Multilevel μTESLA: Broadcast Authentication for Distributed Sensor Networks

DONGGANG LIU and PENG NING
North Carolina State University

Broadcast authentication is a fundamental security service in distributed sensor networks. This paper presents the development of a scalable broadcast authentication scheme named *multilevel μTESLA* based on μTESLA, a broadcast authentication protocol whose scalability is limited by its unicast-based initial parameter distribution. Multilevel μTESLA satisfies several nice properties, including low overhead, tolerance of message loss, scalability to large networks, and resistance to replay attacks as well as denial-of-service attacks. This paper also presents the experimental results obtained through simulation, which demonstrate the performance of the proposed scheme under severe denial-of-service attacks and poor channel quality.

Categories and Subject Descriptors: C.2.0 [**Computer and Communication Networks**]: General—*Security and protection*; C.2.2 [**Network Protocols**]: Applications; D.4.6 [**Operating Systems**]: Security and Protection—*Authentication*

General Terms: Design, Security

Additional Key Words and Phrases: Broadcast authentication, sensor networks, TESLA

1. INTRODUCTION

A distributed sensor network usually consists of one or several computationally powerful nodes called *base stations* and a large number of inexpensive, low-capacity nodes called *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.

This work is partially supported by NCSU Center for Advanced Computing and Communication (CACC). A preliminary version of this paper appeared in *Proceedings of the 10th Annual Network and Distributed Systems Security Symposium*, pages 263–276, February 2003.
Authors' address: Donggang Liu and Peng Ning, Cyber Defense Laboratory, Department of Computer Science, North Carolina State University, Raleigh, NC 27695-8207; email: {dliu,pning}@ncsu.edu.

Broadcast authentication is an essential service in distributed sensor networks. Because of the large number of sensor nodes and the broadcast nature of wireless communication, it is usually desirable for the base stations to broadcast commands and data to the sensor nodes. The authenticity of such commands and data is critical for the normal operation of sensor networks. If convinced to accept forged or modified commands or data, sensor nodes may perform unnecessary or incorrect operations, and cannot fulfill the intended purposes of the network. Thus, in hostile environments (e.g., battle field, antiterrorists operations), it is necessary to enable sensor nodes to authenticate broadcast messages received from the base station.

Providing broadcast authentication in distributed sensor networks turns out to be a nontrivial task. On the one hand, public-key-based digital signatures (e.g., RSA [Rivest et al. 1978]), which are typically used for broadcast authentication in traditional networks, are too expensive to be used in sensor networks, due to the intensive computation involved in signature verification and the resource constraints on sensor nodes. On the other hand, secret-key-based mechanisms (e.g., HMAC [Krawczyk et al. 1997]) cannot be directly applied to broadcast authentication, since otherwise a compromised receiver can easily forge any message from the sender.

A protocol named μTESLA [Perrig et al. 2001b] has been proposed for broadcast authentication in distributed sensor networks, which is adapted from a stream authentication protocol called TESLA [Perrig et al. 2000b]. μTESLA employs a chain of authentication keys linked to each other by a pseudorandom function [Goldreich et al. 1986], which is by definition a one-way function. Each key in the key chain is the image of the next key under the pseudorandom function. μTESLA achieves broadcast authentication through delayed disclosure of authentication keys in the key chain. The efficiency of μTESLA is based on the fact that only pseudorandom function and secret-key-based cryptographic operations are needed to authenticate a broadcast message. (More details of μTESLA can be found in Section 2.)

The original TESLA uses broadcast to distribute the initial parameters required for broadcast authentication. The authenticity of these parameters is guaranteed by a digital signature generated by the sender. However, due to the low bandwidth of a sensor network and the limited computational resources at each sensor node, μTESLA cannot distribute these initial parameters using public-key cryptography. 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 Perrig et al. [2001b] has 10 Kbps at the physical layer and supports 30-byte packets. 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 $\frac{4000\times30\times8}{10{,}240} = 93.75$ s 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 present a series of techniques to extend the capabilities of μTESLA. 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 node and the base station). To provide more flexibility, especially to prolong the lifetime of μTESLA without requiring a very long key chain, we introduce a multilevel 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 transmissions and random selection strategies to deal with the messages that distribute key chain commitments. The resulting scheme, which is named *multilevel μTESLA*, 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).

We also report experimental results obtained through simulation, which are intended to study the performance of multilevel μTESLA under severe DOS attacks and poor channel quality. The experimental results demonstrate that our scheme can tolerate high channel loss rate and is resistant to known DOS attacks to a certain degree.

The rest of this paper is organized as follows. Section 2 gives a brief overview of μTESLA. Section 3 presents the development of the multilevel μTESLA scheme. Section 4 presents our experiments performed through simulation. 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 μTESLA scheme, from which the multilevel μTESLA 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 Perrig et al. [2000b]. Otherwise, a malicious receiver can easily forge any packet from the sender. However, due to the high communication, computation, and storage overheads 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 [Perrig et al. 2001b]. 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 in the key chain and repeatedly performs a pseudorandom 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 is assigned to the ith time interval. With the pseudorandom function F, 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 keys $K_i, j+1 \leq i \leq n$. 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 pseudorandom function operations. When a broadcast message is available in ith 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 receivers, 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 $\lfloor (T_c + \Delta - T_1)/T_{\text{int}} \rfloor < I_i + d - 1$, 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, that is, 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 [Perrig et al. 2000a]. The only difference between TESLA and μTESLA is in their key chain 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 overheads. 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 [Perrig et al. 2001a]. The basic idea is to include an image under a pseudorandom 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. MULTILEVEL 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. That is, the technique is developed for broadcast authentication, but it relies on unicast-based technique to distribute the initial parameters.

In this section, we develop several techniques to extend the capability 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 except for Scheme V, which improves over Scheme IV only in special cases where the base station is very resourceful in terms of computational power. The final scheme, a multilevel μTESLA scheme, then has three variations based on Scheme IV, Scheme V, and a trade-off between Schemes IV and V, respectively.

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 Perrig et al. [2001b]. 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 because 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 long time intervals to divide the time period. However, both options may introduce problems. If a long

key chain is used, the base station has to allocate a large amount of memory to store the key chain. For example, in our later experiments, the duration of each time interval is 100 ms. To cover one day, the base station have to allocate $24 \times 60 \times 60 \times 10 \times 8 = 6{,}912{,}000$ bytes memory to store the keys. This may not be desirable in some applications. In addition, the receivers have to perform intensive computation of pseudorandom functions if there is a long delay (which covers a large number of time intervals) between broadcast messages in order to authenticate a later disclosed key. Continuing from the previous example, if the time between two consecutive messages received in a sensor node is one hour, the node has to perform $60 \times 60 \times 10 = 36{,}000$ pseudorandom operations to verify the disclosed key, which may be prohibitive in resource constrained sensor nodes. 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 a larger buffer at each sensor node. Though the extensions to TESLA [Perrig et al. 2001a] can remove the delay in authenticating the data payload 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 compute a large number of pseudorandom functions in order to authenticate the first broadcast message. As we see in the previous example, one hour delay will introduce a huge number of pseudonumber operations. In addition, the key chain must be fairly long so that it does not run out before the sensor network's lifetime 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 chain commitment, but also those of this simple scheme.

3.2 Scheme II: Naive Two-Level μTESLA

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 multiple levels of key chains. In the following several subsections, we first investigate the special case of two-level key chains to enhance its security and robustness, and then extend the results to multilevel key chains in Section 3.6.

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 lifetime 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 lifetime of a sensor network is divided into n_0 (long) intervals of duration Δ_0, denoted as $I_1, I_2, \ldots,$ and I_{n_0}. The high-level key chain has $n_0 + 1$ elements

$K_0, K_1, \ldots, K_{n_0}$, which are generated by randomly picking K_{n_0} and computing $K_i = F_0(K_{i+1})$ for $i = 0, 1, \ldots, n_0 - 1$, where F_0 is a pseudorandom 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.

Because the duration of the high-level time intervals is usually very long compared with 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 $\delta_{\max}$ is the maximum clock discrepancy between the base station and the sensor node.

Each time interval I_i is further divided into n_1 (short) intervals of duration Δ_1, denoted as $I_{i,1}, I_{i,2}, \ldots, I_{i,n_1}$. If needed, the base station generates a low-level key chain for each time interval I_i by randomly picking K_{i,n_1} and computing $K_{i,j} = F_1(K_{i,j+1})$ for $j = 0, 1, \ldots, n_1 - 1$, where F_1 is a pseudorandom 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 [Perrig et al. 2000b, 2001b]. 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: $i' < (i-1) * n_1 + j + d$, where $i' = \lfloor \frac{t - T_1 + \delta_{\max}}{\Delta_1} \rfloor + 1$, and $\delta_{\max}$ 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 $\delta_{\max}$ between the base station and the sensor nodes throughout the lifetime of the sensor network are distributed to the sensor nodes.

In order for the sensor nodes to use a low-level key chain $\langle K_{i,0} \rangle$ during the time interval I_i, they must authenticate the commitment $K_{i,0}$ before T_i. To achieve this goal, the base station broadcasts a *commitment distribution message*, denoted as CDM_i, during each time interval I_i. (In the rest of this paper, we use commitment distribution message and its abbreviation *CDM* interchangeably.) This message consists of the commitment $K_{i+2,0}$ of the low-level key chain $\langle K_{i+2,0} \rangle$ and the key K_{i-1} in the high-level key chain. Specifically, the base station constructs the CDM_i message as follows:

$CDM_i = i|K_{i+2,0}|MAC_{K'_i}(i|K_{i+2,0})|K_{i-1}$, where "|" denotes message concatenation, and K'_i is derived from K_i with a pseudorandom function other than F_0 and F_1.

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 $K_{i,0}$ in CDM_{i-2}.

Because the high-level authentication key K_i is disclosed in CDM_{i+1} during the time interval I_{i+1}, each sensor node needs to store CDM_i until it

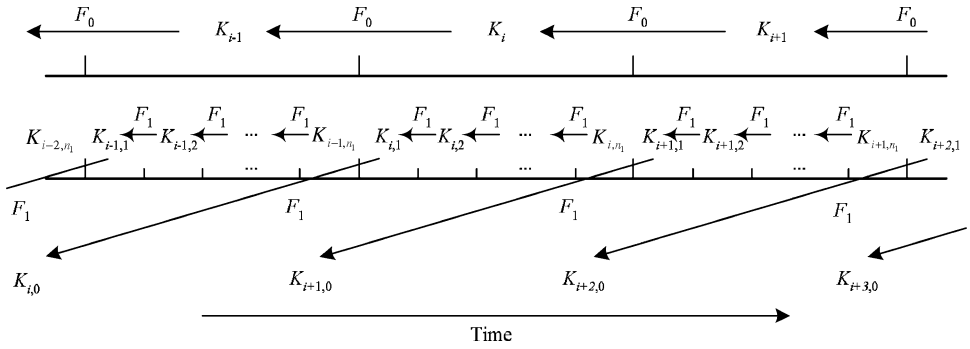


Fig. 1. The two levels of key chains in Scheme II. Each key K_i is used for the high-level time interval I_i, and each key $K_{i,j}$ is used for the low-level time interval $I_{i,j}$. F_0 and F_1 are different pseudorandom functions. Each commitment $K_{i,0}$ is distributed during the time interval I_{i-2}.

