Efficient Distribution of Key Chain Commitments for Broadcast Authentication in Distributed Sensor Networks

Donggang Liu Peng Ning
Department of Computer Science
North Carolina State University
Raleigh, NC 27695-7534
Emails: dliu@unity.ncsu.edu, ning@csc.ncsu.edu

Abstract

Broadcast authentication is a fundamental security service in distributed sensor networks. A scheme named μTESLA has been proposed for efficient broadcast authentication in such networks. However, μTESLA requires initial distribution of certain information based on unicast between the base station and each sensor node before the actual authentication of broadcast messages. Due to the limited bandwidth in wireless sensor networks, this initial unicast-based distribution severely limits the application of μTESLA in large sensor networks. This paper presents a novel technique to replace the unicast-based initialization with a broadcast-based one. As a result, μTESLA can be used in a sensor network with a large amount of sensors, as long as the message from the base station can reach these sensor nodes. This paper further explores several techniques that improve the performance, the robustness, as well as the security of the proposed method. The resulting protocol satisfies several nice properties, including low overhead, tolerance of message loss, scalability to large networks, and resistance to replay attacks as well as some known Denial of Service (DOS) attacks.

1 Introduction

A distributed sensor network usually consists of one or several computationally powerful nodes called *base stations* and a large amount of inexpensive, low capacity nodes called *sensors* (or *sensor nodes*). The nodes in a distributed sensor network communicate through wireless communication, which is usually limited in bandwidth. Distributed sensor networks have extensive applications in military as well as civilian operations, in which it is necessary to deploy sensor nodes dynamically.

Broadcast authentication is an essential service in distributed sensor networks. Because of the large amount of sensor nodes and the broadcast nature of the communication in distributed sensor networks, it is usually desirable for the base stations to broadcast commands and data to the sensor nodes. In hostile environments (e.g., battle field, anti-terrorists operations), it is necessary to enable the sensor nodes to authenticate the broadcast messages received from the base station.

A protocol named μTESLA [12] has been proposed for broadcast authentication in distributed sensor networks, which is adapted from a stream authentication protocol called TESLA [10]. μTESLA employs a chain of authentication keys linked to each other by a pseudo random function, which is by definition a one way function. Each key in the key chain is the image of the next key under the pseudo random function. The efficiency of μTESLA is based on the fact that once a sensor node has an authenticated key in a key chain, only pseudo random function operations are needed to authenticate the subsequent broadcast messages.

However, due to the low bandwidth of a sensor network and the low computational resources at each sensor node, μTESLA cannot distribute the initial parameters required for broadcast authentication using public key cryptography, as in the original TESLA protocol. Instead, the base station has to unicast the initial parameters to the sensor nodes individually. This feature severely limits the application of μTESLA in large sensor networks. For example, The implementation of μTESLA in [12] has 10Kbps bandwidth and supports 30 bytes messages. To bootstrap 2000 nodes, the base station has to send or receive at least 4000 packets to distribute the initial parameters, which takes at least $\frac{4000\times30\times8}{10240} = 93.75$ seconds even if the channel utilization is perfect. Such a method certainly cannot scale up to very large sensor networks, which may have thousands of nodes.

In this paper, we propose an extension to μTESLA to address the above limitation. The basic idea is to *predetermine* and *broadcast* the initial parameters required by μTESLA instead of unicast-based message transmission. In the simplest form, our extension distributes the μTESLA parameters during the initialization of the sensor nodes (e.g., along with the master key shared between each sensor and the base station). To provide more flexibility, especially to prolong the life time of μTESLA without requiring a very long key chain, we introduce a multi-level key chain scheme, in which the higher-level key chains are used to authenticate the commitments of lower-level ones. To further improve the survivability of the scheme against message loss and Denial of Service (DOS) attacks, we use redundant message transmission and random selection strategies to deal with the messages that distribute key chain commitments. The resulting scheme removes the requirement of unicast-based initial communication between base station and sensor nodes while keeping the nice properties of μTESLA (e.g., tolerance of message loss, resistance to replay attacks). Our implementation and experiments further demonstrate that our scheme can tolerate high channel loss rate and is resistant to certain known DOS attacks to a certain degree.

The rest of this paper is organized as follows. The next section gives a brief overview of μTESLA and its extensions. Section 3 presents the development of our multi-level key chain scheme. Section 4 describes the implementation and experiments with our scheme. Section 5 discusses the related work, and section 6 concludes the paper and points out some future research directions. Appendix A presents the details of the two-level key chain scheme, from which the multi-level key chain is extended.

2 An Overview of μTESLA

Authentication of broadcast messages is an important security issue in wired or wireless networks. Generally, an asymmetric mechanism, such as public key cryptography, is required to authenticate broadcast messages. Otherwise, a malicious receiver can easily forge any packet from the sender. However, due to the high communication, computation and storage overhead of the asymmetric cryptographic mechanisms, it is impractical to implement them in resource constrained sensor networks.

μTESLA introduced asymmetry by delaying the disclosure of symmetric keys [12]. A sender broadcasts a message with a Message Authentication Code (MAC) generated with a secret key K, which will be disclosed after a certain period of time. When a receiver receives this message, if it can ensure that the packet was sent before the key was disclosed, the receiver can buffer this packet and authenticate it when it receives the corresponding disclosed key. To continuously authenticate the broadcast packets, μTESLA divides the time period for broadcasting into multiple time intervals, assigning different keys to different time intervals. All packets broadcasted 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 for the key chain and repeatedly performs a pseudo random function F to compute all the other keys: $K_i = F(K_{i+1}), 0 \le i \le n-1$, where the secret key K_i is assigned to the i^{th} time interval. With the pseudo random function F, given K_j in the key chain, anybody can compute all the previous keys $K_i, 0 \le i \le j$, but nobody can compute any of the later keys $K_i, j+1 \le i \le n$. Thus, with the knowledge of the initial key K_0, the receiver can authenticate any key in the key chain by merely performing pseudo random function operations. When a broadcast message is available in i^{th} time interval, the sender generates MAC for this message with a key derived from K_i and then broadcasts this message along with its MAC and discloses the key K_{i-d} assigned to the time interval I_{i-d}, where d is the disclosure lag of the authentication keys. The sender prefers a long delay in order to make sure that all or most of the receivers can receive its broadcast messages. But, for the receiver, a long delay could result in high storage overhead to buffer the messages.

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 a receiver from accepting any broadcast packet authenticated with a disclosed key. When a receiver receives an incoming broadcast packet in time interval I_i, it checks the security condition $\lfloor (T_c + \Delta - T_0)/T_{int} \rfloor < I_i + d$, where T_c is the local time when the packet is received, T_0 is the start time of the time interval 0, 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.

μTESLA is an extension to TESLA [10]. The only difference between TESLA and μTESLA is in their key commitment distribution schemes. TESLA uses asymmetric cryptography to bootstrap new receivers, which is impractical for current sensor networks due to its high computation and storage overhead. μTESLA depends on symmetric cryptography with the master key shared between the sender and each receiver to bootstrap the new receivers individually. In this scheme, the receiver first sends a request to the sender, and then the sender replies a packet containing the current time T_c (for time synchronization), a key K_i of one way key chain used in a past interval i, the start time T_i of interval i, the duration T_{int} of each time interval and the disclosure lag d.

