Dynamic Merkle Trees for Verifying Privileges in Sensor Networks

Li Zhou and Chinya V. Ravishankar
Department of Computer Science & Engineering
University of California, Riverside
Riverside, CA 92521, USA
{lzhou,ravi}@cs.ucr.edu

***Abstract*— Mobile sinks are already used in static sensor networks to facilitate data collection or network management, but it is possible in such networks for mobile sinks to be compromised and mount various insider attacks. Some schemes have been devised recently to limit the privileges of mobile sinks appropriately, but these are secure only when the number of compromised mobile sinks or sensors is below a network-wide threshold. This threshold is determined by a sensor's memory, and is usually too small.**

We introduce dynamic Merkle trees, and show how to use them to restrict and verify the privilege of mobile sinks. Unlike current schemes, our scheme allows all sensors in a region to simultaneously verify mobile sink privileges, and does not assume a network-wide limit on the number of node compromises. As our security analysis shows, our approach is safe from fabrication attacks, impersonation attacks and replay attacks. Further, our performance analysis shows that we incur very low communication, computation, and storage overheads.

I. Introduction

Sensor networks are widely used in applications such as traffic and wildlife monitoring, and in battlefields. In such networks, sensors may be deployed in large numbers and over large areas. Researchers [23], [24], [9], [21], [25] have recently suggested using mobile sinks (MS) within a static sensor network to facilitate data collection or network management. Since sensing regions may be large or far from base stations, sending data directly to the base station will waste energy at intermediate sensors, increase delays, and render transmitted data liable to manipulation en-route. Similar difficulties arise when the base station must notify remote sensors to revoke compromised sensors or send other commands.

Mobile sinks may be used as proxies for base stations, querying sensors or forwarding commands from the base station. Since these represent sensitive or privileged operations, mobile sinks are attractive targets for adversaries, who may compromise mobile sinks to mount insider attacks. Granting mobile sinks full privileges is hence a dangerous practice.

Zhang et al. [25] have proposed several schemes to limit the privileges of mobile sinks to mitigate the dangers from compromised mobile sinks. Their schemes, however, break down completely when the number of system-wide compromises of sensors and mobile sinks exceeds a threshold t. The value t is a network-wide threshold determined by a sensor's memory, and can be as low as 200. This value is too low, since sensor deployments can consist of tens of thousands of nodes. Further, privileges must be negotiated with each sensor separately, introducing large delays. These schemes may not be suitable for applications requiring real-time response.

A. Our Work

In our model, mobile sinks are granted just enough privileges by the base station to carry out their assigned activities, and sensors verify privileges presented by mobile sinks before responding to their activity requests. The base station uses one-way hash chains to create and grant privileges to the mobile sinks, and sensors verify these privileges by constructing dynamic Merkle hash trees. As our security analysis shows, our scheme is secure against fabrication attacks, impersonation attacks and simple replay attacks. We also propose an improved scheme to reduce the time window during which adversaries may mount more complicated replay attacks. Our schemes have the following salient features:

- *Fault-tolerance*: Our schemes are resilient against the compromise of mobile sinks or regular sensors, and are not subject to the threshold limitation in [25].
- *Verification in parallel*: All sensors in a region may perform privilege verification for a mobile sink at the same time. This eliminates the verification delay which is inevitable in the serial schemes of [25].
- *Resilience against fabrication attacks, impersonation attacks and replay attacks.*
- *Low communication and computation overheads.*
- *Low storage requirement.*

The rest of this paper is organized as follows. We describe related work in Section II, and motivate our work in Section III. In Section IV, we present our scheme, and analyze its security and performance in Section VI and Section VII, respectively. Section VIII concludes the paper.

II. Related Work

Perrig et al. originally proposed μTESLA [17] for base stations to broadcast authentication information. Liu et al. [12] have proposed several multi-level μTESLA schemes to improve its capabilities. These schemes provide source authentication only , and can not be used to restrict the privilege of mobile sinks if adopted directly [25]. Further, μTESLA delays

authentication, allowing adversaries to mount denial-of-service attacks (see Section III-A).

Zhang et al. [25] first addressed the problem of handling mobile sink compromises, and presented several designs based on the scheme of Blundo et al. [1]. However, the schemes in [25] work well only below a threshold. When the number of compromised nodes exceeds this threshold, the schemes break down completely. (See Section III-B.2.)

Our scheme uses one-way hash chains [11] and Merkle hash trees [16]. The Merkle hash tree is widely used for authentication [18], [6], [14]. Our schemes, in fact, construct a set of dynamic Merkle trees, and each sensor is able to verify the privilege of the mobile sink by independently updating the root of the tree corresponding to its region.

A number of key pre-distribution schemes have been proposed to establish a pairwise key for any pair of neighboring sensors [7], [3], [5], [13], [4], [2], [26]. Our schemes can be combined with such schemes to further improve the security of the sensor network.

