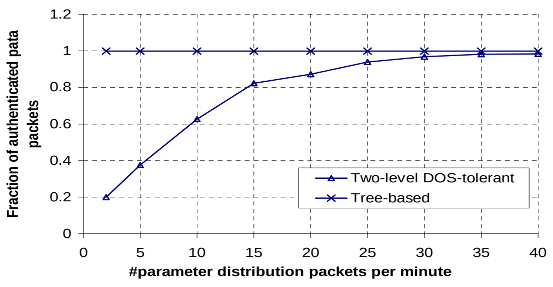
To investigate the average failure recovery delay, we assume the sender distributes 20 parameter distribution packets per minute. (Note that the multi-level DOS-resistant μTESLA scheme has to fall back to the multi-level DOS-tolerant μTESLA scheme if a sensor node loses every authentic copy of a given *CDM* message.)

Figure 6(a) shows the average failure recovery delay for both schemes as the channel loss rate increases. We assume 20 *CDM* buffers for the multi-level μTESLA scheme. We can see that the average failure recovery delay of the tree-based scheme increases with the channel loss rate, while the multi-level μTESLA scheme is not affected when the loss rate is small. However, the recovery delay of the multi-level μTESLA scheme increases rapidly when there are severe DOS attacks. In contrast, the tree-based scheme is not affected by DOS attacks if the attacker does not jam the channel completely. Since the channel loss rate is usually a small value, the tree-based scheme has shorter recovery delay than the multi-level μTESLA scheme in most cases. Figure 6(b) shows the impact of storage overhead on the average failure recovery delay. We assume the channel loss rate is 0.2. The average failure recovery delay of the multi-level μTESLA scheme increase quickly when the number of buffers for parameter distribution packets decreases, while the tree-based scheme has shorter delay and is not affected by the number of buffers for parameter distribution packets.

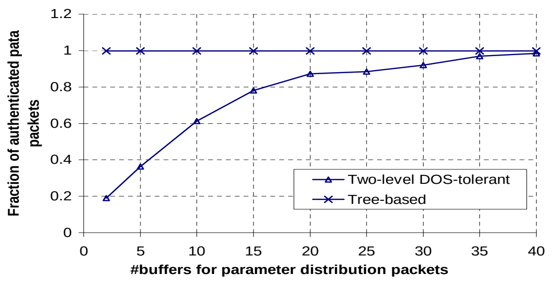
4. Related Work

The technique in this paper is based on μTESLA, which is described in Section 2.1. Other related studies include those discussed in [3, 4, 9, 8, 10]. Perrig et al. proposed to use an earlier key chain to distribute the next key chain commitment [9]. Several multi-level μTESLA schemes was proposed in [3, 4] to distribute the key chain commitments. However, these techniques suffer from DOS attacks during the commitment distribution. Our techniques provide efficient ways to deal with such DOS attacks.

A number of key pre-distribution techniques have been proposed to establish pairwise keys in sensor networks [11, 12, 13, 14, 15]. LEAP is developed to establish individual keys between sensors and a base station, pairwise keys between sensors, cluster keys within a local area, and a group key shared by all nodes [16]. Wood and Stankovic identified a number of DOS attacks in sensor networks [17]. Karlof and Wagner analyzed the vulnerabilities as well as the countermeasures for a number of existing routing protocols [18]. Sastry, Shankar and Wagner proposed a

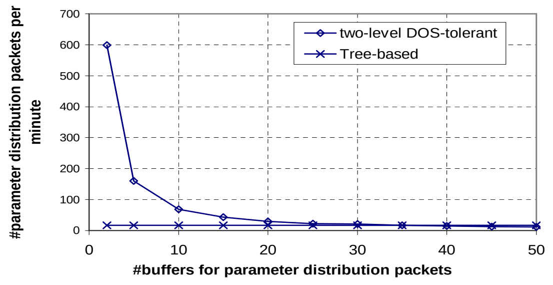


(a) 20 CDM buffers for multi-level μTESLA

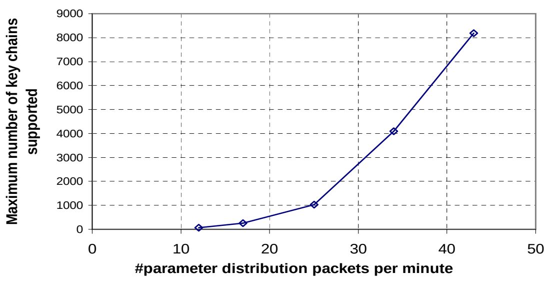


(b) 20 parameter distribution packets per minute

Figure 4. Authentication rate under 0.2 loss rate and 200 forged parameter distribution packet per minute.