TESLA was later extended to include an immediate authentication mechanism [11]. The basic idea is to include an image under a pseudo random function of a late message content in an earlier message so that once the earlier message is authenticated, the later message content can be authenticated immediately after it is received. This extension can also be applied to μTESLA protocol in the same way.

3 Efficient Distribution of Key Chain Commitments for μTESLA

The major barrier of using μTESLA in large sensor networks lies in its difficulty to distribute the key chain commitments to a large number of sensor nodes. In other words, the method for bootstrapping new receivers in μTESLA does not scale to a large group of new receivers, though it is okay to bootstrap one or a few. The essential reason for this difficulty is the mismatch between the *unicast*-based distribution of key chain commitments and the authentication of *broadcast* messages.

In this section, we present our method to address the limitation of μTESLA. The basic idea is to *predetermine* and *broadcast* the key chain commitments instead of unicast-based message transmissions. In the following, we present a series of schemes; each later scheme improves over the previous one by addressing some of its limitations.

We assume each broadcast message is from the base station to the sensor nodes. Broadcast messages from a sensor node to the sensor network can be handled as suggested in [12]. That is, the sensor node unicasts the message to the base station, which then broadcasts the message to the other sensor nodes. The messages transmitted in a sensor network may reach the destination directly, or may have to be forwarded by some intermediate nodes; however, we do not distinguish between them in our schemes.

For the sake of presentation, we denote the key chain with commitment K_0 as $\langle K_0 \rangle$ throughout this paper.

3.1 Scheme I: Predetermined Key Chain Commitment

A simple solution to bypass the unicast-based distribution of key chain commitments is to predetermine the commitments, the starting times, and other parameters of key chains to the sensor nodes during the initialization of the sensor nodes, possibly along with the master keys shared between the sensor nodes and the base station. (Unlike the master keys, whose confidentiality and integrity are both important, only the integrity of the key chain commitments needs to be ensured.) As a result, all the sensor nodes have the key chain commitments and other necessary parameters once they are initialized, and are ready to use μTESLA as long as the starting time is passed.

This simple scheme can greatly reduce the overhead involved in distribution of key chain commitments in μTESLA, since unicast-based message transmission is not required any more. However, this simple solution also introduces several problems.

First, a key chain in this scheme can only cover a fixed period of time. To cover a long period of time, we need either a long key chain, or a large interval to divide the time period. If a long key chain is used, the base station will have to precompute and store this key chain. In addition, the receivers will have to perform intensive computation of pseudo random functions if there is a long delay (which covers a large number of intervals) between broadcast messages. If a long interval is used, there will be a long delay before the authentication of a message after it is received, and it requires larger buffer at the sensor nodes. Though the extensions to TESLA [11] can remove this delay and the buffer requirement at the sensor nodes, the messages will have to be buffered longer at the base station.

Second, it is difficult to predict the starting time of a key chain when the sensor nodes are initialized. If the starting time is set too early, the sensor nodes will have to perform a large number of pseudo random function operations in order to authenticate the first broadcast message. In addition, the key chain must be fairly long so that it does not run out before the sensor nodes' life time ends. If the starting time is set too late, messages broadcasted before it cannot be authenticated via μTESLA.

These problems make this simple scheme not a practical one. In the following, we propose several additional techniques so that we not only avoid the problems of unicast-based distribution of key commitment, but also those of the simple scheme.

3.2 Scheme II: Naive Two-Level Key Chains

The essential problem of scheme I lies in the fact that it is impossible to use both a short key chain and short time intervals to cover a long period of time. This conflict can be mitigated by using two levels of key chains.

The two-level key chains consist of a high-level key chain and multiple low-level key chains. The low-level key chains are intended for authenticating broadcast messages, while the high-level key chain is used to distribute and authenticate commitments of the low-level key chains. The high-level key chain uses a long enough interval to divide the time line so that it can cover the life time of a sensor network without having too many keys. The low-level key chains have short enough intervals so that the delay between the receipt of broadcast messages and the verification of the messages is tolerable.

The life time of a sensor network is divided into n (long) intervals of duration Δ_0, denoted as I_1, I_2, ..., and I_n. The high-level key chain has $n+1$ elements K_0, K_1, ..., K_n, which are generated by randomly picking K_n and computing $K_i = F_0(K_{i+1})$ for $i = 0, 1, ..., n-1$, where F_0 is a pseudo random function. The key K_i is associated with each time interval I_i. We denote the starting time of I_i as T_i. Thus, the starting time of the high-level key chain is T_1.

Since the duration of the high-level time intervals is usually very long compared to the network delay and clock discrepancies, we choose to disclose a high-level key K_i used for I_i in the following time interval I_{i+1}. Thus, we use the following security condition to check whether the base station has disclosed the key K_i when a sensor node receives a message authenticated with K_i at time t: $t + \delta_{Max} < T_{i+1}$, where δ_{Max} is the maximum clock discrepancy between the base station and the sensor node.

Each time interval I_i is further divided into m (short) intervals of duration Δ_1, denoted as $I_{i,1}$, $I_{i,2}$, ..., $I_{i,m}$. If needed, the base station generates a low-level key chain for each time interval I_i by randomly picking $K_{i,m}$ and computing $K_{i,j} = F_1(K_{i,j+1})$ for $j = 0, ..., m-1$, where F_1 is a pseudo random function. The key $K_{i,j}$ is intended for authenticating messages broadcasted during the time interval $I_{i,j}$. The starting time of the key chain $\langle K_{i,0} \rangle$ is predetermined as T_i. The disclosure lag for the low-level key chains can be determined in the same way as μTESLA and TESLA [10, 12]. For simplicity, we assume all the low-level key chains use the same disclosure lag d. Further assume that messages broadcasted during $I_{i,j}$ are indexed as (i, j). Thus, the security condition for a message authenticated with $K_{i,j}$ and received at time t is: $i' < i * m + j + d$, where $i' = \lfloor \frac{t - T_1 + \delta_{Max}}{\Delta_1} \rfloor + 1$, and δ_{Max} is the maximum clock discrepancy between the base station and the sensor node.

When sensor nodes are initialized, their clocks are synchronized with the base station. In addition, the starting time T_1, the commitment K_0 of the high-level key chain, the duration Δ_0 of each high-level time interval, the duration Δ_1 of each low-level time interval, the disclosure lag d for the low-level key chains, and the maximum clock discrepancy δ_{Max} between the base station and the sensor nodes throughout the life time of the sensor network are distributed to the sensors.

In order for the sensors to use the low-level key chain $\langle K_{i,0} \rangle$ during the time interval I_i, they must authenticate the commitment $K_{i,0}$. We propose to use the immediate authentication extension to TESLA [11] to achieve this purpose. Specifically, the base station broadcasts a *commitment distribution message*, denoted as CDM_i, during each time interval I_i. This message consists of the commitment $K_{i+1,0}$ of the low-level key chain $\langle K_{i+1,0} \rangle$, the image $H(K_{i+2,0})$ of the commitment $K_{i+2,0}$, where H is a pseudo random function, and the key K_{i-1} in the high-level key chain.