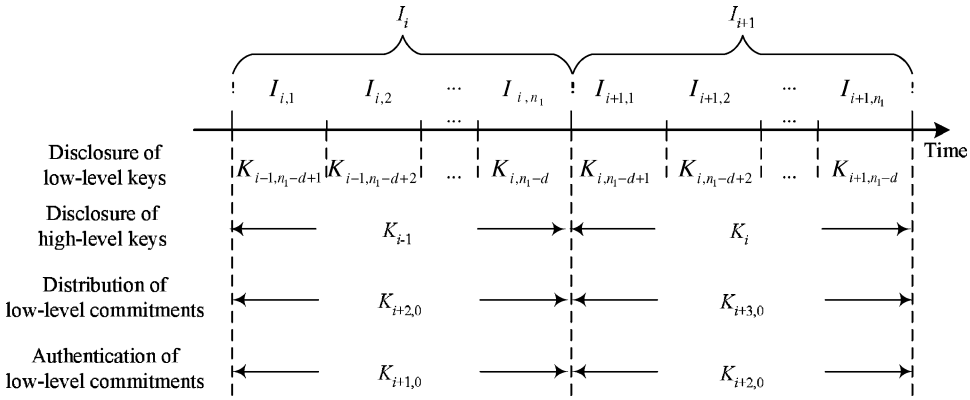


Fig. 2. Key disclosure schedule in Scheme II.

receives CDM_{i+1}. Each sensor node also stores a key K_j, which is initially K_0. After receiving K_{i-1} in CDM_i, the sensor node authenticates it by verifying that $F_1^{i-1-j}(K_{i-1}) = K_j$. Then the sensor node replaces the current K_j with K_{i-1}.

Let us suppose a sensor node has received CDM_{i-2}. Upon receiving CDM_{i-1} during I_{i-1}, the node can authenticate CDM_{i-2} with K_{i-2} disclosed in CDM_{i-1}, and thus verify $K_{i,0}$. As a result, the sensor node can authenticate broadcast messages sent by the base station using the μTESLA key chain $\langle K_{i,0} \rangle$ during the high-level 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 lifetime of the sensor network. The low-level key chains depend on the high-level key chain to distribute and authenticate 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 does not have to be delayed too much.

The security of this scheme follows directly from the security of μTESLA. Note that the high-level key chain is only used to authenticate the commitment of each low-level key chain. As long as the security condition of each μTESLA key chain is satisfied, the two-level μTESLA has the same degree of security as all the μTESLA instances involved in this scheme. Thus, similar to μTESLA and TESLA, a sensor node can detect forged messages by verifying the MAC with the corresponding authentication key once the sensor node receives it. In addition, replay attacks can be easily defeated if a sequence number is included in each message.

In the preliminary version of this paper [Liu and Ning 2003a], we used a variation of this naive two-level key chains scheme based on the immediate authentication extension to TESLA [Perrig et al. 2001a]. The intention was to enable a sensor node to authenticate the key included in a *CDM* immediately after it receives the message. Specifically, we included an image of the key chain commitment contained in the next *CDM* under a pseudorandom function in the current *CDM*. Once this *CDM* is authenticated (after receiving the next *CDM*), the key chain commitment in the next *CDM* can be authenticated immediately. However, a further investigation reveals that this alternative does not save much. Unlike the data to be immediately authenticated in Perrig et al. [2001a], a key chain commitment usually has the same length as its image under a pseudorandom function. Thus, the above variation is equivalent to having each key chain commitment included in two consecutive *CDM*.

3.3 Scheme III: Fault Tolerant Two-Level μTESLA

Scheme II does not tolerate message losses as well as μTESLA and TESLA. There are two types of message losses: the losses of normal messages, and the losses of *CDM*. Both may cause problems for Scheme II. First, the low-level keys are not entirely chained together. Thus, losses of key disclosure messages for later keys in a low-level key chain cannot be recovered even if the sensor node can receive keys in some later low-level key chains. For example, consider the last key K_{i,n_1} that is used to authenticate the packet in the key chain of time interval I_i. If the packets that disclose K_{i,n_1} are lost, the sensor node then has no way to authenticate this packet. As a result, a sensor node 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-2} does not reach a sensor node, the node 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 pretty 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,n_1}

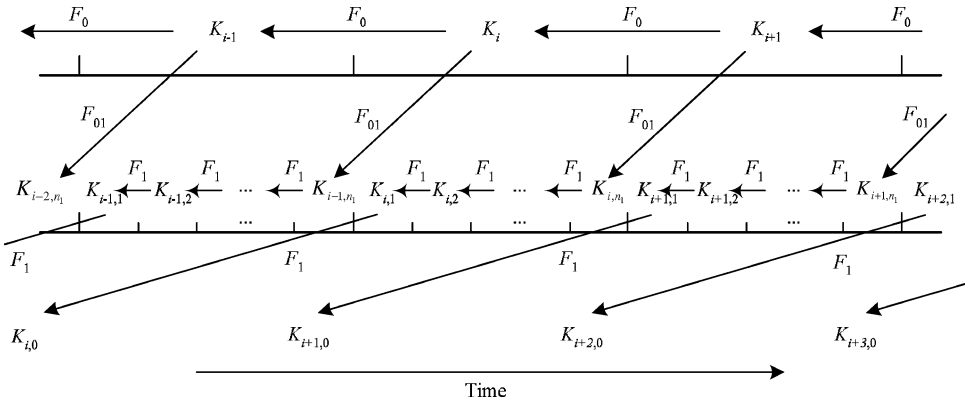


Fig. 3. The two levels of key chains in Scheme III. It differs from Figure 1 in that each K_{i,n_1} is derived from K_{i+1} using an additional pseudorandom function F_{01}.

randomly, we derive each K_{i,n_1} from a high-level key K_{i+1} (which is to be used in the next high-level time interval) through another pseudorandom function F_{01}. That is, $K_{i,n_1} = F_{01}(K_{i+1})$. As a result, a sensor node can recover any authentication key $K_{i,j}$ as long as it receives a *CDM* that discloses $K_{i'}$ with $i' \geq i+1$, even if it does not receive any later low-level key $K_{i,j'}$ with $j' \geq j$. 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 node at all during a time interval I_i, CDM_i will not be delivered to the sensor node. However, the impact of temporary communication failures can be reduced by standard fault tolerant approaches.

One possible solution to mitigate the second problem is to include each key chain commitment in multiple *CDM*s. For example, we may include each key chain commitment $K_{i,0}$ in l consecutive *CDM*s, $CDM_{i-2}, \ldots, CDM_{i-(l+1)}$. As a result, CDM_i includes the key chain commitments $K_{i+2,0}, \ldots, K_{i+1+l,0}$. A sensor node can recover and authenticate $K_{i,0}$ if it receives either any two of the above l *CDM*s, or one of the l *CDM*s and CDM_{i-1}. However, this also increases the size of *CDM*s as well as the *CDM* buffer on sensor nodes. Moreover, the larger a packet is, the more possible that it is lost in wireless communication. Considering the fact that packets in distributed sensor networks usually have limited size (e.g., the payload of each packet in TinyOS [Hill et al. 2000] is at most 29 bytes), we decide not to go with this solution.

Instead, we propose to have the base station periodically broadcast the *CDM* during each time interval. Assuming that the frequency of this broadcast is F, each *CDM* is therefore broadcasted $F \times \Delta_0$ times. To simplify the analysis, we assume the probability that a sensor node cannot receive a broadcast of a *CDM* is p_f. Thus, the probability that a sensor node cannot receive any copy of the *CDM* is reduced to $p_f^{F \times \Delta_0}$.

Note that even if a sensor node cannot receive any *CDM* 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 *CDM* in time interval I_i only prevents a sensor node from authenticating broadcast messages during I_{i+1}. As long as

the sensor node gets a *CDM*, it can derive all the low-level keys in the previous time intervals.

By periodically broadcasting *CDM*s, Scheme III introduces more overhead than Scheme II. Let us consider the overhead on the base station, the sensor nodes, and the communication channel, respectively. Compared with Scheme II, this scheme does not change the computation of *CDM*s in the base station, but increases the overhead to transmit *CDM*s 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 sensor nodes are affected much less than the base station in a benign environment because each sensor node only needs to process one *CDM* for each time interval. Thus, the sensor nodes have roughly the same overhead as in Scheme II. However, we will show that a sensor node has to take a different strategy in a hostile environment in which there are DOS attacks. We will delay the discussion of sensor nodes' overhead until we introduce our counter measures.

This approach increases the overhead in the communication channel by $F \times \Delta_0$ times because the *CDM* for each time interval is repeated $F \times \Delta_0$ times. Assume the probability that a sensor node cannot receive a *CDM* is $p_f = 1/2$ and $F \times \Delta_0 = 10$. Under our simplified assumption, the probability that the sensor node cannot receive any of the 10 *CDM*s is $p_f^{F \times \Delta_0} < 0.1\%$. Further assume that Δ_0 is 1 min, which is quite short as the interval length for the high-level key chain. Thus, there is one *CDM* per 6 s. Assume the bandwidth is 10 Kbps and each *CDM* packet is 36 bytes = 288 bits, which includes the 29 byte *CDM* and the 7 byte packet header as in our experiments (Section 4). Then the relative communication overhead is $\frac{288}{10{,}240 \times 6} = 0.47\%$. This is certainly optimistic because we assume perfect channel utilization. However, it still shows that Scheme III introduces very reasonable communication overhead in typical sensor networks.

The security of Scheme III is similar to that of Scheme II. The only difference between Scheme II and Scheme III is that in Scheme III, each low-level key chain is derived from a high-level key with a pseudorandom function F_{01}. Each high-level key is disclosed at least one high-level time interval after the corresponding low-level key chain is used. Thus, as long as the pseudorandom function is secure (i.e., it is computationally infeasible to distinguish the output of the pseudorandom function from a true random number), Scheme III is equivalent to Scheme II, which does not have F_{01} connecting the two levels of key chains.