III. Motivation

In this section, we describe two main schemes for packet authentication: μTESLA [17] and Blundo-based Scheme [25], analyze their drawbacks and motivate our work.

A. μTESLA & DOS attacks

The μTESLA scheme [17] authenticates messages broadcast by the base station, and operates as follows. The base station maintains a one-way key chain $K_0, \ldots, K_i, K_{i+1}, \ldots$, with K_0 being the commitment. The network lifetime is divided into time intervals, and packets sent during interval i are all signed with a Message Authentication Code (MAC) generated using the key K_i. The key K_i is disclosed at the start of time interval $(i + \delta)$, so that packets sent in any interval can only be authenticated δ intervals later.

This delay makes μTESLA susceptible to Denial of Service (DoS) attacks. Adversaries may use the delay of authentication in μTESLA to send a large volume of fake packets to overflow buffers at the receivers.

In contrast, our schemes enable sensors to verify the activity request sent by the mobile sink immediately, and thus resilient against DoS attacks.

B. Blundo-based Schemes & Threshold Limitation

1) Blundo-based Schemes: Zhang et al. [25] proposed several Blundo scheme-based protocols to grant mobile sinks the least privilege required to accomplish their activities. The central idea of these schemes is to allow a mobile sink and each sensor u it must work with to establish an activity-dependent pairwise key. When the mobile sink (MS) sends the sensor u a request, u will compute this pairwise key based on the parameters of the specified activity, such as the ID of the MS, the type of the transaction, and the region where activity is to be carried out. If the sensor can successfully communicate with the mobile sink with this activity-dependent pairwise key, it is assumed that the MS does have the privilege to execute the specified activity.

To establish an activity-dependent pairwise key, they use the Blundo scheme [1] to construct their protocols. Their protocols are constructed as follows:

- The base station chooses a random symmetric bivariate polynomial $f(x, y)$ of degree t.

$$f(x,y) = \sum_{0 \le i,j \le t} a_{ij} x^i y^j,$$

  where the coefficients a_{ij} are over a finite field $GF(q)$ and q is a prime number large enough to accommodate a symmetric key.
- The base station preloads each sensor u with a polynomial share $f(u, y)$, which is derived by computing $f(x, y)$ at $x = u$.
- When the base station dispatches a mobile sink to carry out an activity A_i, it assigns the mobile sink with an activity-dependent ID m_i and then loads it with a polynomial share $f(m_i, y)$, which is derived by computing $f(x, y)$ at $x = m_i$.
- The mobile sink sends an activity request to sensors in the specified region. Based on the parameters of the specified activity, a sensor u in the specified region obtains the activity-dependent ID m_i of the mobile sink, and computes the pairwise key shared with m_i by evaluating m_i in its polynomial share. At the same time, the mobile sink evaluates u in its own polynomial share. A match of the pairwise keys $f(u, m_i)$ and $f(m_i, u)$ proves that m_i has the privilege to carry out the claimed activity.

2) Threshold Limitation: The Blundo-based schemes inherit the t-secure property that the system is secure only if no more than t polynomial shares are compromised, where t is the degree of the polynomial being used, and is a network-wide threshold. Essentially, if nodes (mobile sinks or regular sensors) that together hold more than t polynomial shares collude and share their polynomials, they can discover the bivariate polynomial $f(x, y)$ and then compromise the entire system. To improve the resilience of this scheme, the base station must choose a bivariate polynomial with large t. However, $t + 1$, the number of coefficients, is limited by a regular sensor's memory. A coefficient is typically the same size as the key, which is 8 bytes. A Mica2 Mote sensor [20] with 4KB SRAM can store no more than a few hundred coefficients, so that the threshold t is limited to no more than a few hundred. This threshold is clearly too small for a sensor network which may be deployed with tens of thousands of sensors.

In practice, the threshold limitation is even worse for Blundo-based schemes. Each mobile sink is loaded with *multiple* activity-dependent polynomial shares, each of which corresponds to a valid activity. Thus an adversary may collect more than t polynomial shares by compromising just a few mobile sinks. Another option for adversaries to break the threshold t is to compromise more than t tiny sensors, since

Notation	Description
$\mathcal{G}$	1-way hash function for generating credential chain seeds
$\mathcal{F}$	a 1-way hash function for generating credential chains
C_α	the credential chain corresponding to activity α
$C_\alpha(p)$	pth hash value on credential chain C_α
T_r	the time chain corresponding to region r
$T_r(q)$	qth hash value on time chain T_r

TABLE I
OUR NOTATION

sensors have far fewer resources than mobile sinks, and are thus more easily compromised.

Once an adversary compromises more than t polynomial shares and discovers the bivariate polynomial $f(x, y)$, it can fabricate any "valid" activity such as collecting confidential sensing data from sensors in entire detecting region, revoking any good sensor or breaking down the entire network.