> $Base\ Station \rightarrow Sensors : CDM_i = i|K_{i+11,0}|H(K_{i+2,0})|MAC_{K'_i}(i|K_{i+1,0}|H(K_{i+2,0}))|K_{i-1}$, where "|" denotes message concatenation, and K'_i is derived from K_i with a pseudo random function other than F_0 and F_1.

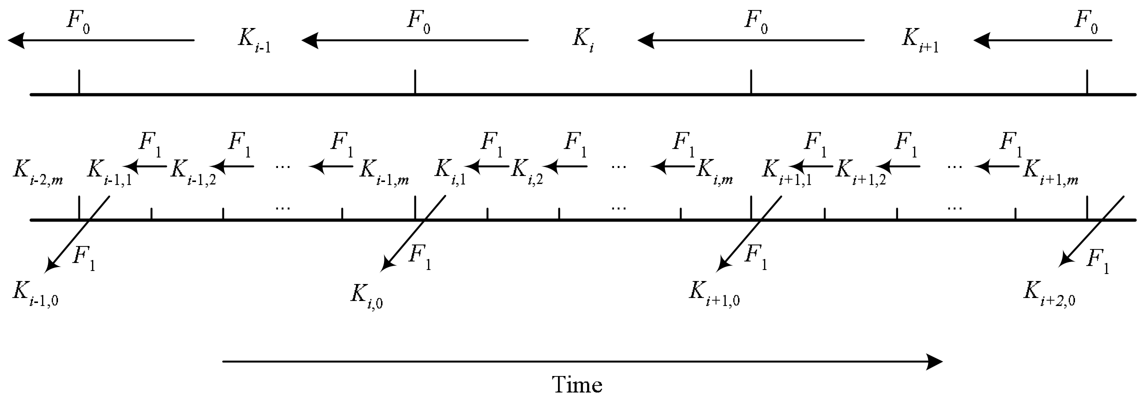


Figure 1: The two levels of key chains in Scheme II. Each key K_i is used for the high-level time interval I_i, and each key $K_{i,j}$ is used for the low-level time interval $I_{i,j}$. F_0 and F_1 are different pseudo random functions. Each commitment $K_{i,0}$ is distributed during the time interval I_{i-1}.

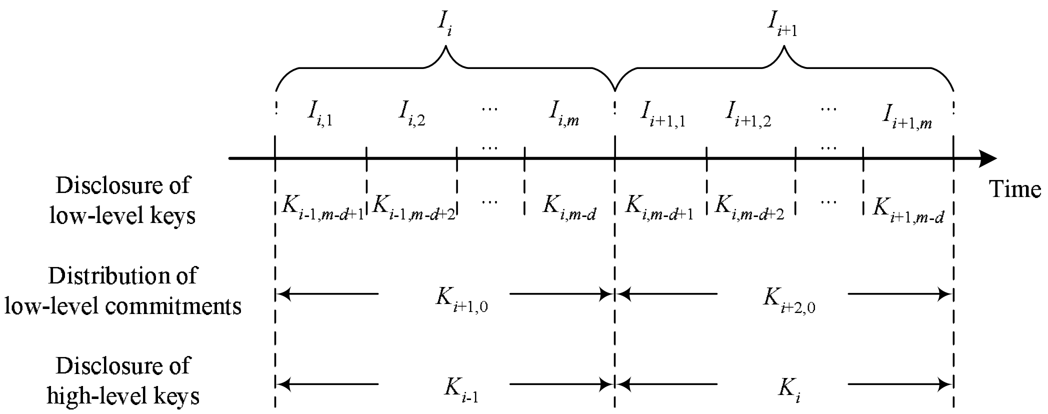


Figure 2: Key disclosure schedule in Scheme II

Thus, to use a low-level key chain $\langle K_{i,0} \rangle$ during I_i, the base station needs to generate the key chain during I_{i-2} and distribute $H(K_{i,0})$ in CDM_{i-2}, and further distribute $K_{i,0}$ in CDM_{i-1}.

Since K_i is disclosed in CDM_{i+1} during the time interval I_{i+1}, each sensor needs to store CDM_i until it receives CDM_{i+1}. Each sensor also stores a key K_j, which is initially K_0. After receiving K_{i-1} in CDM_i, the sensor authenticates it by verifying that $F_1^{i-1-j}(K_{i-1}) = K_j$. Then the sensor replaces the current K_j with K_{i-1}.

Assume a sensor can receive all the commitment distribution messages. The sensor can authenticate the commitment $K_{i,0}$ during I_{i-1}, once it receives CDM_{i-1}. Suppose a sensor has received CDM_{i-2}. Upon receiving CDM_{i-1}, the sensor can authenticate CDM_{i-2} with K_{i-1} disclosed in CDM_{i-1}. Then the sensor can immediately authenticate $K_{i,0}$ by verifying that applying H to $K_{i,0}$ in CDM_{i-1} results in the same $H(K_{i,0})$ included in CDM_{i-2}. As a result, the sensor can authenticate broadcast messages sent by the base station using the μTESLA key chain $\langle K_{i,0} \rangle$ during the time interval I_i.

This scheme uses μTESLA in two different levels. The high-level key chain relies on the initialization phase of the sensor nodes to distribute the key chain commitment, and it only has a single key chain throughout the life time of the sensor network. The low-level key chains depend on the high-level key chain to distribute the commitments. Figure 1 illustrates the two-level key chains, and Figure 2 displays the key disclosure schedule for the keys in these key chains.

The two-level key chains scheme mitigates the problem encountered in scheme I. On the one hand, by having long time intervals, the high-level key chain can cover a long period of time without having a very long key chain. On the other hand, the low-level key chain has short time intervals so that authentication of broadcast messages doesn't have

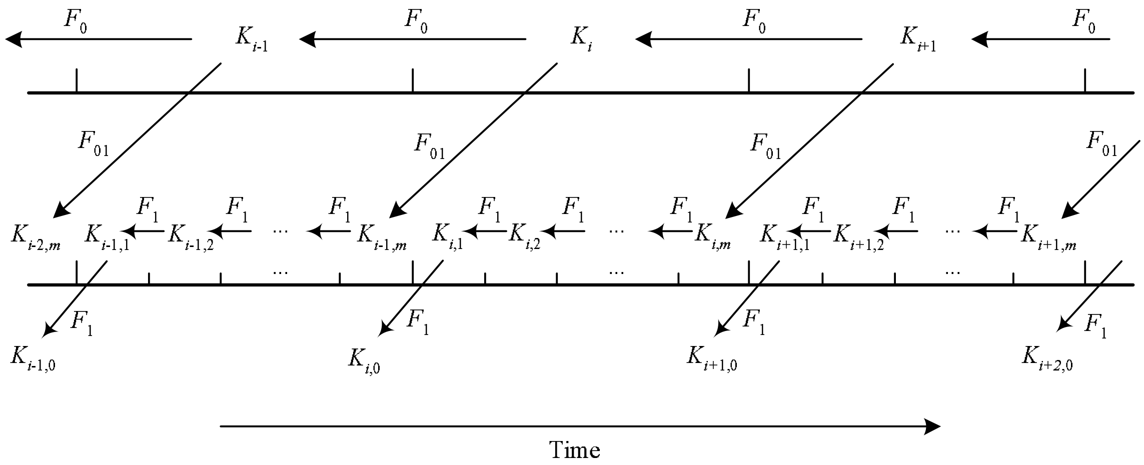


Figure 3: The two levels of key chains in Scheme III. It differs from Figure 1 in that each $K_{i,m}$ is derived from K_{i+1} using an additional pseudo random function F_{01}.

to be delayed too much.