One limitation of Scheme III is that if a sensor node misses all copies of CDM_i during the time interval I_i, it cannot authenticate any data packets received during I_{i+2} before it receives an authentic K_j, $j > i + 2$. (Note that the sensor node does not have to receive an authentic *CDM*. As long as the sensor node can authenticate a high-level key K_j with $j > i + 2$, it can derive the low-level keys through the pseudorandom functions F_0, F_{01}, and F_1.) Since the earliest high-level key K_j that satisfies $j > i + 2$ is K_{i+3}, and K_{i+3} is disclosed during I_{i+4}, the sensor node has to buffer the data

packets received during I_{i+2} for at least the duration of one high-level time interval.

3.4 Scheme IV: DOS-Tolerant Two-Level μTESLA

In Scheme III, the usability of a low-level key chain depends on the authentication of the key chain commitment contained in the corresponding *CDM*. A sensor node cannot use the low-level key chain $\langle K_{i,0} \rangle$ for authentication before it can authenticate $K_{i,0}$ distributed in CDM_{i-2}. This makes the *CDM*s attractive targets for attackers. An attacker may disrupt the distribution of *CDM*s, and thus prevent the sensor nodes 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 sensor nodes 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 sensor nodes' 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 *CDM*s is to jam the communication channel. We may have to resort to techniques such as frequency hopping if the attacker completely jams the communication channel. This is out of the scope of this paper. The attacker may also jam the communication channel only when the *CDM*s 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 *CDM*s randomly or in a pseudorandom 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 *CDM*s randomly.

An attacker may forge commitment distribution messages to confuse the sensor nodes. If a sensor node does not have a copy of the actual CDM_i, it will not be able to get the correct $K_{i+2,0}$, and cannot use the low-level key chain $\langle K_{i+2,0} \rangle$ during the time interval I_{i+2}.

Consider a *CDM*: $CDM_i = i|K_{i+2,0}|MAC_{K'_i}(i|K_{i+2,0})|K_{i-1}$. Once seeing such a message, the attacker learns i and K_{i-1}. Then the attacker can replace the actual $K_{i+2,0}$ or $MAC_{K'_i}(i|K_{i+2,0})$ with arbitrary values $K'_{i+2,0}$ or MAC', and forge another message: $CDM'_i = i|K'_{i+2,0}|MAC'|K_{i-1}$. Assume a sensor node has an authentic copy of CDM_{i-1}. The sensor node can verify K_{i-1} with K_{i-2} because K_{i-2} is included in CDM_{i-1}. However, the sensor node has no way to verify the authenticity of $K'_{i+2,0}$ or MAC' without the corresponding key, which will be disclosed later. In other words, the sensor node cannot distinguish between the authentic CDM_is and those forged by the attacker. If the sensor node does not save an authentic copy of CDM_i during I_i, it will not be able to get an authenticated $K_{i+2,0}$ even if it receives the authentication key K_i in CDM_{i+1} during I_{i+1}. As a result, the sensor node cannot use the key chain $\langle K_{i+2,0} \rangle$ during I_{i+2}.

One may suggest to distribute each $K_{i,0}$ in some earlier time intervals than I_{i-2}. However, this does not solve the problem. If a sensor node does not have an

authentic copy of the *CDM*, it can never get the correct $K_{i,0}$. To take advantage of this, an attacker can simply forge *CDM*s as discussed earlier.

We propose a random selection method to improve the reliable broadcast of commitment distribution messages. For the CDM_is received during each time interval I_i, each sensor node first tries to discard as many forged messages as possible. There is a simple test for a sensor node to identify some forged CDM_is during I_i. The sensor node 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. (Note that such a K_j always exists because the commitment K_0 of the high-level key chain is distributed during the initialization of the sensor nodes.) Messages that fail this test are certainly forged and should be discarded.

The simple test can filter out some forged messages; however, they do not rule out the forged messages discussed earlier. To further improve the possibility that the sensor node has an authentic CDM_i, the base station uses a random selection method to store the CDM_is that pass the above test. Our goal is to make the DOS attacks so difficult that the attacker would rather use constant signal jamming instead to attack the sensor network. In other words, we want to prevent the DOS attacks that can be achieved by sending a few packets. 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 test.

3.4.1 *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 *CDM*s broadcasted in each time interval. In a time interval I_i, each sensor node randomly selects one message from all copies of CDM_i it receives. 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 *CDM* 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 $1/k$. 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 $p = \frac{\#\text{forged copies}}{\#\text{total copies}}$. To maximize his attack, an attacker has to send as many forged copies as possible.

3.4.2 *Multiple-Buffer Random Selection.* The single-buffer random selection can be easily improved by having additional buffers for the *CDM*s. 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}, a 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 authenticates a copy, the sensor node can discard all the other buffered copies.

If a 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 *CDM*s with this strategy: m buffers for copies of CDM_i, and one buffer for the first weakly authenticated copy of CDM_{i+1}.

It is also easy to see that each sensor node needs to verify the MACs for at most m times. The number of pseudorandom function operations required to weakly authenticate the *CDM*s depends on the total number of (true and forged) *CDM*s a sensor node receives. 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 $p = \frac{\#\text{forged copies}}{\#\text{total copies}}$.

3.4.3 *Effectiveness of Random Selection.* In the rest of this subsection, we perform a further analysis using Markov chain theory to understand the effectiveness of the random selection strategy. Specifically, we would like to compute the probability that a sensor node has an authentic low-level key chain commitment before the key chain is used.

We assume that the base station sends out multiple *CDM*s in each high-level time interval so that the probability of all these *CDM*s being lost due to lossy channel is negligible. As our concern is about the availability of an authentic commitment for the low-level key chain before it is used, we consider the state of a sensor node only at the end of each high-level time interval.

At the end of each high-level time interval, we use Q_1 to represent that a sensor node buffers at least one authentic *CDM* in the previous high-level time interval, and Q_2 to represent that a sensor node buffers at least one authentic *CDM* in the current high-level time interval. We use $\neg Q_1$ (or $\neg Q_2$) to represent that Q_1 (or Q_2) is not true. Thus, with Q_1, Q_2, and their negations, we totally have four combinations, each of which makes one possible state of the sensor node. Specifically, state 1 represents $Q_1 \wedge Q_2$, which indicates the sensor node has an authentic copy of *CDM* in both the previous and the current high-level time interval. Similarly, state 2 represents $Q_1 \wedge \neg Q_2$, state 3 represents $\neg Q_1 \wedge \neg Q_2$, and state 4 represents $\neg Q_1 \wedge Q_2$. A sensor node may transit from one state to another when the current time moves from the end of one high-level time interval to the end of the next high-level time interval.

Figure 4 shows the state transition diagram, which is equivalent to the following transition matrix:

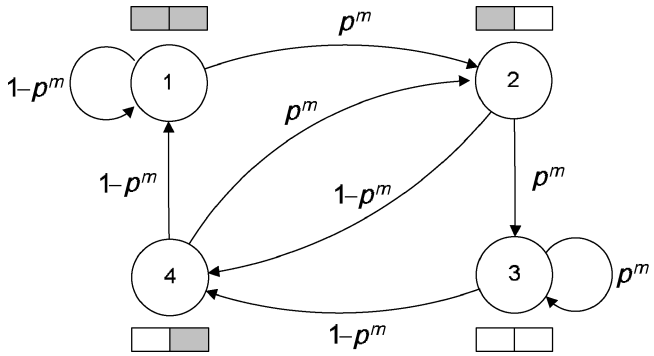


Fig. 4. State transition diagram for Scheme IV.

$$\mathbf{P} = \begin{pmatrix} 1-p^m & p^m & 0 & 0 \\ 0 & 0 & p^m & 1-p^m \\ 0 & 0 & p^m & 1-p^m \\ 1-p^m & p^m & 0 & 0 \end{pmatrix},$$

where $p = \frac{\#\text{forged copies of each } CDM}{\#\text{total copies of each } CDM}$ and m is the number of buffers for *CDM*s in each sensor node.

Among the four states, both states 1 and 2 imply that the sensor node gets an authentic key chain commitment for the low-level key chain to be used in the next high-level time interval. The reason is as follows: In both states 1 and 2, the sensor node already has an authentic *CDM* in the previous high-level time interval. Thus, it only needs a disclosed key to authenticate this message. If an attacker wants the DOS attack to be successful, he/she has to ensure the forged *CDM*s can be weakly authenticated. As a result, the sensor node can obtain a key to authenticate the *CDM* distributed in the previous high-level time interval, and then obtain an authenticated commitment of the low-level key chain to be used in the next high-level time interval, even if it does not have an authentic copy of the *CDM*. Therefore, the overall probability of having an authentic key chain commitment for the next key chain is the sum of the probabilities in state 1 and state 2.

To determine the probability of a sensor node being in each state, we need to find the steady state of the above process. Thus, we need to solve the equation $\Pi = \Pi \times \mathbf{P}$, where $\Pi = (\pi_1, \pi_2, \pi_3, \pi_4)$ and π_i represents the probability of the sensor node being in state i. That is,

$$(\pi_1, \pi_2, \pi_3, \pi_4) = (\pi_1, \pi_2, \pi_3, \pi_4) \times \begin{pmatrix} 1-p^m & p^m & 0 & 0 \\ 0 & 0 & p^m & 1-p^m \\ 0 & 0 & p^m & 1-p^m \\ 1-p^m & p^m & 0 & 0 \end{pmatrix}.$$

By solving the above equation and considering that $\pi_1 + \pi_2 + \pi_3 + \pi_4 = 1$,

we get

$$\begin{cases} \pi_1 = (1-p^m)^2 \\ \pi_2 = p^m(1-p^m) \\ \pi_3 = p^{2m} \\ \pi_4 = p^m(1-p^m). \end{cases}$$

Therefore, the probability that a sensor node has an authentic key chain commitment for the next low-level key chain is $P = \pi_1 + \pi_2 = 1 - p^m$. This result shows that the more buffers we have, the more effective this random selection strategy is. Moreover, according to the exponential form of the above formula, having a few more buffers can significantly increase the availability of an authenticated key chain commitment before the key chain is used.