IV. OUR SCHEME: PRELIMINARIES

In this paper, we focus on how to grant and verify privileges for mobile sinks to perform transactions. Revoking the privileges of compromised mobile sinks is an orthogonal issue, which has been considered in [25].

A. Assumptions

We assume that sensors have resource limitations typical of the current generation of sensors, such as MICA2 motes [20]. They have only 4KB of SRAM, so we assume that they can store a few hundred keys. The mobile sinks may either be resource-rich class devices, or be resource-limited sensors as in [19], [15]. We also assume that once a sensor or a mobile sink is compromised, all keys stored at it can be accessed by attackers. Further, compromised sensors or mobile sinks are assumed to be able to collude in mounting attacks in various forms. The base station, on the other hand, has sufficient resources to protect itself against compromise.

We make the standard assumption [25] that the monitoring area is divided into regions. Each sensor knows its region [22], but may not know its exact location. Further, as in [17], [14], we assume that clocks on sensors are loosely synchronized. We use the notation in Table I.

B. Definitions and Problem Setup

We use the term *transaction* for the actions that a mobile sink may carry out in a sensor region, such as data collection, network management, the relaying of commands, and so on. Let M be the set of mobile sinks, $\mathcal{T}$ be the set of transaction types, and R be the set of sensor regions.

Definition 1: An *activity* is a triple $\langle ms, tt, r\rangle$, where $ms \in M$, $tt \in \mathcal{T}$ and $r \in R$. It specifies that *ms* will carry out a transaction of type *tt* in region r.

We will refer to an activity either by listing the elements of its tuple, or as we will often find convenient, by using the *activity identifier* $\alpha = \mathcal{H}(ms, tt, r)$, where $\mathcal{H}$ is a collision-resistant hash function. We refer to the components of the activity tuple as $\alpha.ms$, $\alpha.tt$ and $\alpha.r$, respectively.

Definition 2: An *credential chain* C_α corresponding to an activity α is a one-way hash chain whose elements are derived from the activity identifier α (see Section IV-D). The value $C_\alpha(k)$ is the kth value on this chain.

C. Overview

The base station grants mobile sink $\alpha.ms$ a *credential* allowing it to execute transaction $\alpha.tt$ in region $\alpha.r$ by loading it with the next available hash value $C_\alpha(p)$ on C_α. The base station then dispatches $\alpha.ms$ to the region $\alpha.r$.

Upon arrival in region $\alpha.r$, the sink $\alpha.ms$ presents credential $C_\alpha(p)$ and attempts to execute transaction $\alpha.tt$. Sensors in region $\alpha.r$ are required to assist $\alpha.ms$ in the transaction $\alpha.tt$. A sensor acknowledges the right of $\alpha.ms$ to execute transaction $\alpha.tt$ upon verifying $C_\alpha(p)$ to be the next available value on C_α.

If the activity α is to be executed several times, we require $\alpha.ms$ to present a fresh credential corresponding to each execution of α. Consequently, a compromised mobile sink $\alpha.ms$ can not reuse old credentials to claim fresh privileges.

Credential freshness is verified as follows. For each region $r \in R$, the base station first identifies all activities α such that $\alpha.r = r$. It then constructs a Merkle hash tree over the commitments for all credential chains C_α. The root of the Merkle hash tree is loaded into all sensors in region r. We show in Section V how all sensors in region r can verify credential freshness using this root. Unlike uses of Merkle hash trees [6], [14], we must construct Merkle trees *dynamically*.

D. Credential Chain Generation

Let K_M be a master key known only to the base station and $\mathcal{G}$ be a collision-resistant one-way hash function. For each activity $\alpha = \langle ms, tt, r\rangle$, the base station first computes a hash chain seed $C_\alpha(n) = \mathcal{G}(K_M, \alpha)$. Next, it uses another one-way hash function $\mathcal{F}$ to create a credential chain, which is a one-way hash chain $C_\alpha = C_\alpha(0), C_\alpha(1), \cdots, C_\alpha(n)$

$$C_\alpha(j) = \mathcal{F}(C_\alpha(j+1)),\ j = n-1, \cdots, 1, 0. \qquad (1)$$

To save memory, the base station may choose to store parts of this chain, although it is typically assumed that the base station has ample storage. For example, it may only store every kth hash value, and derive the rest using the function $\mathcal{F}$. This is a trade-off between memory and computation.

The values in this chain are used in the order $C_\alpha(1), C_\alpha(2), \cdots, C_\alpha(n)$. The value $C_\alpha(0)$ is the commitment for the hash chain, and is made public. A sensor can now verify that any value $C_\alpha(k)$ belongs to chain C_α by checking whether $C_\alpha(0) = \mathcal{F}^k(C_\alpha(k))$.