Similar to μTESLA and TESLA, a sensor can detect forged messages by verifying the MAC with the corresponding authentication key once the sensor receives it. In addition, replay attacks can be easily defeated if a sequence number is included in each message.

3.3 Scheme III: Fault Tolerant Two-Level Key Chains

Scheme II does not tolerate message loss as well as μTESLA and TESLA. There are two types of message losses: the loss of normal messages, and the loss of commitment distribution messages. Both may cause problems for scheme II. First, the low-level keys are not entirely chained together. Thus, loss of key disclosure messages for later keys in a low-level key chain cannot be recovered even if the sensor can receive keys in some later low-level key chains. As a result, a sensor may not be able to authenticate a stored message even if it receives some key disclosure messages later. In contrast, with μTESLA a receiver can authenticate a stored message as long as it receives a later key. Second, if CDM_{i-1} does not reach a sensor, the sensor will not be able to use the key chain $\langle K_{i,0} \rangle$ for authentication during the entire time interval I_i, which is usually very long to make the high-level key chain short.

To address the first problem, we propose to further connect the low-level key chains to the high-level one. Specifically, instead of choosing each $K_{i,m}$ randomly, we derive each $K_{i,m}$ from a high-level key K_{i+1}, which is to be used in the next high-level time interval, through another pseudo random function F_{01}. That is, $K_{i,m} = F_{01}(K_{i+1})$. As a result, a sensor can recover any authentication key $K_{i,j}$ as long as it receives a commitment distribution message that discloses $K_{i'}$ with $i' >= i + 1$, even if it does not receive any later low-level key $K_{i,j'}$ with $j' >= j$. Thus, the first problem can be resolved. Figure 3 illustrates this idea.

The second problem does not have an ultimate solution; if the base station cannot reach a sensor at all during a time interval I_i, CDM_i will not be delivered to the sensor. However, the impact of temporary communication failures can be reduced by standard fault tolerant approaches.

To mitigate the second problem, we propose to have the base station periodically broadcast the commitment distribution message during each time interval. Assuming that the frequency of this broadcast is F, each commitment distribution message is therefore broadcasted $F \times \Delta_0$ times. To simplify the analysis, we assume the probability that a receiver cannot receive a broadcast of a commitment distribution message is p_f. Thus, the probability that a receiver cannot receive any copy of the commitment distribution message is reduced to $p_f^{F \times \Delta_0}$.

Note that even if a sensor cannot receive any commitment distribution message during a time interval I_i, it still has the opportunity to authenticate broadcast messages in time intervals later than I_{i+1}. Not having the commitment distribution message in time interval I_i only prevents a sensor from authenticating broadcast messages during I_{i+1}.

As long as the sensor gets a commitment distribution message, it can derive all the low-level keys in the previous time intervals.

By periodically broadcasting commitment distribution messages, scheme III introduces more overhead than scheme II. Let's consider the overhead on the base station, the sensors, and the communication channel, respectively. Compared with Scheme II, this scheme increases the overhead of the base station by $F \times \Delta_0$ times. Base stations in a sensor network are usually much more powerful than the sensor nodes. Thus, the increased overhead on base stations may not be a big problem as long as $F \times \Delta_0$ is reasonable.

The sensors are affected much less than the base station in a benign environment, since each sensor only needs to process one commitment distribution message for each time interval. Thus, the sensors have roughly the same overhead as in scheme II. However, we will show that a sensor has to take a different strategy in a hostile environment in which there are DOS attacks. We will delay the discussion of sensors' overhead until we introduce our counter measures.

The overhead on the communication channel is increased by $F \times \Delta_0$ times, since the commitment distribution message for each time interval is repeated $F \times \Delta_0$ times. Assume the probability that a receiver cannot receiver a commitment distribution message is $p_f = 1/2$ and $F \times \Delta_0 = 10$. Under our simplified assumption, the probability that the receiver cannot receive any of the 10 commitment distribution messages is $p_f^{F \times \Delta_0} < 0.1\%$. Further assume that Δ_0 is 1 minutes, which is quite short as the interval length for the high-level key chain. Thus, there is one commitment distribution message per 6 seconds. Assume the bandwidth is 10K bps and each commitment distribution message is 41 bytes = 328 bits, as in our experiments (Section 4). Then the relative communication overhead is $\frac{328}{10240 \times 6} = 0.53\%$. Therefore, scheme III introduces very reasonable communication overhead in typical sensor networks.

3.4 Scheme IV: (Final) Two-Level Key Chains

In scheme III, the usability of a low-level key chain depends on the authentication of the key chain commitment contained in the corresponding commitment distribution message. A sensor cannot use the low-level key chain $\langle K_{i,0} \rangle$ for authentication before it can authenticate $K_{j,0}$ with $j \geq i$ distributed in CDM_{j-1}. This makes the commitment distribution messages attractive targets for attackers. An attacker may disrupt the distribution of commitment distribution messages, and thus prevent the sensors from authenticating broadcast messages during the corresponding high-level time intervals. Although the high-level key chain and the low-level ones are chained together, and such sensors may store the broadcast messages and authenticate them once they receive a later commitment distribution message, the delay between the receipt and the authentication of the messages may introduce a problem: Indeed, an attacker may send a large amount of forged messages to exhaust the sensors' buffer before they can authenticate the buffered messages, and force them to drop some authentic messages.

The simplest way for an attacker to disrupt the commitment distribution messages is to jam the communication channel. We may have to resort to techniques such as frequency hopping if the attacker completely jam the communication channel. This is out of the scope of this paper. The attacker may also jam the communication channel only when the commitment distribution messages are being transmitted. If the attacker can predict the schedule of such messages, it would be much easier for the attacker to disrupt such message transmissions. Thus, the base station needs to send the commitment distribution messages randomly or in a pseudo random manner that cannot be predicted by an attacker that is unaware of the random seed. For simplicity, we assume that the base station sends the commitment distribution messages randomly.

An attacker may forge commitment distribution message to confuse the sensors. If a sensor does not have a copy of the actual CDM_i, it will not be able to get the correct $K_{i+1,0}$, and cannot use the low-level key chain $\langle K_{i+1,0} \rangle$ during the time interval I_{i+1}. Although the immediate authentication extension to TESLA proposed in [11] is intended to deter a similar DOS attack, it cannot defeat the above attack completely.

Consider a commitment distribution message: $CDM_i = i|K_{i+1,0}|H(K_{i+2,0})|MAC_{K'_i}(i|K_{i+1,0}|H(K_{i+2,0}))|K_{i-1}$. Once seeing such a message, the attacker learns K_{i-1} and $K_{i+1,0}$. Then the attacker can replace the actual $H(K_{i+2,0})$ with $H(K'_{i+2,0})$, and forge another message: $CDM'_i = i|K_{i+1,0}|H(K'_{i+2})|MAC_{K'_i}(i|K_{i+1,0}|H(K_{i+2,0}))|K_{i-1}$. Assume a sensor has an authentic copy of CDM_{i-1}. The sensor can verify K_{i-1} and $K_{i+1,0}$ with K_{i-2} and $H(K_{i+1,0})$, respectively, since both K_{i-2} and $H(K_{i+1,0})$ are included in CDM_{i-1}. However, the sensor has no way to verify the authenticity of $H(K'_{i+2})$. If the sensor does not save an authentic copy of CDM_i that contains the correct $H(K_{i+2,0})$, it cannot authenticate $K_{i+2,0}$ in CDM_{i+1} during the time interval I_{i+1}. If the sensor node further misses the chance to get a copy of CDM_{i+1} that contains the authentic $K_{i+2,0}$, then it cannot use the key chain $\langle K_{i+2,0}\rangle$ during the time interval I_{i+2}.