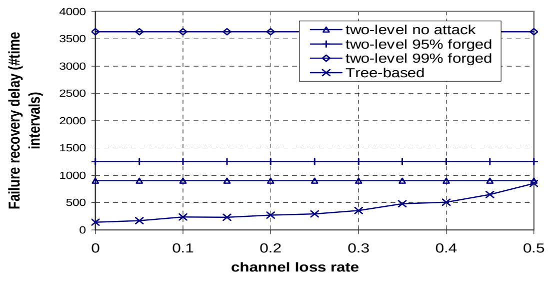


(a) Communication overhead v.s. storage overhead

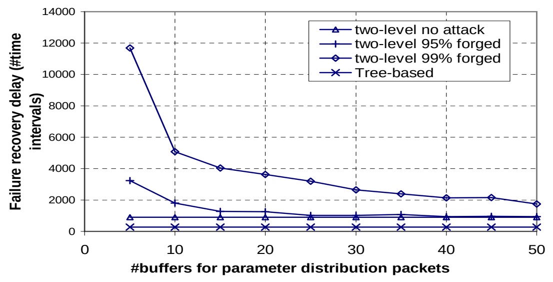


(b) Maximum number of key chains v.s. frequency of distributing parameter distribution packets

Figure 5. Channel loss rate: 0.2; # forged commitment distribution: 200 per minute; distribution rate: 95%.



(a) 20 *CDM* buffers for multi-level μTESLA.



(b) 0.2 channel loss rate

Figure 6. Average failure recovery delay. Assume 20 parameter distribution packet per minute.

location verification technique based on the round trip time (RTT) [19] to detect false location claims. Hu and Evans proposed to use directional antenna to detect worm-hole attacks in wireless Ad Hoc networks [20]. Newsome et al. studied the Sybil attack in sensor networks and developed techniques to defend against this attack [21]. Our proposed techniques can be combined with these techniques to further enhance the security in sensor networks.

Merkle tree [7] has been widely used for authentication purposes. In particular, Hu et al. use Merkle tree to authentication multiple key chains for authentication in routing protocols [22]. Our results differ from theirs in that our techniques are specifically targeted at enhancing μTESLA in sensor networks and that we perform a thorough comparison with the previous approaches.

5. Conclusion and Future Work

This paper identified new challenges in broadcast authentication for wireless sensor networks. Several practical broadcast authentication techniques were developed to support multiple senders, distribute parameters for μTESLA instances, and revoke the broadcast authentication capabilities of compromised senders in wireless sensor networks. Our analysis and experiment show that the proposed techniques are efficient and practical, and have better performance than previous approaches.

Several problems are worth further studying. First, in addition to the long-lived schemes, other alternatives deserve additional research effort. Second, we are also interested in more efficient strategies to distribute the fragments of a certificate. Finally, one limitation of μTESLA protocol is that it requires loosely time synchronization between senders and receivers. Thus, it is interesting to look for new techniques that do not require time synchronization.

Acknowledgment The authors would like to thank the anonymous reviewers for their valuable comments.

References

[1] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: A survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, 2002.

[2] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and D. Tygar, "SPINS: Security protocols for sensor networks," in *Proceedings of Seventh Annual International Conference on Mobile Computing and Networks*, July 2001.

[3] D. Liu and P. Ning, "Efficient distribution of key chain commitments for broadcast authentication in distributed sensor networks," in *Proceedings of the 10th Annual Network and Distributed System Security Symposium (NDSS'03)*, February 2003, pp. 263–276.