E. Issues in Validating Credentials

When $\alpha.ms$ arrives at the region $\alpha.r$, it presents credential $C_\alpha(p)$ to all the sensors in $\alpha.r$. Each sensor must now verify whether $C_\alpha(p)$ is a *fresh* hash value from the chain C_α. The sensor proceeds by verifying first, that $C_\alpha(p)$ belongs to credential chain C_α, and second, that $C_\alpha(p)$ is the next fresh value from C_α.

The first step would be easy if the commitments $C_\alpha(0)$ and the numbers of hash values disclosed from all chains

C_α were stored at the sensor. However, sensor memory is limited, and there may be a very large set of activities, so this is unrealistic. Many mobile sinks may perform the same function. The number of activities m is the product of the number of transaction types and the number of mobile sinks, so with ten transaction types and a hundred mobile sinks, we have a thousand commitments. Sensors would need around 8KB to hold these commitments, which are typically the same size as keys (8 bytes). Current sensors, such as the Mica2 Mote [20] have only about 4KB SRAM.

V. Dynamic Credential Trees

We propose a novel approach for credential validation by defining a dynamic variant of static Merkle trees [16]. For each region $r \in R$, the base station first identifies all activities α_i such that $\alpha_i.r = r$. Then it creates a Merkle tree with leaves corresponding to the α_i as follows.

N_i, the leaf node corresponding to activity α_i, is loaded with a value obtained by hashing the identifier α_i and the commitment $C_{\alpha_i}(0)$ for the chain C_{α_i}. A Merkle tree is now constructed bottom-up in the usual fashion. We denote the leaf node corresponding to α_i by N_i, and the value that N_i holds by $\widehat{N}_i$.

These Merkle trees are maintained dynamically by the base station, and are referred to as *Dynamic Credential Trees* (DC-trees). We will show how a sensor in region r can check whether $C_{\alpha_i}(p)$ appears on credential chain C_{α_i} using only the value at the root of the DC-tree. We also show how each sensor can independently update its DC-tree root value after verifying a credential.

1) Initialization: A DC-tree may have up to $m = |M| \times |\mathcal{T}|$ leaf nodes, where $|M|$ is the number of mobile sinks, and $|\mathcal{T}|$ is the number of transaction types. Leaf node N_i in the DC-tree is initially given the contents $\widehat{N}_i = \mathcal{H}(C_{\alpha_i}(0), \alpha_i)$. Figure 1 shows a DC-tree with 8 leaf nodes.

After sensor deployment, the base station will broadcast the roots of all the DC-trees using μTESLA [17]. Each sensor stores the root of the DC-tree corresponding to its region. Over time, as mobile sinks visit a region r, they will present successive values from their respective credential chains.

Definition 3: The *index* of a credential chain is the number of hash values, excluding the commitment, that have been disclosed from it.

Definition 4: A DC-tree has *configuration* $\kappa = s_1, \ldots, s_m$ if the index of the credential chain C_{α_i} is s_i.

After initialization, the index of each credential chain C_{α_i} is 0, and the DC-tree's initial configuration $\kappa = 0, 0, \ldots, 0$.

2) Credential Authentication: Let the DC-tree τ_r for region r have configuration $\kappa(\tau_r) = s_1, \ldots, s_i, \ldots, s_m$, and let τ_r's root value be ρ. Let $C_{\alpha_i}(s_i + 1)$ be a new credential.

For each credential $C_{\alpha_i}(s_i+1)$, a mobile sink also presents a *proof* $\mathcal{P}(N_i)$, which it gets from the base station. This proof comprises all the hash values in τ_r for the siblings of the nodes on the path from N_i to the root. For example, in Figure 1, $\{\widehat{N}_5, \widehat{N}_{78}, \widehat{N}_{14}\}$ is the proof for a credential on C_{α_6}.

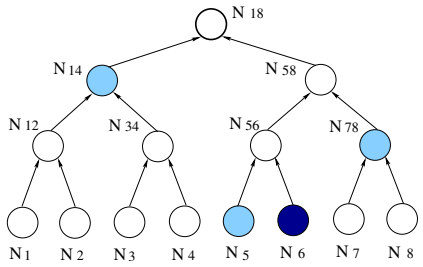

Fig. 1. The initial DC-tree constructed with commitments for 8 credential chains. The values at N_6 and N_{56} are $\mathcal{H}(C_{\alpha_6}(0), \alpha_6)$ and $\mathcal{H}(\widehat{N}_5, \widehat{N}_6)$, respectively. The proof $\mathcal{P}(N_6) = \{\widehat{N}_5, \widehat{N}_{78}, \widehat{N}_{14}\}$.

A sensor verifies $C_{\alpha_i}(s_i + 1)$ indirectly by verifying that the previous value $C_{\alpha_i}(s_i)$ on C_α was used in computing ρ, the current root value. It can obtain $C_{\alpha_i}(s_i)$ from Equation 1, compute the contents $\widehat{N}_i$ of N_i, and verify that this value leads to value ρ under the proof $\mathcal{P}(N_i)$.