One possible counter measure is to distribute each $K_{i,0}$ in some earlier time intervals than I_{i-1}. The benefit is that before the time interval I_i, a sensor that has received the corresponding commitment distribution message can authenticate $K_{i,0}$ even if it doesn't have $H(K_{i,0})$. However, this doesn't solve all the problem. If a sensor doesn't have an authentic copy of the commitment distribution message, it can never get the correct $K_{i,0}$. To take advantage of this, an attacker can simply forge commitment distribution messages as discussed earlier.

We propose a random selection method to improve the reliable broadcast of the commitment distribution messages. For the CDM_i messages received during each time interval I_i, each sensor first tries to discard as many forged messages as possible. There are two ways for a sensor to identify forged CDM_i message during I_i. First, the sensor can verify if $F_0^{i-1-j}(K_{i-1}) = K_j$, where K_{i-1} is the high-level key disclosed in CDM_i and K_j is a previously disclosed high-level key. Messages that fail this test are certainly forged and should be discarded. Second, if a CDM_i passes the first test, the sensor uses the key K_{i-1} disclosed in CDM_i to authenticate a CDM_{i-1} message it has received during I_{i-1}. If the sensor can authenticate CDM_{i-1}, it can further authenticate $K_{i+1,0}$ in CDM_i with $H(K_{i+1,0})$ contained in CDM_{i-1}. The sensor can discard the CDM_i message if it fails this test.

These two tests can filter out some forged messages; however, they do not rule out all forged messages, as discussed earlier. To further improve the possibility that the sensor has an authentic CDM_i message, the base station uses a random selection method to store the CDM_i messages that pass the above two tests. Our goal is to make the DOS attack so difficult that the attacker would rather use constant signal jamming instead to attack the sensor network. Some of the strategies are also applicable to the low-level key chains as well as the (extended) TESLA and μTESLA protocols.

Without loss of generality, we assume that each copy of CDM_i has been weakly authenticated in the time interval I_i by using the aforementioned two tests.

Single Buffer Random Selection

Let us first look at a simple strategy: *single buffer random selection*. Assume that each sensor node only has one buffer for the commitment distribution message broadcasted in each time interval. In a time interval I_i, each sensor node randomly selects one message from all the copies of CDM_i. The key issue here is to make sure all copies of CDM_i have equal probability to be selected. Otherwise, an attacker who knows the protocol may take advantage of the unequal probabilities and make a forged commitment distribution message be selected.

To achieve this goal, for the kth copy of CDM_i a sensor node receives during the time interval I_i, the sensor node saves it in the buffer with probability $\frac{1}{k}$. Thus, a sensor node will save the first copy of CDM_i in the buffer, substitute the second copy for the buffer with probability $1/2$, substitute the third copy for the buffer with probability $1/3$, and so on. It is easy to verify that if a sensor node receives n copies of CDM_i, all copies have the same probability $1/n$ to be kept in the buffer.

The probability that a sensor node has an authentic copy of CDM_i can be estimated as $P(CDM_i) = 1 - p$, where $p = \frac{\#forged\ copies}{\#total\ copies}$. To maximize his attack, an attacker has to send as many forged copies as possible.

Multiple Buffer Random Selection

The single buffer random selection can be easily improved by having some additional buffers for the commitment distribution messages. Assume there are m buffers. During each time interval I_i, a sensor node can save the first m copies of CDM_i. For the kth copy with $k > m$, the sensor node keeps it with probability $\frac{m}{k}$. If a copy is to be kept, the sensor node randomly selects one of the m buffers and replaces the corresponding copy. It is easy to verify that if a sensor node receives n copies of CDM_i, all copies have the same probability $\frac{m}{n}$ to be kept in one of the buffers.

During the time interval I_{i+1}, the sensor node can verify if it has an authentic copy of CDM_i once it receives and weakly authenticates a copy of CDM_{i+1}. Specifically, the sensor node uses the key K_i disclosed in CDM_{i+1} to verify the MAC of the buffered copies of CDM_i. Once it finds an authentic copy, the sensor node can discard all the other buffers. Note that if this happens, the sensor node can authenticate the content of CDM_{i+1} immediately.

If the sensor node cannot find an authentic copy of CDM_i after the above verification, it can conclude that all buffered copies of CDM_i are forged and discard all of them. The sensor node then needs to repeat the random selection process for the copies of CDM_{i+1}. Thus, a sensor node needs at most $m + 1$ buffers for commitment distribution messages with this strategy: m buffers for copies of CDM_i, and one buffer for the first weakly authenticated copy of

CDM_{i+1}.

With m buffer random selection strategy, the probability that a sensor node has an authentic copy of CDM_i can be estimated as $P(CDM_i) = 1 - p^m$, where $p = \frac{\#forged\ copies}{\#total\ copies}$.

3.5 Scheme V: Multi-Level Key Chains

Scheme IV can be easily extended to an m-level key chain scheme. The m-level key chains are arranged from level 0 to level $m-1$ from top down. The keys in the $(m-1)$-level key chains are used for authenticating data packets. Each higher-level key chain is used to distributed the commitments of the immediately lower-level key chains. Only the last key of the top-level (level 0) key chain needs to be selected randomly; all the other keys in the top-level key chain can be generated from this key, and all the key chains in level i, $1 < i \leq m-1$ are generated from the keys in level $i-1$, in the same way that the low-level key chains are generated from the high-level keys in the two-level key chain scheme. Each higher-level key chain is responsible for broadcasting commitment distribution messages to distribute the commitments for the immediately lower-level key chains. For security concern, we need a family of pseudo random functions. The pseudo random function for each level and between adjacent levels should be different from each other. Such a family of pseudo random functions has been proposed in [10]. Similar to scheme IV, we also use multiple buffer random selection mechanism for the buffering of CDM packets.

The benefit of having multi-level key chains is that it requires less number of keys in each key chain, or equivalently, shorter duration at each key chain level, compared with the two-level key chain scheme. As a result, scheme V can scale up to long period of time.

Compared with scheme IV, the multi-level key chain scheme is not more vulnerable to the DOS attacks. The success of the DOS attacks depends on percentage of forged CDM messages and the buffer capacity in sensor node. As long as the base station maintains a certain authentic CDM message rate, scheme V will not have higher percentage of forged CDM messages than scheme IV. The base station can further piggy-back the CDM messages for different levels of key chains so as to reduce the communication cost. Nevertheless, having more levels of key chains does increase the overhead at both the base station and the sensor nodes. Though devoting more resources for multi-level key chains may not be a problem for the base station, the resource constrained sensor nodes have to maintain more buffers for the key chain commitments as well as CDM messages at different levels. In addition, the more levels we have, the more bandwidth is required to transmit the CDM messages. Thus, we want as few levels as possible to cover the life time of a sensor network.