3.4.4 *Frequency of CDMs.* One critical parameter in our proposed technique is the frequency of *CDM*s. We describe one way to determine this parameter. Consider a desirable probability P that a sensor node has an authenticated copy of a key chain commitment before the key chain is used. Let R_d, R_c, and R_a denote the fractions of bandwidth used by data, authentic *CDM*s, and forged *CDM*s, respectively. Assume each message has the same probability p_l of being lost in the communication channel. To simplify the analysis, we assume an attacker uses all available bandwidth to launch a DOS attack. Then we have $R_d + R_c + R_a = 1$. (Note that increasing the transmission of any type of messages will reduce the bandwidth for the other two types of messages. Thus, it is usually difficult in practice to *choose* R_d, R_c, and R_a as desired. Here we consider the relationship among the actual rates as they happen in communication.) To ensure the probability that a sensor node has an authentic low-level key chain commitment (before the use of the key chain) is at least P, we have

$$1 - \left(\frac{R_a \times (1-p_l)}{R_c \times (1-p_l) + R_a \times (1-p_l)} \right)^m \geq P.$$

This implies

$$R_a \leq \frac{\sqrt[m]{1-P}}{1-\sqrt[m]{1-P}} \times R_c.$$

Together with $R_d + R_c + R_a = 1$, we have

$$R_c \geq (1-R_d)(1-\sqrt[m]{1-P}). \tag{1}$$

Equation (1) presents a way to determine the frequency of *CDM*s to mitigate severe DOS attacks that use all available bandwidth to prevent the distribution and authentication of low-level key chain commitments. In other words, if we can determine the number m of *CDM* buffers based on resources on sensor nodes, the fraction R_d of bandwidth for data packets based on the expected application behaviors, the probability P of a sensor node authenticating a low-level key chain commitment before the key chain is used based on the expected security performance under severe DOS attacks, we can compute R_c and then determine the frequency of *CDM*s. Moreover, we may examine different choices of these parameters and make a trade-off most suitable for the sensor networks.

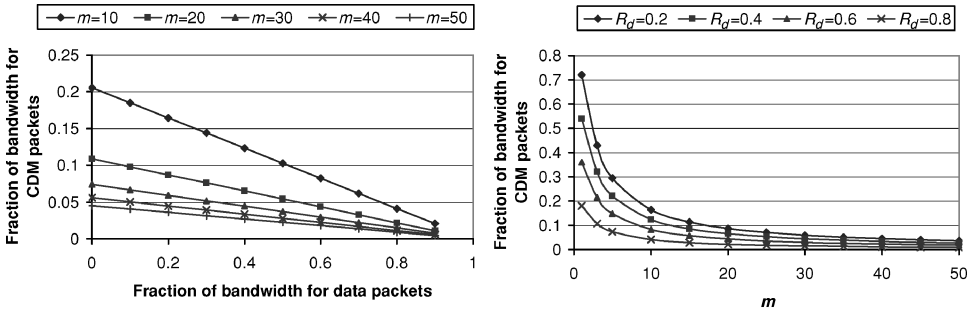


Fig. 5. Bandwidth required for *CDM*s to ensure 90% of low-level key chain commitments are authenticated before the key chains are used.

Figure 5 shows the fraction of bandwidth required for *CDM*s for different combinations of R_d and m given $P = 0.9$. We can see that the bandwidth required for *CDM*s in order to ensure $P = 0.9$ is substantially more than that required to deal with message losses. For example, as shown in Figure 5(a), when there are few data packets and each sensor node has only 10 buffers for *CDM*s, about 20% of the bandwidth must be used for *CDM*s in order to ensure 90% authentication rate for low-level key chain comments when there are severe DOS attacks. This is understandable because under such circumstances the sensor network is facing aggressive attackers that try everything possible to disrupt the normal operations of the network.

It is also shown in Figure 5(b) that the increase in the number of *CDM* buffers can significantly reduce the requirement for *CDM*s. As shown in Figure 5(b), when each sensor node has 40 *CDM* buffers, less than 5% of the bandwidth is required for *CDM*s. In addition, the shape of the curves in Figure 5(b) also shows that the smaller m is, the more effective an increase in m is.

Figure 5(a) further shows that the increase in data rate results in the decrease in the fraction of bandwidth required for *CDM*s. This is because when the data consume more bandwidth, there is less bandwidth for the DOS attacks, and in effect the requirement for *CDM*s is also reduced.

It is worth noting that the fractions for data and *CDM*s are the *actual* fractions of these messages that the sensor nodes receive, not the fractions *planned* by the base station. A message scheduled for transmission by the base station is not guaranteed to be transmitted if the DOS attack consumes too much bandwidth. Nevertheless, the above analysis provides a target frequency of *CDM*s, and the base station can adaptively change its transmission strategy to meet this target.

3.5 Scheme V: DOS-Resistant Two-Level μTESLA

Scheme IV can be further improved if the base station has enough computational and storage resources. Indeed, when at least one copy of each *CDM* can reach the sensor nodes, we can completely defeat the aforementioned DOS attack without the random selection mechanism.

The solution can be considered a variation of the immediate authentication extension to TESLA [Perrig et al. 2001a]. The idea is to include in CDM_i the image $H(CDM_{i+1})$ for each i, where H is a pseudorandom function. As a result, if a sensor node can authenticate CDM_i, it can get authentic $H(CDM_{i+1})$ and then authenticate CDM_{i+1} when it is received. Specifically, the base station constructs CDM_i for the high-level time interval I_i as follows:

$CDM_i = i|K_{i+1,0}|H(CDM_{i+1})|MAC_{K_i'}(i|K_{i+1,0}|H(CDM_{i+1}))|K_{i-1}$, where "|" denotes message concatenation, H is a pseudorandom function other than F_0 and F_1, and K_i' is derived from K_i with a pseudorandom function other than H, F_0 and F_1.

Let us suppose a sensor node has received CDM_i. Upon receiving CDM_{i+1}, the sensor node can authenticate CDM_i with K_i disclosed in CDM_{i+1}. Then the sensor node can immediately authenticate CDM_{i+1} by verifying that applying H to CDM_{i+1} results in the same $H(CDM_{i+1})$ included in CDM_i. As a result, the sensor node can authenticate a commitment distribution message immediately after receiving it.

Alternatively, if $H(CDM_1)$ is predistributed before deployment, the sensor node can immediately authenticate CDM_1 when receiving it, and then use $H(CDM_2)$ included in CDM_1 to authenticate CDM_2, and so on. One may observe that in this case, a sensor node does not use the disclosed high-level keys in *CDM*s directly. However, including such keys in *CDM*s are still useful. Indeed, when a sensor node fails to receive or keep an authentic *CDM*, it can use the random selection mechanism and the approach described in the previous paragraph to recover from the failure.

The cost, however, is that the base station has to compute the *CDM*s in the reverse order. That is, in order to include $H(CDM_{i+1})$ in CDM_i, the base station has to have CDM_{i+1}, which implies that it also needs CDM_{i+2}, and so on. Therefore, the base station needs to compute both the high-level and the low-level key chains completely to get the commitments of these key chains, and construct all the *CDM*s in the reverse order before the distribution of the first one of them. (Note that in Scheme IV, the base station only needs to compute the high-level key chain but not all the low-level ones during initialization. The base station may delay the computation of a low-level key chain until it needs to distribute the commitment of that key chain.)

This imposes additional computation during the initialization phase. Assume that all the key chains have 1000 keys. The base station needs to perform about 1,001,000 pseudorandom function operations to generate all the key chain commitments, and 1000 pseudorandom function operations and 1000 MAC operations to generate all the *CDM*s. Due to the efficiency of pseudorandom functions, such computation is still practical if the base station is relatively resourceful. For example, using MD5 as the pseudorandom function, a modern PDA can finish the above computation in several seconds. Moreover, the base station does not have to save the low-level key chains. Indeed, to reduce the storage overhead, the base station may compute a low-level key chain (again) when the key chain is needed. Thus, the base station only needs to store the high-level key chain and the MACs of all the *CDM*s. Further assume both the authentication

key and the image of a pseudorandom function are 8 bytes. To continue the earlier example, the base station needs $(8 + 8) \times 1000 = 16{,}000$ bytes to store the high-level key chain and the MACs.

The immediate authentication of CDM_i depends on the successful receipt of CDM_{i-1}. However, if a sensor node cannot receive an authentic CDM_i due to communication failure or an attacker's active disruption, the sensor node has to fall back to the techniques introduced in Scheme IV (i.e., the random selection strategies). This implies that the base station still needs to distribute *CDM*s multiple times in a random manner. The combination of these techniques is straightforward; we do not discuss it further in this paper.

Now let us assess how difficult it is for a sensor node to recover if it fails to receive an authentic *CDM*. We assume an attacker will launch a DOS attack to deter this recovery. To recover from the failure, the sensor node has to buffer an authentic *CDM* by the end of a later high-level time interval and then authenticate this message. For example, suppose a sensor node buffers an authentic CDM_{i+j}. If it receives a disclosed key in interval I_{i+j+1}, it can authenticate CDM_{i+j} immediately and gets $H(CDM_{i+j+1})$. The sensor node then recovers from the failure. Thus, if a sensor node fails to receive an authentic CDM_i, the probability that it recovers from this failure within the next l high-level time intervals is $1 - p^{m \times l}$, where $p = \frac{\text{\#forged copies ofeach } CDM}{\text{\#total copies of each } CDM}$ and m is the number of buffers for *CDM*s.

It is sensible to dynamically manage *CDM* buffers in sensor nodes in this scheme. There are three cases: (1) During normal operations, each sensor node only needs one buffer to save an authenticated *CDM* during each high-level time interval; (2) When a sensor node tries to recover from communication failures, it needs a relatively small number of *CDM* buffers to tolerate communication failures, as discussed in Section 3.3; (3) When a sensor node tries to recover from a loss of authentic *CDM*s under severe DOS attacks, the sensor node needs as many buffers as possible to increase its chance of recovery. Once a sensor node recovers an authentic *CDM*, it can fall back to only one *CDM* buffer because it can authenticate the next *CDM* once the message is received. This requires that each sensor node be able to detect the presence of DOS attacks. Fortunately, this can be done easily with high precision: If most buffered *CDM*s are forged, there must be a DOS attack.

The base station needs to broadcast each *CDM* multiple times to mitigate communication failures and to help sensor nodes recover from failures under potential DOS attacks. The frequency of *CDM*s required in this scheme can be determined in a similar way to Scheme IV. However, a sensor node in this scheme only needs a large number of *CDM* buffers temporarily during recovery. Moreover, a sensor node only needs to recover one authentic *CDM* in order to go back to normal operations, and the sensor node may recover over several high-level time intervals. Indeed, if we allow a sensor node to recover from such a failure over l high-level time intervals, by using the same process to derive equation (1), we can get the following equation:

$$R_c \geq (1 - R_d)(1 - \sqrt[m \cdot l]{1 - P}) \tag{2}$$

where R_c is the fraction of bandwidth required for *CDM*s, R_d is the fraction of bandwidth used by data packets, m is the number of buffers for *CDM*s, and P is the desired probability to recover from the failure over the next l high-level time intervals. It is easy to see that R_c decreases when m and l increase. Thus, the bandwidth required for *CDM*s can be much less than in Scheme IV.

Because the probability that a sensor node fails to receive an authentic *CDM* is unknown, it is not possible to derive the probability that the sensor node has an authentic low-level key chain commitment before the key chain is used. Nevertheless, this probability can be easily computed in the same way as in Section 3.4 if the aforementioned information is available.