When mobile sink $\alpha_i.ms$ moves to region $\alpha_i.r$, it broadcasts a request of the form $(\alpha_i.ms, \alpha_i.tt, \alpha_i.r, C_{\alpha_i}(s_i + 1), \mathcal{P}(N_i))$. Upon receiving this request, a sensor u verifies whether $C_{\alpha_i}(s_i+1)$ is the next available value on the credential chain C_{α_i} in the following steps:

1) Compute activity identifier $\alpha_i = \mathcal{H}(\alpha_i.ms, \alpha_i.tt, \alpha_i.r)$.
2) Compute $\widehat{N}_i = \mathcal{H}(\mathcal{F}(C_{\alpha_i}(s_i + 1)), \alpha_i)$, and a predicted root value ρ' using the proof $\mathcal{P}(N_i)$.
3) If the actual root value at u is ρ, and $\rho' = \rho$, then $C_{\alpha_i}(s_i+1)$ is validated, and sensor u will assist $\alpha_i.ms$ in transaction $\alpha_i.tt$. Otherwise, u will generate a report of mobile sink compromise and send it to the base station.

In Figure 1, let a mobile sink wish to carry out activity α_6. It broadcasts a request which includes α_6, $C_{\alpha_6}(1)$, and the proof $\{\widehat{N}_5, \widehat{N}_{78}, \widehat{N}_{14}\}$. Each sensor within region $\alpha_6.r$ computes $\widehat{N}_6 = \mathcal{H}(\mathcal{F}(C_{\alpha_6}(1)), \alpha_6)$, $\widehat{N}_{56} = \mathcal{H}(\widehat{N}_5, \widehat{N}_6)$, then $\widehat{N}_{58} = \mathcal{H}(\widehat{N}_{56}, \widehat{N}_{78})$, and finally $\rho' = \mathcal{H}(\widehat{N}_{14}, \widehat{N}_{58})$. The sensor accepts the credential if $\rho' = \rho$, the current root value held at the sensor.

To ensure confidentiality, our scheme requires sensors in region $\alpha_i.r$ to encrypt responses with pairwise keys shared with the mobile sink. A large literature exists on pairwise key establishment [7], [3], [5], [13], [4], [2], [26].

3) DC-Tree Updating: If credential validation succeeds, each sensor updates the root value ρ it holds using the newly received credential and the corresponding proof, so that it can verify the next credential to be received. After $C_{\alpha_i}(s_i + 1)$ is verified, each sensor u obtains $\widehat{N}_i = \mathcal{H}(C_{\alpha_i}(s_i + 1), \alpha_i)$, and then computes the new root using $\mathcal{P}(N_i)$. This new root value will be used by u for the verification in the future.

In Figure 2, suppose $C_{\alpha_6}(1)$ has just been verified. All sensors in the region will compute the new root by first computing $\widehat{N}_6 = \mathcal{H}(C_{\alpha_6}(1), \alpha_6)$, and regenerating the other nodes as in the example above. At the same time, the base station will update the DC-tree, updating the nodes $\{N_6, N_{56}, N_{58}, N_{18}\}$.

In our scheme each sensor computes and holds only the root value of the updated DC-tree, while the base station updates and maintains the DC-tree. Figure 2 shows how the values at N_6, N_{56}, N_{58}, and the root are updated.

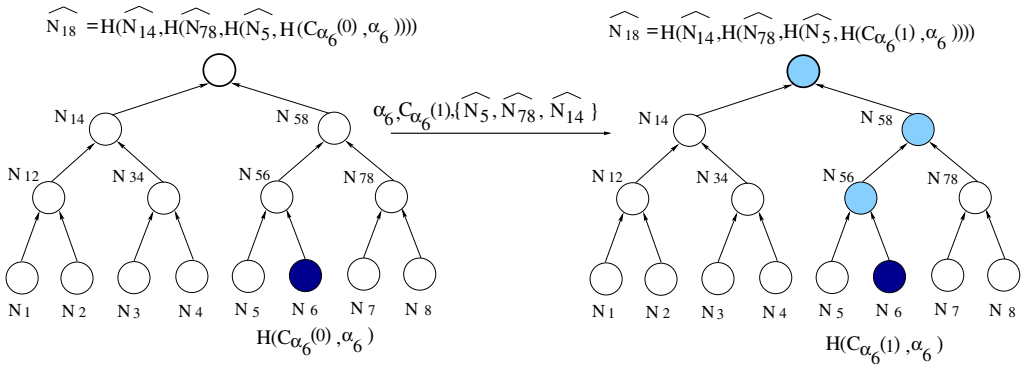

Fig. 2. DC-tree updating. An activity request of α_6 with $C_{\alpha_6}(1)$ and proof $\{\widehat{N}_5, \widehat{N}_{78}, \widehat{N}_{14}\}$ has been verified, and then the nodes $\{\widehat{N}_6, \widehat{N}_{56}, \widehat{N}_{58}, \widehat{N}_{18}\}$ have been updated. Each sensor only maintains the updated root value $\widehat{N}_{18}$. The entire DC-tree is updated and maintained by the base station.