4 Experiments

We have done a proof-of-concept implementation of the multi-level key chain scheme. (Note that the two-level key chain scheme is a special case of the multi-level one.) To save development time, we implemented the scheme in Java on the basis of a broadcast channel emulated with IP multicast. The implementation involves a base station and multiple sensor nodes. The broadcast channel has a fixed bandwidth shared by all the components. This is implemented by having all the components check the channel availability before they send a message. The broadcast channel is lossy with a message loss rate r_l. This is simulated by having each sensor node drop the received messages with probability r_l. Following [12], we implemented pseudo random functions with a Message Authentication Code (MAC), which was implemented using the CBC-MAC [16] with RC5 as the block cipher [13]. Our experiments use RC5 with 32 bit words, 12 rounds, and 8 byte keys.

To further study the performance of the scheme in presence of attacks, we also implemented an attacker component, which listens to the CDM messages broadcasted by the base station and inserts forged CDM messages into the broadcast channel to disrupt the broadcast authentication. We assume that the attacker is intelligent in that it uses every piece of authentic information that a sensor node can determine in the forged messages. That is, it only modifies the image $H(K_{i+2})$ in a CDM_i message, since any other modification can be detected by a sensor node immediately. There are other attacks against the scheme. Since they are either defeatable by the scheme (e.g., reply attacks, modification of data packets), or not specific to our extension (e.g., DOS attacks against the data packets), we do not consider them in our implementation.

We have performed a series of experiments to evaluate the performance of the scheme when there are packet loss

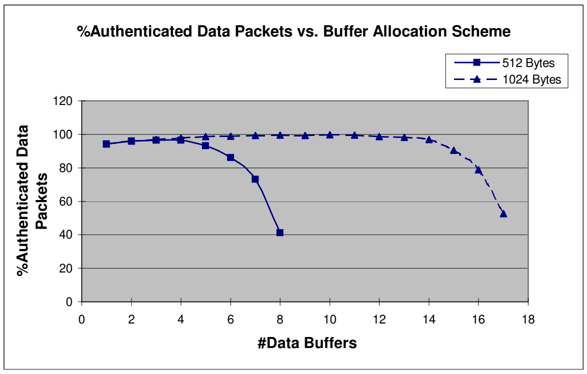


Figure 4: The performance with different buffer allocation scheme for total memory size of 512 and 1024 bytes. Assume 95% of CDM packets are forged.

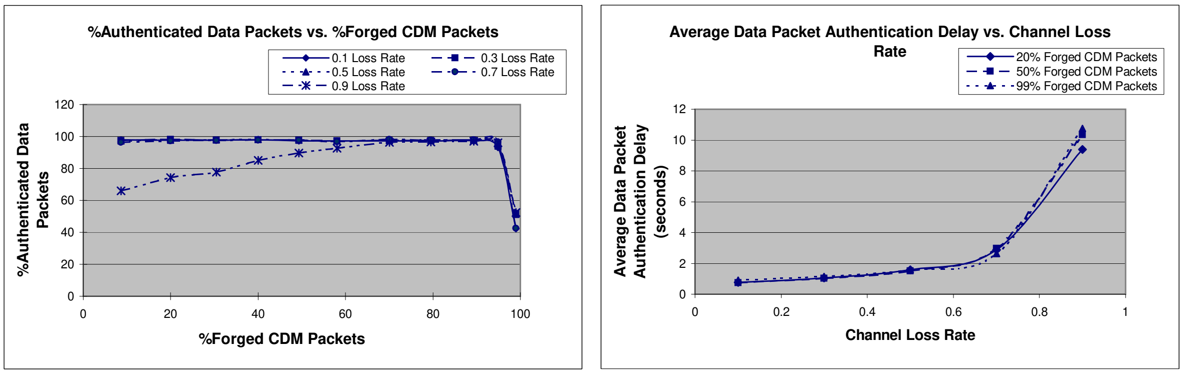
and DOS attacks against CDM messages. The focus of the evaluation is on the overall effectiveness of the proposed techniques (e.g., multi-buffer random selection) in tolerating packet loss and DOS attacks, and the impact of different choices of certain parameters (e.g., buffer size, percentage of forged CDM packets).

To concentrate on the design decisions we made in our schemes, we fix the following parameters in all the experiments. We assume the bandwidth of the broadcast channel is 10Kbps, according to [12]. We only performed the experiments with the two-level key chain scheme, since multi-level key chain scheme is a direct extension to two-level key chain, and its only purpose is to scale up to long period of time. We assume the duration of each low-level time interval is 100 ms, and each low-level key chain consists of 600 keys. Thus, the duration of each time interval for the high-level key chain is 60 seconds. We put 200 keys in the high-level key chain, which covers up to 200 minutes in time. We also set the data packets rate at base station to 100 data packets per minute. Our analysis and experiments indicate that the number of high-level keys does not have an obvious impact on the performance measures. Nevertheless, the life time of the two-level key chains can be extended by having more keys in the high-level key chain or another higher level of key chain. Since our purpose is to study the performance of the scheme w.r.t. to packet loss and DOS attacks, we did not do so in our evaluation.

The performance of scheme is evaluated with the following metrics: average percentage of authenticated data packets (i.e., $\frac{\#authenticated\ data\ packets}{\#received\ data\ packets}$ averaged over the sensor nodes) and average data packet authentication delay (i.e., the average time between the receipt and the authentication of a data packet). In these experiments, we focused on the impact of the following parameters on these performance metrics: sensor node's buffer size for data and CDM messages, percentage of forged CDM packets and the packet loss rate.

Because of the extremely limited memory available on sensor nodes, the buffer allocation for data packets and CDM messages becomes a major concern when we deploy a real sensor network. We evaluate the performance of different memory allocation schemes with a memory constraint. In our implementation, a data packet consists of 65 bytes, which includes a packet header (1 bytes), an index (8 bytes), data (40 bytes), MAC (8 bytes) and a disclosed key (8 bytes). A CDM_i packet is 41 bytes long, which includes a packet header (1 byte), a level number and an index (8 bytes together), next commitment K_{i+1} (8 bytes), $H(K_{i+2})$ (8 bytes), a MAC (8 bytes), and a disclosed key (8 bytes). However, when a sensor node receives a data packet, it does not need to buffer the header and the disclosed key for future authentication; only the other 56 bytes need to be stored. For CDM packets, all copies of the same CDM message have the same values for the fields other than the image of commitment (i.e., $H(K_{i+2,0})$ in CDM_i), since all forged messages without these values can be filtered out by the weak authentication. As a result, except for the first copy of CDM_i, the only field that needs saving is $H(K_{i+2,0})$. Therefore, if we already has one copy of CDM_i (in which 40 bytes need to be saved), we only need to save the 8 bytes of $H(K_{i+2,0})$ for the other copies. Assuming the totally available memory for data and CDM packets is C bytes, and the sensor node can store up to x data packets, then we can have $y = 1 + \lfloor \frac{C-40-56\times x}{8} \rfloor$ for CDM messages.