From the above analysis, we can see that this scheme introduces additional computation requirement before deployment, though it can defeat the DOS attacks when at least one copy of each *CDM* reaches the sensor nodes. Fortunately, such computation is affordable if the base station is relatively resourceful. It is also possible to perform such computation on powerful machines and then download the result to the base station before deployment. In addition, the communication overhead and the storage overhead on sensor nodes in this scheme is potentially much less than that in Scheme IV, as discussed earlier. Thus, when the required computational resources are available (on either the base station or some other machines), Scheme V is more desirable. Otherwise, Scheme IV could be used to mitigate the DOS attacks.

3.6 Scheme VI: Multilevel μTESLA

Both Scheme IV and Scheme V can be extended to M-level key chain schemes. 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 distribute 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 are generated from this key, and all the key chains in level i, $1 \leq i \leq M-1$, 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 schemes. For security concerns, we need a family of pseudorandom functions. The pseudorandom function for each level and between adjacent levels should be different from each other. Such a family of pseudorandom functions has been proposed in Perrig et al. [2000b].

The benefit of having multi-level key chains is that it is more flexible in providing short key chains with short delays in authenticating data packets, compared with the two-level key chain schemes. As a result, a multilevel μTESLA scheme can scale-up to cover a long period of time. In practice, a three-level scheme is usually sufficient to cover the lifetime of a sensor network. For example, if the duration of a lowest-level time interval is 100 ms, and each key chain has 1000 keys, then a three-level scheme can cover a period of 10^8 s, which is over 3 years. In the following, we still present our techniques as generic multilevel key chains schemes for the sake of generality.

In addition to multilevel μTESLA schemes directly extended from Schemes IV and V, we can combine them into a hybrid scheme to achieve a trade-off between precomputation and operational overheads. Thus, we have three variations of multilevel μTESLA schemes. The first variation, which is named *DOS-tolerant multilevel μTESLA*, is extended from Scheme IV and is suitable for sensor networks where the base station is not very resourceful. The second variation, which is named *DOS-resistant multilevel μTESLA*, is extended from Scheme V. This variation is suitable for sensor networks with relatively short lifetime and relatively powerful base stations. The third variation, which is named *hybrid multilevel μTESLA*, is a trade-off between the above two variations. It sacrifices certain immediate authentication capability to exchange for less precomputation requirement.

In the following, we describe and analyze these variations, respectively.

3.6.1 *Variation I: DOS-Tolerant Multilevel μTESLA.* This variation of multilevel μTESLA scheme is a direct extension to Scheme IV. Each *CDM* has the same format as in Scheme IV, and each sensor node uses the multiple buffer random selection mechanism to save *CDM*s. The only difference is that this variation may have more than two key chain levels.

Compared with Scheme IV, this variation is not more vulnerable to DOS attacks. The success of the DOS attacks depends on the percentage of forged *CDM*s and the buffer capacity in sensor nodes. As long as the base station maintains a certain authentic *CDM* rate, this variation will not have higher percentage of forged *CDM*s than Scheme IV. The base station can further piggyback the *CDM*s for different levels of key chains so as to reduce the communication cost.

Having more levels of key chains does increase the overhead at both the base station and the sensor nodes. This variation requires the base station to maintain one active key chain at each level. Because of the available resource in typical bases stations, this overhead is usually tolerable. Similarly, sensor nodes have to maintain more buffers for the key chain commitments as well as *CDM*s in different key chain levels. This is usually not desirable because of the resource constraints in sensor nodes. In addition, the more levels we have, the more bandwidth is required to transmit the *CDM*s. Thus, we should use as few levels as possible to cover the lifetime of a sensor network.

Now let us consider the frequency of *CDM*s in DOS-tolerant multilevel μTESLA. To increase the chance to succeed, the attacker may target at a particular key chain level instead of attacking all levels simultaneously. Further assume that the base station sends out the *CDM*s of each key chain level in the same frequency, and the buffer in each sensor node can accommodate m (authentic and/or forged) copies of a *CDM*. Thus, for DOS-tolerant M-level μTESLA, equation (1) can be generalized to

$$R_c \geq \frac{(M-1)(1-R_d)(1-\sqrt[m]{1-P})}{(M-1)(1-\sqrt[m]{1-P})+\sqrt[m]{1-P}}, \tag{3}$$

where R_c is the fraction of bandwidth required for *CDM*s in all key chain levels, and R_d is the fraction of bandwidth used for data packets, m is the number of

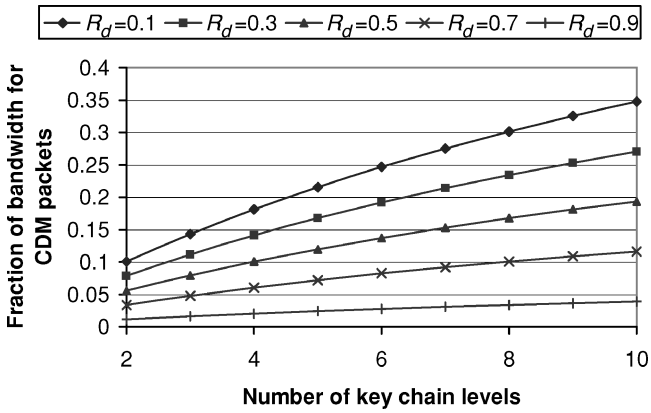


Fig. 6. Bandwidth for *CDM*s versus number of key chain levels. Assume the number of *CDM* buffers in each key chain level is $m = 40$.

CDM buffers in each key chain level, and P is the desired probability that a sensor node has an authenticated key chain commitment before the key chain is used.

We may still use the approach in Section 3.4.4 to determine the frequency of *CDM*s in order to maintain broadcast authentication service when the network is under severe DOS attacks. Figure 6 shows the required fraction of bandwidth for *CDM*s to guarantee that each sensor node has the probability $P = 0.9$ to have an authenticated low-level key chain commitment before the key chain is used. It is easy to see that the addition of more key chain levels does introduce additional communication overhead. Similar to Figure 5, Figure 6 shows smaller fraction of bandwidth required for *CDM*s when the data rate is higher. As discussed earlier, the increase in data rate consumes more bandwidth for data and leaves less bandwidth for forged *CDM*s. As a result, the requirement for *CDM*s is also reduced.

In the following, we give an analysis of the overheads introduced by the DOS-tolerant multilevel μTESLA scheme. For simplicity, we assume there are totally M levels in our scheme and L keys in each key chain. Thus, if the duration of each lowest-level time interval (level $M - 1$) is Δ, the duration of each level i time interval is $\Delta_i = \Delta \times L^{M-i-1}$, and the maximum lifetime of the scheme is $\Delta \times L^M$.

The storage overhead in sensor nodes is mainly due to the buffer of *CDM*s. Each sensor node has to buffer weakly authenticated *CDM*s for the top $M - 1$ levels. Assuming a sensor node uses m *CDM* buffers, this totally requires about $m \cdot (M - 1)$ buffers. (Note that for each *CDM*, only the disclosed key chain commitment and the MAC need to be stored.) In addition, each sensor node needs to store 1 most recently authenticated key for level 0 key chain and 3 most recently authenticated keys for each of the other levels (one for the previous key chain because it is possible that the sensor node receives a packet which discloses a key in the previous key chain, another for the current key chain, and a third for the next key chain). Thus, each sensor node needs to store $3M - 2$ more keys.

A base station only needs to keep the current key chain for each level, which occupies at most $M \times L$ storage space in total. This is because a lower-level key chain can be generated directly from a key in its adjacent upper-level key chain, and the length of key chain in our technique can be short enough to allow computation of a key chain in real time. In contrast, in the original μTESLA scheme [Perrig et al. 2001b], the base station has to precompute and store L^M keys to cover the same period of time as in our scheme.

Consider the communication overhead due to the *CDM*s. In order to mitigate severe DOS attacks, the base station has to use a fair amount of bandwidth to broadcast *CDM*s, as indicated by equation (3). For example, Figure 6 shows that when the fraction of bandwidth for data packets is 0.1, the number of key chain levels is 3, and each sensor node has 40 buffers for each *CDM*, the base station needs about 15% of the bandwidth for *CDM*s.

The computational overhead in sensor nodes is mainly due to the authentication of disclosed keys and MACs. A sensor node's computation for data packets is dependent on the number of data packets the sensor node receives. However, a sensor node's computation for *CDM* packets is bounded by the number m of *CDM* buffers, since the sensor node has at most m copies of each *CDM*, and it can stop once it authenticates a copy.

As discussed earlier, in the original μTESLA protocol, if there is a long delay between the receipts of two data packets, a sensor node has to perform a large number of pseudorandom functions in order to authenticate the key disclosed in the packet. In the worst case, it has to perform about L^M pseudorandom functions if it only receives the first and the last packets. In contrast, with the DOS-tolerant multilevel μTESLA scheme, such a sensor node needs to perform at most $M \times L$ pseudorandom functions. In general, if a sensor node does not receive packets for n_l lowest-level time intervals, the number of pseudorandom functions that it needs to perform in order to authenticate a key received later never exceeds $L \times \log_L(n_l)$.

It appears that the overheads in this scheme, especially the communication overhead and the storage overhead in sensor nodes, are not negligible. In the following, we introduce the second variation of multilevel μTESLA scheme that is more efficient in terms of communication overhead and storage overhead in sensor nodes.

3.6.2 *Variation II: DOS-Resistant Multilevel μTESLA.* The DOS-resistant multilevel μTESLA scheme is extended directly from Scheme V. There are multiple key chain levels, with lower-level key chains generated from keys in the immediately higher-level key chains. There are multiple key chains in all levels except for level 0. Among these levels, only level $M - 1$ is used to authenticate data packets; all the other levels are used to distribute the key chain commitments in the immediately lower-level. Each *CDM* consists of the image of the next *CDM* under a pseudorandom function. In level i, $0 < i < M - 1$, the last *CDM* in an earlier key chain contains the image of the first *CDM* in the immediately next key chain. As a result, the end of a key chain does not interrupt the immediate authentication of later *CDM*s in the same level.

Similar to its two-level counter part, this scheme requires precomputation to generate all the key chains in each level and all the *CDM*s. This computation cost could be prohibitive if the lifetime of a sensor network is very long. However, it may be tolerable for relatively short-lived sensor networks. For example, consider a three-level scheme with 100 keys in each key chain and 100 ms lowest-level time intervals. Such a scheme can cover 10^5 s, which is about 27 h. The precomputation required to initialize the scheme consists of 1,010,100 pseudorandom functions to generate all the key chains, and 10,100 pseudorandom functions to generate all the *CDM*s. Such computation can be finished in several seconds on a modern PC or PDA. Thus, the precomputation can be either performed on base stations directly, or performed on a regular PC and then downloaded to the base station.