Liu et al. [14] also propose to use Merkle hash tree to distribute and authenticate the commitments of one-way hash chains. However, they use static Merkle trees, and can support only one-time verification of commitments, but not any further credentials disclosed over time. In contrast, our DC-tree scheme allows sensors to verify credentials over time. To the best of our knowledge, the DC-tree is the first dynamic Merkle hash tree structure.

4) Some Additional Issues: We have chosen not to include the transaction time in the definition of activity, since including the time would greatly increase the number of activities m. The DC-tree also has m leaves, so the size $\log m$ of proofs would also increase correspondingly, significantly increasing the communication overhead. If time were divided into 10-minute segments, we would have 144 intervals per day, and the size of a proof would increase by $8\lceil \log_2 144 \rceil = 64$ bytes. Since sensors typically use short-length messages (29-byte payload in TinyOS [20]), this would significantly increase the communication overhead, which is a dominant source of power consumption for sensors. Instead, we choose to assign a new credential for each execution of a same activity.

Problems can arise since sensor networks are prone to packet loss. If a sensor misses a credential or a proof, it will be unable to update the stored root of the DC-tree, and will be unable to verify future credentials. We address this problem by assuming that at most t successive requests can be lost. The base station loads the mobile sink with t previous credentials and proofs. When a sensor is unable to verify a credential due to packet loss, the mobile sink can make the earlier credentials and proofs available to this sensor.

A proof for a Merkle tree with m leaves includes $\log m$ hash values, each of which is typically 8 bytes. These values may be transmitted in multiple packets since sensor networks use small packets. Since a sensor must receive all packets transmitted before verification, an adversary may mount DoS attacks by sending fake hash values. We adopt the scheme in [10] to securely transmit the proof, so that each packet of proof can be verified immediately, preventing DoS attacks. Due to page limits, we refer readers to [10] for details.

VI. Security Analysis

We now show that our scheme is secure against fabrication attacks, impersonation attacks and simple replay attacks. We propose an improved scheme in Section VI-C.1, reducing the time window during which adversaries may mount more complicated replay attacks.

A. Credential Fabrication Attacks

A credential is legitimate only if it is a fresh hash value on the proper credential chain. The one-way property of credential chains ensures that neither adversaries nor compromised mobile sinks can derive the next available hash value on the credential chain. An adversary can not fabricate a credential chain using intercepted information. The credential chain corresponding to activity $\langle \alpha.ms, \alpha.tt, \alpha.r \rangle$ uses a seed which is generated as $\mathcal{G}(K_M, \alpha)$. Since the value K_M is known only to the base station, the adversary can not generate this seed. The one-way property of our hash functions also prevents the adversary from fabricating a credential and proof combination that yields the correct value for the DC-tree root.

B. Impersonation Attacks

Adversaries or compromised mobile sinks can not impersonate a good mobile sink to collect feedback from sensors. Messages sent by sensors to mobile sinks are encrypted by pairwise keys, which can be generated using the schemes in [7], [3], [5], [13], [4], [2], [26].

C. Replay Attacks

Our scheme is safe from simple replay attack, in which a valid activity request is maliciously repeated by a compromised mobile sink. A fresh credential must be presented for each execution of an activity. If a compromised mobile sink replays some credential $C_\alpha(p)$, sensors will use $C_\alpha(p)$ to compute the DC-tree root ρ'. This computed root value will not match the root value ρ stored, since ρ has already incorporated $C_\alpha(p)$ at some earlier time. Therefore, the replay will be detected and discarded by sensors.

However, replay attacks may take other forms, for many of which there is no defense [8]. For example, a compromised

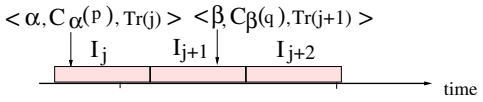

Fig. 3. ms_1 is dispatched to carry out activity α at time window I_j, and ms_2 is dispatched to carry out activity β at time window I_{j+1}. Compromised mobile sink ms_1 can replay the activity request of α only during the time window I_j.

mobile sink may send a transaction request and a valid credential only to a subset of sensors in a specified region, and later resend this same message to the other sensors at a later time. Since these other sensors have not already seen this credential, validation succeeds.

1) Defending Against Replay Attacks: We show how to reduce the window of vulnerability in which a compromised mobile sink may mount complex replay attacks. To this end, we define time windows $I_1, I_2, \cdots, I_k$, covering an appropriate duration into the future. The actual number and size of time windows depends on the application security requirements and the accuracy of loose time synchronization. The base station generates a one-way hash chain $T_r = T_r(0), T_r(1), \cdots, T_r(\eta)$ for each region $r \in R$, and uses a hash value $T_r(j)$ on this chain to "timestamp" all activity request dispatched during time window I_j. We refer to these one-way hash chains as *time chains*.