Figure 4 shows the performance of different allocation schemes under severe DOS attacks against CDM messages (95% forged CDM packets). In these experiments, we have total memory of 512 bytes or 1K bytes. As in Figure 4, three data buffers (168 bytes) are enough to authenticate over 95% of the received data packets. The figure also shows that if the number of data buffers is too many, having more data buffers does not increase the performance. Instead, it



(a) Percentage of authenticated data packets

(b) Average data packet authentication delay

Figure 5: Experimental results under different channel loss rate and percentage of forged CDM packets. Assuming 3 data packet buffers, 39 CDM buffers and fixed data rate (100 $data\ packets/minute$).

may decrease the performance, since less memory is left for buffering the CDM messages.

To measure the performance under DOS attacks, we assume that each sensor node has 512 bytes for both data and CDM packets. According to the previous result, we allocate 168 bytes for data packets and 344 bytes for CDM packets so that the sensor node can store up to 3 data packets and 39 CDM packets. The experimental results are shown in Figures 5(a) and 5(b). Figure 5(a) shows that our scheme can tolerate DOS attacks to a certain degree; however, when there are extremely severe DOS attacks (over 95% of forged CDM packets), the performance decreases dramatically. This result is reasonable; a sensor node is certainly not able to get an authentic CDM message if all of the CDM messages it receives are forged. Nevertheless, an attacker has to make sure he/she sends much more forged CDM packets than the authentic ones to increase his/her chance of success.

Figure 5(a) also shows that if the base station rebroadcast sufficient CDM messages so that on average, at least one copy of such authentic CDM message can reach sensor node in the corresponding high-level time interval (e.g., when loss rate $\leq 70\%$), the channel loss rate does not affect our scheme much. When the loss rate is large (e.g., 90% as in Figure 5(a)), we can observe the drop of data packet authentication rate when the percentage of forged CDM packets is low. An interesting result is that the data packet authentication rate begins to increase when the percentage of forged CDM packets increases. This is because the sensor nodes can get the disclosed key from forged CDM packets when they cannot get it from the authentic ones.

The channel loss rate does affect the average authentication delay, which can be seen in Figure 5(b). The reason is that a sensor node needs to wait longer time to get the disclosed key. In addition, the figure also shows that the percentage of forged CDM message does not have an significant impact on the average data packet authentication delay.

5 Related Work

Security issues such as broadcast authentication in sensor networks have been investigated by many researchers [15, 6, 12]. Due to the limited resources at sensor nodes, asymmetric cryptography based solutions [8, 14, 17] are usually impractical for sensor networks. In the following, we only review authentication schemes based on symmetric cryptography.

Cheung proposed a scheme (OLSV) based on delayed disclosure of keys by the sender to authenticate the link-state routing updates between routers [7]. Anderson et al. used the same technique in their Guy Fawkes protocol to authenticate the message between two parties [1]. However, their protocol cannot tolerate packet loss. Briscoe proposed the FLAMeS protocol [4], and Bergadano et al. presented an authentication protocol for multicast [3]. Both are similar to the OLSV protocol [7]. Canetti et al. proposed to use k different keys to authenticate the multicast messages with k different MAC's for sender authentication [5]. But, their scheme has high communication overhead because of the k MAC's for each message. Perrig [9] introduced a verification efficient signature scheme named BiBa based on one-way hash functions without trapdoors. The drawback of this scheme is its high signature generation and large communication overhead for public key distribution.

Perrig et al. proposed two schemes (TESLA and EMSS) for efficient multicast authentication over lossy channels [10]. TESLA requires loose time synchronization between sender and receiver, and does not provide non-repudiation. In contrast, EMSS does not require time synchronization, but introduces more signatures and communication overhead. However, TESLA requires a digital signature operation to bootstrap itself, which is impractical in resource constrained sensor networks. Instead of a digital signature, μTESLA [12] simply uses symmetric cryptography to distribute initial parameters to the sensor nodes individually. The drawback of this solution is its high communication overhead when the number of sensor nodes is large. Our extension in this paper is to address this problem. Some other extensions to TESLA, such as immediate authentication, multiple concurrent TESLA instances, were later proposed in [11].

μTESLA and our proposed scheme do not assume tamper-resistant hardware, and do not guarantee the confidentiality of the broadcast packets. Based on the assumption of tamper-resistant hardware, Basagni et al. presented a key management scheme to periodically update the symmetric keys shared by all sensor nodes [2]. With this key shared among all sensor nodes, authenticated broadcast can be easily implemented. However, this scheme cannot prevent a (compromised) sensor node from sending forged messages if an attacker can reuse the tamper-resistant hardware.

6 Conclusion and Future Work

In this paper, we presented a multi-level key chain scheme to efficiently distribute the key chain commitments for the broadcast authentication scheme named μTESLA. By using pre-determination and broadcast, our scheme removed μTESLA's requirement of a unicast-based distribution of initial key chain commitments, which introduces high communication overhead in large distributed sensor networks. We also proposed several techniques, including periodic broadcast of commitment distribution messages and random selection strategies, to improve the survivability of our scheme and defeat some DOS attacks. Our experiments and analysis showed that the resulting scheme satisfies several nice properties, including low overhead, tolerance of message loss, scalability to large networks, and resistance to replay attacks as well as some known DOS attacks.

The limitation of our scheme is that when a sensor node doesn't get a commitment during a time interval, it must wait for a long period of time to recover from this failure. We will seek solutions to this problem in our future research. In addition, we will study broadcast authentication involving multiple base stations and the implementation of our scheme in real sensor networks.

References

[1] Ross Anderson, Francesco Bergadano, Bruno Crispo, Jong-Hyeon Lee, Charalampos Manifavas, and Roger Needham. A new family of authentication protocols. In *Operating Systems Review*, October 1998.

[2] Stefano Basagni, Kris Herrin, Danilo Bruschi, and Emilia Rosti. Secure pebblenets. In *Proceedings of ACM International Symposium on Mobile ad hoc networking and computing*, pages 156–163, 2001.

[3] F. Bergadano, D. Cavagnino, and B. Crispo. Individual single source authentication on the mbone. In *ICME 2000*, August 2000.

[4] B. Briscoe. Flames: Fast, loss-tolerant authentication of multicast stream. Technical report, BT Research, 2000.

[5] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas. Multicast security: A taxonomy and some efficient constructions. In *Infocom'99*, 1999.

[6] D.W. Carman, P.S. Kruus, and B.J.Matt. Constrains and approaches for distributed sensor network security. Technical report, NAI Labs, 2000.

[7] Steven Cheung. An efficient message authentication scheme for link state routing. In *13th Annual Computer Security Applications conference*, San Diego, Calif, December 1997.

[8] R. Gennaro and P. Rohatgi. How to sign digital streams. Technical report, IBM T.J.Watson Research Center, 1997.

[9] Adrian Perrig. The biba one-time signature and broadcast authentication protocol. In *Proceedings of the ACM Conference on Computer and Communications Security*, November 2001.

[10] Adrian Perrig, Ran Canetti, Dawn Song, and Doug Tygar. Efficient authentication and signing of multicast streams over lossy channels. In *Proc. of IEEE Security and Privacy Symposium*, May 2000.

[11] Adrian Perrig, Ran Canetti, Dawn Song, and Doug Tygar. Efficient and secure source authentication for multicast. In *Proceedings of Network and Distributed System Security Symposium*, February 2001.