The base station does not have to store all these values due to the low cost involved in computing pseudorandom functions. To continue the above example, the base station may simply store the keys for the active key chain of each level and the images of *CDM*s under pseudorandom functions. Assume that both a key and an image of a pseudorandom function takes 8 bytes. Then the base station only needs to save about $8 \times 300 + 8 \times 10,100 \approx 82$ Kbytes.

In general, for a DOS-resistant M-level μTESLA scheme, where each key chain consists of L keys, a base station needs to precompute $L + L^2 + \cdots + L^M = \frac{L^{M+1}-L}{L-1}$ keys and $L + L^2 + \cdots + L^{M-1} = \frac{L^M-L}{L-1}$ *CDM*s, respectively. In addition, the base station needs to store $M \times L$ keys and $\frac{L^M-1}{L-1}$ *CDM* images, respectively. Additional trade-off is possible to reduce the storage requirement (by not saving but computing some *CDM* images when they are needed) if the base station does not have space for all these keys and *CDM* images.

This scheme inherits the advantage of its two-level counter part. That is, a sensor node can get an authenticated key chain commitment as long as it receives one copy of the corresponding *CDM*. As we discussed in Section 3.5, this property substantially reduces the communication overhead introduced by *CDM*s because the base station only needs to send enough copies of a *CDM* to make sure the sensor nodes have a high probability to receive *CDM*s during normal operations, and have a high probability to recover from failures over a period of time when the sensor nodes are under DOS attacks. Specifically, if we would like a sensor node to recover from a failure of receiving a *CDM* within l time intervals (in the same level), by using the same process to get equation (3), we have the following equation:

$$R_c \geq \frac{(M-1)(1-R_d)(1-\sqrt[ml]{1-P})}{(M-1)(1-\sqrt[ml]{1-P})+\sqrt[ml]{1-P}} \tag{4}$$

where R_c is the fraction of bandwidth required for *CDM*s in all key chain levels, and R_d is the fraction of bandwidth used for data packets, m is the number of *CDM* buffers in each key chain level, and P is the desired probability that a sensor node recovers from the failure over the next l time intervals. It is easy to verify that when m and l increase, the right-hand side of equation (4) decreases, and so does the requirement for R_c. Moreover, a sensor node may use dynamic buffer management as discussed in Section 3.5 to arrange buffers for *CDM*s.

Though a *CDM* in this scheme is slightly larger than that in variation I (by one pseudorandom function image per *CDM*), the frequency of *CDM*s can be reduced substantially. Thus, the overall storage requirement in sensor nodes can be much less than that in variation I.

The computational overhead in a sensor node is not as clear as in variation I. In variation I, the number of authentication a sensor node needs to perform is bounded by the number of *CDM* buffers. In contrast, in this scheme, a sensor node may only need to authenticate one copy of *CDM* if the first received message is authentic, but may also have to authenticate every received copy of a *CDM* if no copy is authentic in the worst case.

The limitation of this variation is its scalability. It is easy to see that the precomputation cost is linear to the number of lowest-level time intervals. Consider a long-lived sensor network that requires a 3-level key chains scheme, where each key chain consists of 1000 keys and the duration of each lowest-level time interval is 10 ms. The lifetime of this scheme is 10^7 s, which is about 116 days. Using 3-level key chains implies that the base station needs to precompute about 1,001,001,000 pseudorandom functions to compute the key chains and another 1,001,000 pseudorandom functions to compute the images of *CDM*s. In addition, the base station needs to store about 3000 keys and 1,001,000 images of pseudorandom functions, which take about 8 Mbytes memory. Though this is still feasible for typical PCs and workstations, it may be too expensive for base stations that are not very resourceful.

3.6.3 *Variation III: Hybrid Multilevel μTESLA.* Variation III is essentially a trade-off between the first two variations. To make the techniques in variation II practical for low-end base stations, we reduce the precomputation and storage overheads by sacrificing certain immediate authentication capability. Specifically, we limit the precomputed *CDM*s to the active key chain being used in each level. For a given key chain in a particular level, the base station computes the images of the *CDM*s (under the pseudorandom function H) only when the first key is needed for authentication, and this computation does not go beyond this key chain in this level. As a result, the *CDM* authenticated with the last key in a key chain will not include the image of the next *CDM* in the same level because this information is not available yet. The base station may simply set this field as NULL. For the first key chain in each level i, where $0 \leq i \leq M-1$, the image of the first *CDM* can be distributed during the initialization phase.

The behavior of a sensor node is still very simple. If the sensor node has an authentic image of the next *CDM* in a certain level, it can authenticate the next *CDM* immediately after receiving it. Otherwise, the sensor node simply uses the random selection strategy to buffer the weakly authenticated copies. To increase the chance that the sensor nodes receive an authentic image of the first *CDM* for a key chain, the base station may also broadcast it in data packets.

Such a method reduces the computation and storage requirement significantly compared with variation II. For an M-level μTESLA with L keys in each key chain, the base station only needs to precompute around $M \cdot L$ pseudorandom functions and store $(M-1) \cdot L$ images of *CDM*s. In the earlier example

with 3-level key chains and 1000 keys per key chain, the base station only needs to compute about 3000 (instead of 1,001,001,000 in variation II) pseudo-random operations during initialization and store 2000 (instead of 1,001,000 in variation II) *CDM* images.

An obvious weak point of this multilevel μTESLA scheme is the handover of two consecutive key chains in the same level. Consider two consecutive key chains in level i, where $i < M - 1$. These key chains are used to distribute *CDM*s for the immediately lower-level key chains. For all the keys except for the last one in each key chain, the corresponding *CDM*s include an image of the next *CDM*, which enables a sensor node to authenticate the next *CDM* immediately after receiving it. However, the last *CDM* corresponding to the earlier key chain does not have an image of the first *CDM* corresponding to the later key chain, as discussed earlier. Thus, the first *CDM* of the later key chain cannot be authenticated immediately after it is received, though the commitment of this key chain can be authenticated with the immediately upper-level *CDM*. As a result, a sensor node has to wait for the next *CDM* to disclose the corresponding μTESLA key in order to authenticate the first *CDM*.

An attacker may take advantage of this opportunity to launch DOS attacks. However, this scheme will not perform worse than variation I because each sensor node can always fall back the random selection mechanism to mitigate the impact of such an attack. In addition to the dynamic buffer management discussed in Section 3.5, the base station can also use an adaptive method to determine the frequency of *CDM*s to improve the resistance against DOS attacks without substantially increasing the communication overhead. That is, the base station may use a low frequency to send out *CDM*s corresponding to later intervals in a key chain, and use a high frequency for the early ones. The analysis performed for variation I to decide the desirable frequency of *CDM*s is also applicable to variation III.

Though having less overhead than variation II, variation III introduces more overheads into base stations than variation I. Besides computing a key chain before using it, a base station using this variation has to compute all the corresponding *CDM*s because each earlier *CDM* includes the image of the immediately following *CDM*. The storage overhead in the base station in this scheme is also higher than that in variation I due to the storage of these *CDM*s.

Variation III introduces lower overheads in sensor nodes than variation I, but has higher overheads than variation II. In normal situations when a sensor node has an authenticated image of the following *CDM*, it only needs to save one copy of that *CDM*. A sensor node's computation and storage overheads are the same as in variation II. During the handover of two key chains (in the same level), a sensor node needs to increase the number of *CDM* buffers to mitigate potential DOS attacks. This is similar to variation I. However, unlike in variation I, a sensor node using variation III can recover to the above normal situation once it authenticates one *CDM*. This is essentially the same as recovering from failures (to receive an authentic *CDM*) in variation II. As discussed earlier, the storage overhead in sensor nodes is much smaller than that in variation I when the sensor nodes are allowed to recover over several time intervals. But such overheads in a sensor node using variation III are higher than in variation II

because such recovery processes are "scheduled" in addition to those due to failures.

A sensor node using variation III may use an adaptive approach to save *CDM*s during handover of key chains. Specifically, a sensor node may just save a few (or even a single copy of) of the first *CDM* corresponding to a new key chain. When the next *CDM* arrives, the sensor node can then decide whether there is an on-going DOS attack by attempting to authenticate the earlier *CDM*. If the earlier *CDM* is authenticated, the sensor node can continue to authenticate later *CDM*s with the corresponding image; otherwise, the sensor node can determine that there is a DOS attack and adaptively increase the number of *CDM* buffers.

Consider the communication overhead in variation III introduced by *CDM*s. We can use equation (4) to determine the frequency of *CDM*s given the fraction of bandwidth used by data packets, the number M of key chain levels, the number m of *CDM* buffers in each sensor node, and the probability P that a sensor node recovers from a failure (or get the first authenticated *CDM* for a key chain) over l time intervals. The base station may increase the frequency of the first several *CDM*s in a key chain based on equation (3) to increase their probability to be authenticated by sensor nodes. Thus, the communication overhead in variation III is between those of variation I and variation II.

Among these variations, variation II has a distinctive advantage over the other two variations. Indeed, variation II can substantially reduce the impact of DOS attacks. In order to get an authentic key chain commitment in a *CDM*, a sensor node only needs to receive an authentic copy of this message in most of cases because the sensor node can immediately authenticate it. Though a sensor node has to rely on the random selection mechanism to recover from failures, the cost is much less than those required by variations I and III. The disadvantage of variation II is its precomputation and storage overhead. Thus, if the base station has enough resources, variation II should be used. Variation III sacrifices some immediate authentication capability to reduce the precomputation and storage requirements in variation II. Thus, if the base station has certain, but not enough resources, variation III should be used. If the base station cannot afford the precomputation and storage overheads required by variation III at all, variation I can be used to mitigate the potential DOS attacks.

4. EXPERIMENTAL RESULTS

We have implemented the DOS-tolerant multilevel μTESLA scheme on TinyOS [Hill et al. 2000], which is an operating system for networked sensors. We have performed a series of experiments to evaluate the performance of the DOS-tolerant multilevel μTESLA when there are packet losses and DOS attacks against *CDM*s. The communication, storage, and computation overheads are discussed in earlier sections. The focus of the evaluation in this section is on the overall effectiveness of the proposed techniques (e.g., multibuffer random selection) in tolerating packet losses and DOS attacks, and the impact of different choices of certain parameters (e.g., buffer size, percentage of forged *CDM* packets). The experiments were performed using Nido, the TinyOS simulator.

To simulate the lossy communication channel, we have each sensor node drop each received packet with a given probability.

To further study the performance of the scheme in presence of attacks, we also implemented an attacker component, which listens to the *CDM*s broadcasted by the base station and inserts forged *CDM*s 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 $K_{i+2,0}$ and the MAC value in a *CDM* because any other modification can be detected by a sensor node immediately. There are other attacks against the scheme. Because they are either defeatable by the scheme (e.g., modification of data packets), or not specific to our extension (e.g., DOS attacks against the data packets), we did not consider them in our experiments.