During the initialization, the base station securely broadcasts the commitments $T_r(0)$ for all time chains (and the DC-tree roots) using μTESLA. Each sensor stores the commitment and the root corresponding to its region. In our new approach, a content $\widehat{N}_i$ of the DC-tree corresponding to an activity α_i is initialized to be

$$\widehat{N}_i = \mathcal{H}(C_{\alpha_i}(0), \alpha_i, T_r(0)).$$

Now let s_i hash values on the credential chain C_{α_i} have been disclosed, and let the latest execution of α_i have been in the time window I_j. In our new scheme, $\widehat{N}_i$ will have been updated to

$$\widehat{N}_i = \mathcal{H}(C_{\alpha_i}(s_i), \alpha_i, T_r(j)).$$

Since the DC-tree node contents incorporate these timestamps, all proofs generated by the base station corresponding to N_i will also incorporate the timestamps, as we describe below.

When the base station dispatches a mobile sink to region $\alpha_i.r$ during time window I_j, it provides it with the credential $C_{\alpha_i}(s_i{+}1), T_r(j)$, and proof $\mathcal{P}(N_i, T_r(j{-}1))$. Upon receiving a request from a mobile sink, a sensor in $\alpha_i.r$ will first verify $T_r(j)$ to be the jth hash value on the time chain T_r, using the commitment for T_r. Next, it computes the content of leaf node $\widehat{N}_i$ as

$$\widehat{N}_i = \mathcal{H}(C_{\alpha_i}(s_i), \alpha_i, T_r(j-1)),$$

after deriving $C_{\alpha_i}(s_i)$ and $T_r(j-1)$ from $C_{\alpha_i}(s_i+1)$ and $T_r(j)$, respectively. Finally, as in Section IV-E, the sensor computes the DC-tree root using $\widehat{N}_i$ and the proof $\mathcal{P}(N_i, T_r(j{-}1))$ and checks if they match.

Now consider the complex replay attacks we have described earlier. In Figure 3, activities α and β are to be executed during time windows I_j and I_{j+1}, respectively. Suppose the mobile sink $\alpha.ms$ is compromised and wants to replay an activity request of α during the time window I_{j+1}. Under our new scheme, this replay attack is not feasible. First, if $\alpha.ms$ tries to use the credential for α and the hash value $T_r(j)$, a sensor will reject this request, since it is expecting the $(j{+}1)$th hash value $T_r(j{+}1)$. Second, let us assume $\alpha.ms$ eavesdrops and picks up $T_r(j+1)$ when the request for activity β is transmitted. Now, $\alpha.ms$ has the proof for $\mathcal{H}(C_{\alpha_i}(s_i), \alpha_i, T_r(j-1))$ but does not have the proof for $\mathcal{H}(C_{\alpha_i}(s_i), \alpha_i, T_r(j))$. Therefore, it can not replay the activity request using $C_{\alpha_i}(s_i)$ and $T_r(j+1)$.

VII. Performance Analysis

We show that our schemes are efficient in terms of computation, communication, and storage overhead.

A. Computation Overhead

To verify a credential in the basic scheme described in Section IV, a sensor must perform a hash operation to compute the activity identifier α, and another hash operation to compute the current contents of leaf node in the DC-tree corresponding to α. Finally, it performs $\log m$ hash operations to compute the root of the DC-tree using the proof. If the verification succeeds, the sensor performs $\log m$ hash operations to update the root of the DC-tree. Therefore, the total computation overhead is $2(\log m + 1)$ hash operations.

In the improved scheme in Section VI-C.1, a sensor first performs one hash operation to validate the time chain value. If the request is a replay, it is rejected, and the computation overhead remains at one hash operation. Otherwise, the sensor will perform $2(\log m + 1)$ hash operations as in the basic scheme. Thus the total computation overhead is $(2\log m + 3)$ hash operations.

B. Communication Overhead

In our basic scheme, the mobile sink needs only to send a request that includes the values $\alpha.ms, \alpha.tt, \alpha.r$, an 8-byte hash value $C_\alpha(p)$, and a proof with $\log m$ hash values. For a sensor network with 100 mobile sinks and 10 transaction types, $m = 1,000$ and $\log m = 10$. In the improved scheme in Section VI-C.1 for preventing complicated replay attacks, the request also includes an 8-byte hash value $T_r(j)$ beyond the request in the basic scheme.