[4] D. Liu and P. Ning, "Multi-level μTESLA: Broadcast authentication for distributed sensor networks," *ACM Transactions in Embedded Computing Systems (TECS)*, vol. 3, no. 4, 2004.

[5] C. Karlof, N. Sastry, Y. Li, A. Perrig, and J. Tygar, "Distillation codes and applications to dos resistant multicast authentication"," in *Proc. 11th Network and Distributed Systems Security Symposium (NDSS)*, 2004.

[6] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D.E. Culler, and K. S. J. Pister, "System architecture directions for networked sensors," in *Architectural Support for Programming Languages and Operating Systems*, 2000, pp. 93–104.

[7] R. Merkle, "Protocols for public key cryptosystems," in *Proceedings of the IEEE Symposium on Research in Security and Privacy*, Apr 1980.

[8] A. Perrig, R. Canetti, D. Song, and D. Tygar, "Efficient authentication and signing of multicast streams over lossy channels," in *Proceedings of the 2000 IEEE Symposium on Security and Privacy*, May 2000.

[9] A. Perrig, R. Canetti, Briscoe, J. Tygar, and D. Song, "TESLA: Multicast source authentication transform," IRTF draft, draft-irtf-smug-tesla-00.txt, November 2000.

[10] A. Perrig, R. Canetti, D. Song, and D. Tygar, "Efficient and secure source authentication for multicast," in *Proceedings of Network and Distributed System Security Symposium*, February 2001.

[11] H. Chan, A. Perrig, and D. Song, "Random key predistribution schemes for sensor networks," in *IEEE Symposium on Research in Security and Privacy*, 2003, pp. 197–213.

[12] W. Du, J. Deng, Y. S. Han, S. Chen, and P. Varshney, "A key management scheme for wireless sensor networks using deployment knowledge," in *Proceedings of IEEE INFOCOM'04*, March 2004.

[13] W. Du, J. Deng, Y. S. Han, and P. Varshney, "A pairwise key pre-distribution scheme for wireless sensor networks," in *Proceedings of 10th ACM Conference on Computer and Communications Security (CCS'03)*, October 2003, pp. 42–51.

[14] L. Eschenauer and V. D. Gligor, "A key-management scheme for distributed sensor networks," in *Proceedings of the 9th ACM Conference on Computer and Communications Security*, November 2002, pp. 41–47.

[15] D. Liu and P. Ning, "Establishing pairwise keys in distributed sensor networks," in *Proceedings of 10th ACM Conference on Computer and Communications Security (CCS'03)*, October 2003, pp. 52–61.

[16] S. Zhu, S. Setia, and S. Jajodia, "LEAP: Efficient security mechanisms for large-scale distributed sensor networks," in *Proceedings of 10th ACM Conference on Computer and Communications Security (CCS'03)*, October 2003, pp. 62–72.

[17] A. D. Wood and J. A. Stankovic, "Denial of service in sensor networks," *IEEE Computer*, vol. 35, no. 10, pp. 54–62, 2002.

[18] C. Karlof and D. Wagner, "Secure routing in wireless sensor networks: Attacks and countermeasures," in *Proceedings of 1st IEEE International Workshop on Sensor Network Protocols and Applications*, May 2003.

[19] N. Sastry, U. Shankar, and D. Wagner, "Secure verification of location claims," in *ACM Workshop on Wireless Security*, 2003.

[20] L. Hu and D. Evans, "Using directional antennas to prevent wormhole attacks," in *Proceedings of the 11th Network and Distributed System Security Symposium*, February 2003, pp. 131–141.

[21] J. Newsome, R. Shi, D. Song, and A. Perrig, "The sybil attack in sensor networks: Analysis and defenses," in *Proceedings of IEEE International Conference on Information Processing in Sensor Networks (IPSN 2004)*, Apr 2004.

[22] Y. Hu, A. Perrig, and D. V. Johnson, "Efficient security mechanisms for routing protocols," in *Proceedings of the 10th Annual Network and Distributed System Security Symposium*, February 2003, pp. 57–73.