To concentrate on the design decisions we made in our schemes, we fix the following parameters in all the experiments. We only performed the experiments with DOS-tolerant two-level μTESLA because the only purpose of having multiple levels is to scale up to a 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 s. We put 200 keys in the high-level key chain, which covers up to 200 min in time. We also set the data packet 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 lifetime 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. Because our purpose is to study the performance of the scheme with respect to packet losses and DOS attacks, we did not do so in our evaluation.

The performance of our techniques depends on the probability of having an authentic key chain commitment, which is mainly affected by the number of *CDM* buffers in sensor nodes and the percentage of forged *CDM* packets in the communication channel as we discussed before. Thus, in our experiments, we simply fix the *CDM* packet rate but use different attack rates to evaluate the performance of our system.

The performance of our system is evaluated with the following metrics: average percentage of authenticated data packets (i.e., $\frac{\#\text{authenticated data packets}}{\#\text{received data packets}}$ 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*s, 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*s becomes a major concern when we deploy a real sensor network. We evaluate the performance of different memory allocation schemes with a memory constraint. The format of data packet in our proposed technique is the same as in the original μTESLA except for a level number, which only occupies 1 byte. In our implementation, both *CDM* and data packets consist of 29 bytes. The data packet includes a level number (1 bytes),

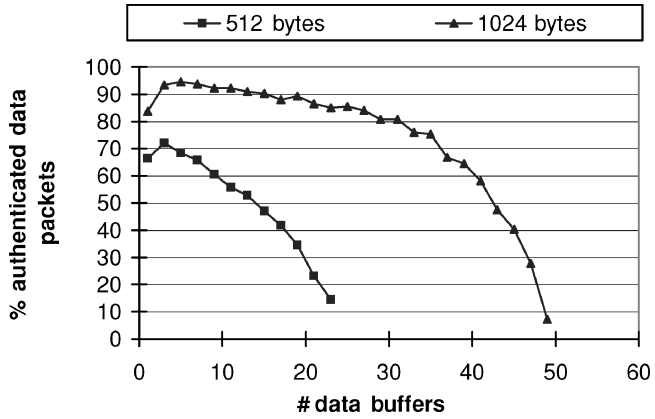


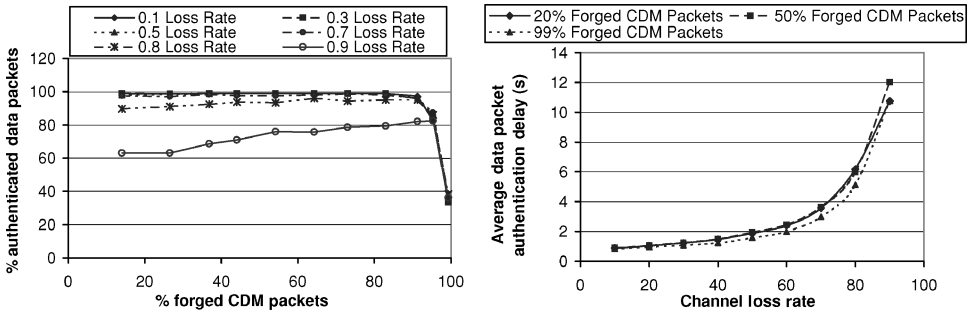
Fig. 7. The performance with different buffer allocation schemes for total memory 512 and 1024 bytes to buffer data and *CDM*s. Assume 95% of *CDM* packets are forged and 50% of packets are lost when transmitted over the channel.

an index (4 bytes), data (8 bytes), MAC (8 bytes), and a disclosed key (8 bytes). A *CDM* packet includes a level number (1 byte), an index (4 bytes), a key chain commitment $K_{i+2,0}$ (8 bytes), a MAC (8 bytes), and a disclosed key (8 bytes).

It is true that our schemes (and μTESLA) have relatively high overhead in data packets with the above settings. This is in some sense because of the small packet size. However, broadcast authentication is usually used to broadcast commands or control data from the base station to sensor nodes. We expect typical commands or control data can fit in the 8 bytes payload. The base station also has the option to split long commands or data into multiple packets. Moreover, it is possible to modify the maximum packet size in TinyOS to decrease the overhead. In our experiment, we only consider the default maximum packet size supported by TinyOS, because the effect of CDM packets is our main concern.

When a sensor node receives a data packet, it does not need to buffer the level number and the disclosed key for future authentication; only the other 20 bytes need to be stored. For *CDM* packets, all copies of the same *CDM* have the same values for the fields other than the key chain commitment and the MAC value (i.e., $K_{i+2,0}$ and MAC in CDM_i) because all forged messages without these values can be filtered out by the weak authentication mechanism. As a result, for all copies of CDM_i, the only fields that need saving are $K_{i+2,0}$ (8 bytes) and MAC (8 bytes), assuming that the level number and the index are used to locate the buffer and the disclosed key K_{i-1} is stored elsewhere to authenticate later disclosed keys. Further, assume the totally available memory for data and *CDM*s is C bytes, and the sensor node decides to store up to x data packets. Then the node can save up to $y = \lfloor \frac{C-20\times x}{16} \rfloor$ copies of *CDM*s.

Figure 7 shows the performance of different memory allocation schemes under severe DOS attacks against *CDM*s (95% forged *CDM* packets). In these experiments, we have total memory of 512 bytes or 1 Kbytes. As shown in Figure 7, three data buffers (60 bytes) are enough to authenticate over 90% of the received data packets when the total memory is 1 Kbytes. This is because



(a) Authenticated data packets (b) Data packet authentication delay

Fig. 8. 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/min).

the data packet arrived in later time interval carries the key that can be used to authenticate the data packets arrived in earlier time intervals. If there are no DOS attacks on data packets (such attacks are not considered in our experiments), the sensor node can authenticate those data packets that arrived no less than d time intervals earlier and remove them from the buffer. Thus, the buffer size for data packets depends on the data rate, the key disclosure lag d and the duration of the lowest key chain time interval. In practice, it only needs to be large enough to hold all data packets within d lowest-level time intervals.

The figure also shows that after a certain point, having more data buffers does not increase the performance. Instead, it decreases the performance because less memory is left for buffering the *CDM*s.

To measure the performance under intensive DOS attacks, we assume that each sensor node can store up to 3 data packets and 39 *CDM* packets, which totally occupy 684 bytes memory space. The experimental results are shown in Figures 8(a) and 8(b). Figure 8(a) shows that our system 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* if all of the *CDM*s 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 8(a) also shows that if the base station rebroadcasts sufficient number of *CDM*s so that on average, at least one copy of such authentic *CDM* can reach a 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 8(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 when the channel loss rate is 90%, the data packet authentication rate initially 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 8(b). The reason is that a sensor node needs to wait a longer time to get the disclosed key. Though the number of dropped packets increases dramatically under sever DOS attack (over 95%) as seen in Figure 8(a), Figure 8(b) shows that the percentage of forged *CDM* does not have a significant impact on the average data packet authentication delay for those packets that have been authenticated.

In summary, the experimental results demonstrate that our system can maintain reasonable performance even with high channel loss rate under severe DOS attacks.

5. RELATED WORK

Security in sensor networks has attracted intensive research efforts recently [e.g., Stajano and Anderson 1999; Carman et al. 2000; Perrig et al. 2001b]. Due to the limited resources at sensor nodes, solutions based on asymmetric cryptography [Gennaro and Rohatgi 1997; Rohatgi 1999; Wong and Lam 1998] are usually impractical for sensor networks. In the following, we restrict our discussion to related techniques based on symmetric cryptography.

One-way hash functions play an important role in our schemes. The use of one-way hash functions for authentication can be traced back to Lamport [1981], which was later implemented as the S/Key one-time password system [Haller 1994]. Cheung proposed OLSV that uses delayed disclosures of keys by the sender to authenticate link-state routing updates between routers [Cheung 1997]. Anderson et al. used the same technique in their Guy Fawkes protocol to authenticate messages between two parties [Anderson et al. 1998]. Briscoe proposed the FLAMeS protocol [Briscoe 2000], and Bergadano et al. presented an authentication protocol for multicast [Bergadano et al. 2000]. Both are similar to the OLSV protocol [Cheung 1997]. Canetti et al. proposed to use k different keys to authenticate the multicast messages with k different MACs for sender authentication [Canetti et al. 1999]. However, their scheme has high communication overhead because of the k MACs for each message. Perrig introduced a verification efficient signature scheme named BiBa based on one-way hash functions without trapdoors [Perrig 2001]; however, BiBa has high overhead in signature generation and public key distribution. These techniques either do not address, or cannot be applied to broadcast authentication in sensor networks.

Our techniques in this paper are closely related to TESLA, a broadcast authentication protocol, which has been described in Section 2. TESLA was originally proposed in Perrig et al. [2000b] to efficiently authenticate multicast streams over lossy channels. Though TESLA requires loose time synchronization between a sender and multiple receivers, it is extremely efficient because it mainly uses symmetric cryptography for authentication. TESLA was later extended to provide additional capabilities such as immediate authentication (to remove the delay between the receipt and the authentication of a data packet) and concurrent TESLA instances (to accommodate heterogeneous networks with different bandwidths) [Perrig et al. 2001a]. TESLA requires a digital

signature operation to bootstrap itself, and thus is impractical in resource constrained sensor networks. As an adaption of TESLA, μTESLA uses symmetric cryptography to distribute initial parameters to the sensor nodes individually [Perrig et al. 2001b]. As discussed earlier, the drawback of this solution is the high communication overhead required for initializing sensor nodes when the number of sensor nodes is large. The work in this paper is to address this problem.

Perrig et al. proposed to use an earlier key chain to distribute the commitments of the next key chain [Perrig et al. 2000a]. Multiple early TESLA packets are used to tolerate packet losses. However, because reliable distribution of later commitment cannot be fully guaranteed, if all the packets used to distribute commitments are lost (e.g., due to temporary network partition), a receiver will not be able to recover the commitment of the later key chain. As a result, the sender and the receivers will have to repeat the costly bootstrap process. In contrast, because of the connection between two consecutive levels of key chains, our techniques allow a receiver to recover the key chains even if all the commitment distribution messages during one high-level time interval are lost.

Besides broadcast authentication, key management is also a fundamental security service in sensor networks. (In some sense, our techniques can also be considered as key management techniques for broadcast authentication.) 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 [Basagni et al. 2001]. 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.