[12] Adrian Perrig, Robert Szewczyk, Victor Wen, David Culler, and J.D. Tygar. Spins: Security protocols for sensor networks. In *Proceedings of Seventh Annual International Conference on Mobile Computing and Networks*, July 2001.

[13] Ron Rivest. The rc5 encryption algorithm. In *Proceedings of the 1st International Workshop on Fast Software Encryption*, volume 809, pages 86–96, 1994.

[14] Pankaj Rohatgi. A compact and fast hybrid signature scheme for multicat packet authentication. In *6th ACM Conference on Computer and Communications Security*, November 1999.

[15] F. Stajano and R. Anderson. The resurrecting duckling: security issues for ad hoc networks. In *Proc. of Security Protocols: 7th International Workshop*, pages 172–194, 1999.

[16] U.S. National Institute of Standards and Technology. DES modes of operation. Federal Information Processing Standards Publication 81 (FIPS PUB 4-3), December 1980.

[17] C.K. Wong and S. S. Lam. Digital signatures for flows and multicasts. In *Proc. IEEE ICNP'98*, 1998.

A A Detailed Description of Scheme IV

Initialization

During the initialization phase, all the sensor nodes synchronize their clocks with the base station. (Alternatively, the base station and all the sensor nodes may synchronize their clocks with a time service.) In addition, the base station generates the following parameters: (1) the initial random key K_n for the high-level key chain; (2) a sequence of keys $K_i = F_0(K_{i+1})$ in the high-level key chain, where $i = 0, 1, ..., n-1$, and F_0 is a pseudo random function; (3) the duration Δ_0 of each time interval for the high-level key chain; (4) the starting time T_1 for the high-level key chain; (5) duration Δ_1 of the low-level time intervals; (6) the disclosure lag d for the low-level key chains; (7) the the maximum clock discrepancy δ_{Max} during the life time of the sensor network.

A constraint for these parameters is that $\Delta_1 \times d + \delta_{Max} <$ the duration of the time interval for the high-level key chain. Otherwise, the disclosure of a high-level key may disclose a low-level key that should not be disclosed.

The base station distributes the following parameters to the sensor nodes: (1) K_0, (2) Δ_0, (3) T_1, (4) Δ_1, (5) d, and (6) δ_{Max}. Here we predetermine all the parameters for the low-level key chains except for the commitments. Alternatively, we may allow the base station to dynamically choose these parameters and distribute them to the sensors in the commitment distribution messages. In this case, the authentication procedure below should be changed slightly.

Note that the initialization phase does not introduce significantly more overhead than the original μTESLA. In the original μTESLA, it is at least necessary to distribute the master keys to the sensor nodes so that the base station shares some common keying material with each sensor node. The aforementioned parameters can be distributed to the sensor nodes along with the master keys.

Broadcast of Commitment Distribution Messages

When the base station needs to broadcast authenticated messages to the sensors, it generates parameters for each low-level key chain in a similar way to TESLA and μTESLA [10, 11, 12]. Assume the base station decides to divide each time interval I_i into m smaller intervals, denoted $I_{i,1}$, $I_{i,2}$, ..., $I_{i,m}$. The base station generates the low-level key chain by computing $K_{i,m} = F_{01}(K_{i+1})$, and $K_{i,j} = F_1(K_{i,j+1})$, where $j = 0, 1, ..., m-1$ and F_1 is a pseudo random function. Thus, the base station has the low-level key chain $\langle K_{i,0} \rangle$. The base station distributes the relevant information about the low-level key chain $\langle K_{i,0} \rangle$ in CDM_{i-1} during the time interval I_{i-1}, and the image of this information under a pseudo random function H in CDM_{i-2} during I_{i-2}.

Each commitment distribution message CDM_i contains the index of the high-level time interval, the commitment of the low-level key chain $\langle K_{i+1,0} \rangle$, the image under H of the commitment of $\langle K_{i+2,0} \rangle$, and the high-level authentication key K_{i-1}.

$BS \rightarrow S : CDM_i = i|K_{i+1,0}|H(K_{i+2,0})|MAC_{K_i'}(i|K_{i+1,0}|H(K_{i+2,0}))|K_{i-1}$.

The base station randomly chooses $F \times \Delta_0$ points during each time interval I_i, and broadcasts CDM_i at these time points.

Optionally, the base station may distribute $K_{1,0}$ and $H(K_{2,0})$ to the sensors during the initialization phase so that the sensors can use the low-level key chain $\langle K_{1,0} \rangle$ in the time interval I_1 and authenticate $K_{2,0}$ upon receiving CDM_1.

Authentication of Commitment Distribution Messages

Assume that a sensor node S has $m+1$ buffers for commitment distribution messages. When S receives a copy of CDM_i at time t_i during the time interval I_i, it processes this message according to the following procedure.

1. S checks the security condition for CDM_i, i.e., $t_i + \delta_{Max} < T_{i+1}$. S discards the packet and stops if the security condition is not held.

2. If S has an authenticated copy of CDM_{i-1}, S must have received a previous copy of CDM_i and saved K_{i-1} and $H(K_{i+1,0})$. Check whether the K_{i-1} and $H(K_{i+1,0})$ in the current CDM_i are the same as those saved copies. If yes, go to step 6. Otherwise, S discards the message and stops.

3. S authenticates K_{i-1} against a previously disclosed key K_j by verifying that $K_{i-1} = F^{i-1-j}(K_j)$. (Note that K_j always exists since K_0 was distributed to each sensor node during initialization.) If this verification fails, S discards the message and stops. Otherwise, S replaces K_j with K_{i-1}.

4. For each copy c of CDM_{i-1}, S authenticates c by verifying its MAC with K_{i-1} disclosed in CDM_i. If this verification fails, S discards c and continues the verification for the next copy of CDM_{i-1}. Otherwise, S discards all the other copies of CDM_{i-1} and makes c the authenticated copy of CDM_{i-1}.

5. If S has an authenticated copy of CDM_{i-1}, it then further authenticates $K_{i+1,0}$ enclosed in CDM_i by verifying that applying H to $K_{i+1,0}$ results in $H(K_{i+1,0})$ included in CDM_{i-1}. If this verification fails, S simply drops the copy of CDM_i and stops. Otherwise, S saves $H(K_{i+1,0})$.

6. S uses the random selection strategy discussed in 3.4 to decide whether to save the current copy of CDM_i or not. (Note that if the current step is being executed, all the copies of CDM_{i-1} should have been discarded.) Further assume the current copy of CDM_i is the jth copy. If $j < m$, S still has free buffers available, and S saves it in one of the empty buffers. Otherwise, S keeps this copy with the probability m/j, and places it in a randomly selected buffer (among the m occupied buffers).

Note that the immediate authentication of $K_{i+1,0}$ in CDM_i does not imply the authentication of CDM_i itself. An attacker can replace $H(K_{i+2,0})$ in CDM_i and still have the resulting message pass the verification. Thus, S has to use the random selection strategy to save the copies of CDM_i.

Broadcast and Authentication of Normal Messages

Broadcast and authentication of normal messages are performed in the same way as in the extended TESLA [11], except for the distribution of the key chain commitments, which is handled in the distribution and authentication of commitment distribution messages.