Due to the resource constraints in sensor nodes, several new key management techniques have been proposed recently. A probabilistic key predistribution technique was first proposed in Eschenauer and Gligor [2002]. The basic idea is to let each sensor node randomly pick a set of keys from a key pool so that two sensor nodes can have a certain probability of sharing a common key. Chan et al. improved this idea to a q-composite key predistribution scheme, which requires at least q shared common keys in order to set up a pairwise key [Chan et al. 2003]. Moreover, Chan et al. also investigated a random pairwise keys scheme, which predistributes a unique random pairwise key between a random pair of sensor nodes [Chan et al. 2003]. Liu and Ning developed a framework to predistribute pairwise keys using bivariate polynomials, and two efficient instantiations, a random subset assignment scheme, and a grid-based key predistribution scheme, to establish pairwise keys in sensor networks [Liu and Ning 2003b]. Instead of using a bivariate polynomial, Du et al. proposed another approach based on Blom's key predistribution scheme [Du et al. 2003]. Liu and Ning, Du et al. later independently developed techniques to use sensors' expected locations to improve the performance of pairwise key predistribution [Liu and Ning 2003c; Du et al. 2004]. Zhu et al. proposed a protocol suite named LEAP (Localized Encryption and Authentication Protocol) to help establish individual keys between sensor nodes and a base station, pairwise keys between sensor nodes, cluster keys within a local area, and a group key shared by all

nodes [Zhu et al. 2003]. These techniques address the fundamental problem of secure communication between (or among) sensor nodes. However, they cannot provide broadcast authentication capabilities. Thus, we consider them complementary to our techniques in this paper.

Wood and Stankovic identified a number of DOS attacks in sensor networks [Wood and Stankovic 2002]. Karlof and Wagner analyzed the vulnerabilities and the countermeasures for a number of routing protocols for sensor networks [Karlof and Wagner 2003]. The broadcast authentication techniques proposed in this paper may help address some attacks identified in these papers.

6. CONCLUSION AND FUTURE WORK

In this paper, we developed a multilevel key chain scheme to efficiently distribute the key chain commitments for the broadcast authentication scheme named μTESLA. By using predetermination and broadcast, our approach 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. The resulting protocol, named multilevel μTESLA, satisfies several nice properties, including low overhead, tolerance of message loss, scalability to large networks, and resistance to replay attacks as well as DOS attacks.

Several problems are worth further investigation. First, the authentication delay and the failure recovery delay are still not fully solved. For example, when a sensor node does not get a key chain commitment during a time interval, it must wait for a relatively long period of time to recover from this failure. We will seek solutions to this problem in our future research. Second, the assumption of loose time synchronization in sensor networks may not be true in some applications; there are many ways to disrupt the time synchronization. Thus, it may be desirable to have alternative approaches to authenticating broadcast messages without the assumption of time synchronization. Third, in this paper, we assumed a single base station in a sensor network, which is assumed to be well protected. However, in some scenarios, there may exist multiple base stations and one or some of them may be compromised by attackers. We would like to study broadcast authentication mechanisms that can support multiple base stations more efficiently and that can tolerate compromised base stations.

APPENDIX

A. A DETAILED DESCRIPTION OF SCHEME IV

A.1 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_0} 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 $i = 0, 1, \ldots, n_0 - 1$, and F_0 is a pseudorandom 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 maximum clock discrepancy $\delta_{\max}$ during the lifetime of the sensor network; (8) the frequency of *CDM* packets.

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) $\delta_{\max}$. 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. In addition, if the base station wants to enable the sensors to broadcast authenticated messages during the high-level time intervals I_1 and I_2, the base station needs to distribute $K_{1,0}$ and $K_{2,0}$ to the sensors.

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.

A.2 Broadcast of CDMs

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 [Perrig et al. 2000b, 2001a, 2001b]. Assume the base station decides to divide each time interval I_i into n_1 smaller intervals, denoted $I_{i,1}, I_{i,2}, \ldots, I_{i,n_1}$. The base station generates the low-level key chain by computing $K_{i,n_1} = F_{01}(K_{i+1})$, and $K_{i,j} = F_1(K_{i,j+1})$, where $j = 0, 1, \ldots, n_1 - 1$ and F_1 is a pseudorandom 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-2} during the time interval I_{i-2}.

Each CDM_i contains the index of the high-level time interval, the commitment of the low-level key chain $\langle K_{i+2,0} \rangle$, the MAC generated over the above fields with the key K'_i, which is derived from the high-level key K_i, and the disclosed high-level authentication key K_{i-1}.

$$\textit{Base Station} \rightarrow \textit{Sensors} : CDM_i = i | K_{i+2,0} | MAC_{K'_i}(i | K_{i+2,0}) | K_{i-1}.$$

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

A.3 Authentication of CDMs

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, that is, $t_i + \delta_{\max} < T_{i+1}$. S discards the packet and stops if the security condition is not satisfied.

(2) 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 because 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}.

(3) 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}. The key chain commitment $K_{i+1,0}$ contained in this copy of CDM_{i-1} is then selected as the commitment of the low-level key chain $\langle K_{i+1,0} \rangle$ for the next high-level time interval I_{i+1}.

(4) S uses the random selection strategy discussed in Section 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).

A.4 Broadcast and Authentication of Normal Messages

Broadcast and authentication of normal messages are performed in the same way as in the extended TESLA [Perrig et al. 2001a], except for the distribution of the key chain commitments, which is handled in the distribution and authentication of commitment distribution messages.

ACKNOWLEDGMENTS

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

REFERENCES

ANDERSON, R., BERGADANO, F., CRISPO, B., LEE, J., MANIFAVAS, C., AND NEEDHAM, R. 1998. A new family of authentication protocols. In *Operating Systems Review*.

BASAGNI, S., HERRIN, K., BRUSCHI, D., AND ROSTI, E. 2001. Secure pebblenets. In *Proceedings of ACM International Symposium on Mobile ad hoc networking and computing*. 156–163.

BERGADANO, F., CAVAGNINO, D., AND CRISPO, B. 2000. Individual single source authentication on the mbone. In *IEEE International Conference on Multimedia & Expo (ICME)*.

BRISCOE, B. 2000. FLAMeS: Fast, loss-tolerant authentication of multicast stream. Tech. rep., BT Research.

CANETTI, R., GARAY, J., ITKIS, G., MICCIANCIO, D., NAOR, M., AND PINKAS, B. 1999. Multicast security: A taxonomy and some efficient constructions. In *Proceedings of IEEE INFOCOM '99*. 708–716.

CARMAN, D., KRUUS, P., AND MATT, B. J. 2000. Constrains and approaches for distributed sensor network security. Tech. rep., NAI Labs.

CHAN, H., PERRIG, A., AND SONG, D. 2003. Random key predistribution schemes for sensor networks. In *IEEE Symposium on Research in Security and Privacy*. 197–213.

CHEUNG, S. 1997. An efficient message authentication scheme for link state routing. In *13th Annual Computer Security Applications conference*. San Diego, California.

DU, W., DENG, J., HAN, Y. S., CHEN, S., AND VARSHNEY, P. 2004. A key management scheme for wireless sensor networks using deployment knowledge. In *IEEE INFOCOM '04 (to appear)*.

DU, W., DENG, J., HAN, Y. S., AND VARSHNEY, P. 2003. A pairwise key pre-distribution scheme for wireless sensor networks. In *Proceedings of 10th ACM Conference on Computer and Communications Security (CCS'03)*. 42–51.

ESCHENAUER, L. AND GLIGOR, V. D. 2002. A key-management scheme for distributed sensor networks. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*. 41–47.

GENNARO, R. AND ROHATGI, P. 1997. How to sign digital streams. Tech. rep., IBM T.J. Watson Research Center.

GOLDREICH, O., GOLDWASSER, S., AND MICALI, S. 1986. How to construct random functions. *J. ACM 33,* 4 (October), 792–807.

HALLER, N. M. 1994. The S/KEY one-time password system. In *Proceedings of the ISOC Symposium on Network and Distributed System Security*. 151–157.

HILL, J., SZEWCZYK, R., WOO, A., HOLLAR, S., CULLER, D., AND PISTER, K. S. J. 2000. System architecture directions for networked sensors. In *Architectural Support for Programming Languages and Operating Systems*. 93–104.

KARLOF, C. AND WAGNER, D. 2003. Secure routing in wireless sensor networks: Attacks and countermeasures. In *First IEEE International Workshop on Sensor Network Protocols and Applications*.

KRAWCZYK, H., BELLARE, M., AND CANETTI, R. 1997. HMAC: Keyed-hashing for message authentication. IETF RFC 2104.

LAMPORT, L. 1981. Password authentication with insecure communication. *Commun. ACM 24,* 11, 770–772.

LIU, D. AND NING, P. 2003a. 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*. 263–276.

LIU, D. AND NING, P. 2003b. Establishing pairwise keys in distributed sensor networks. In *Proceedings of 10th ACM Conference on Computer and Communications Security (CCS'03)*. 52–61.

LIU, D. AND NING, P. 2003c. Location-based pairwise key establishments for static sensor networks. In *2003 ACM Workshop on Security in Ad Hoc and Sensor Networks (SASN '03)*.

PERRIG, A. 2001. The BiBa one-time signature and broadcast authentication protocol. In *Proceedings of the ACM Conference on Computer and Communications Security*. 28–37.

PERRIG, A., CANETTI, R., BRISCOE, TYGAR, J., AND SONG, D. 2000a. TESLA: Multicast source authentication transform. IRTF draft, draft-irtf-smug-tesla-00.txt.

PERRIG, A., CANETTI, R., SONG, D., AND TYGAR, D. 2000b. Efficient authentication and signing of multicast streams over lossy channels. In *Proceedings of the 2000 IEEE Symposium on Security and Privacy*.

PERRIG, A., CANETTI, R., SONG, D., AND TYGAR, D. 2001a. Efficient and secure source authentication for multicast. In *Proceedings of Network and Distributed System Security Symposium*.

PERRIG, A., SZEWCZYK, R., WEN, V., CULLER, D., AND TYGAR, D. 2001b. SPINS: Security protocols for sensor networks. In *Proceedings of Seventh Annual International Conference on Mobile Computing and Networks*.

RIVEST, R., SHAMIR, A., AND ADLEMAN, L. 1978. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM 21,* 2, 120–126.

ROHATGI, P. 1999. A compact and fast hybrid signature scheme for multicast packet authentication. In *6th ACM Conference on Computer and Communications Security*.

STAJANO, F. AND ANDERSON, R. 1999. The resurrecting duckling: Security issues for ad hoc networks. In *Proceedings of the 7th International Workshop on Security Protocols*. 172–194.

WONG, C. AND LAM, S. S. 1998. Digital signatures for flows and multicasts. In *Proceedings of IEEE ICNP'98*.

WOOD, A. D. AND STANKOVIC, J. A. 2002. Denial of service in sensor networks. *IEEE Comput. 35,* 10, 54–62.

ZHU, S., SETIA, S., AND JAJODIA, S. 2003. LEAP: Efficient security mechanisms for large-scale distributed sensor networks. In *Proceedings of 10th ACM Conference on Computer and Communications Security (CCS'03)*. 62–72.

Received March 2003; revised September 2003 and March 2004; accepted April 2004