In contrast, the communication overhead of the Blundo-based scheme in [25] includes the activity description, $\log(n_b)$ hash values, and two mutual authentication packets which include two 8-byte nonces and two 8-byte MACs per sensor in the specified region, where n_b is the number of blocks the mobile sinks traveled [25]. Thus the communication overhead for activity verification is $8\log(n_b) + 8 \times 4 \times n_r$ bytes, where n_r is the number of sensors per region. Figure 4 compares the communication overhead for activity verification, excluding the overhead of transmitting the activity elements. With a typical configuration of $n_b = 256$ [25], $m = 1,000$, the

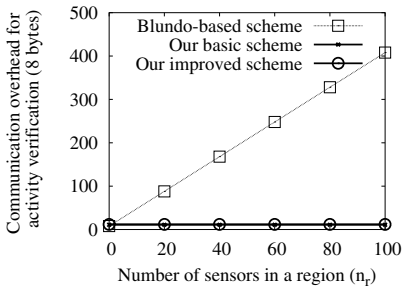

Fig. 4. Communication Overhead Comparison

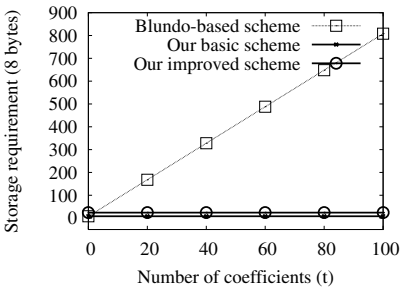

Fig. 5. Storage Requirement Comparison

communication overheads for activity verification of our basic scheme and improved scheme are 88 bytes and 96 bytes, respectively, while that of the Blundo-based scheme is $2,624$ bytes, where the number of sensors per region is $n_r = 80$.

C. Storage Requirement

In our basic scheme, each sensor only needs to store the root of the DC-tree for the region where it is located. Thus the storage requirement is only 8 bytes. In our scheme with timestamps, each sensor must also store the commitment for the time chain, which is 8 bytes.

In contrast, each sensor in the Blundo-based scheme in [25] must store $(t + 1)$ coefficients, where t is the degree of the polynomial. The size of each coefficient is 8 bytes, and thus t is no more than 500 as the SRAM for a Mica2 Mote sensor is only 4KB.

Figure 5 compares the storage requirement of the Blundo-based scheme and our two schemes. Our basic scheme and improved scheme only require 8 bytes and 16 bytes, respectively, while the storage requirement of the Blundo-based scheme is proportional to the number of coefficients.

VIII. Conclusion

In this paper, we propose a resilient scheme to restrict and verify the privileges of mobile sinks, using one-way hash chains and by constructing *dynamic* credential trees. We have shown how to overcome the threshold limitations of existing schemes in terms of the number of mobile sink compromises allowed. Further, unlike the work in [25], our method allows all sensors in a region to verify the privilege of mobile sinks at the same time, making it suitable for applications that require real-time response. Our security and performance analysis show that our methods are secure and efficient.

Acknowledgment

This work is supported in part by grants from Tata Consultancy Services, Inc., and the Fault-Tolerant Networks program of Defense Advanced Research Projects Agency, under contract F30602-01-2-0536.

References

[1] C. Blundo, A. santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung. Perfectly-secure key distribution for dynamic conferences. In *Advances in Cryptology-CRYPTO*, 1993.

[2] H. Chan and A. Perrig. PIKE: Peer intermediaries for key establishment in sensor networks. In *INFOCOM*, 2005.

[3] H. Chan, A. Perrig, and D. Song. Random key predistribution schemes for sensor networks. In *IEEE Symposium on Security and Privacy*, 2003.

[4] W. Du, J. Deng, Y. Han, S. Chen, and P. Varshney. A key management scheme for wireless sensor networks using deployment knowledge. In *INFOCOM*, 2004.

[5] W. Du, J. Deng, Y. Han, and P. Varshney. A pairwise key predistribution scheme for wireless sensor networks. In *ACM CCS*, 2003.

[6] W. Du, R. Wang, and P. Ning. an efficient scheme for authenticating public keys in sensor networks. In *MobiHoc*, 2005.

[7] L. Eschenaer and V.D.Gligor. A key-management scheme for distributed sensor networks. In *ACM CCS*, 2002.

[8] L. Gong. Variations on the Themes of Message Freshness and Replay, or the Difficulty in Devising Formal Methods to Analyze Cryptographic Protocols. In *Proceedings of the Computer Security Foundations Workshop VI*, 1993.

[9] A. Kansal, A. A. Somasundara, D. D. Jea, M. B. Srivastava, and D. Estrin. Intelligent fluid infrastructure for embedded networks. In *ACM MobiSys*, pages 111–124, 2004.

[10] C. Karlof, N. sastry, Y. Li, A. Perrig, and J. Tygar. Distillation codes and applications to DOS resistant multicast authentication. In *NDSS*, 2004.

[11] L. Lamport. Password authentication with insecure communication. *Communication of the ACM*, 24(11), November 1981.

[12] D. Liu and P. Ning. Efficient distribution of key chain commitments for broadcast authentication in distributed sensor networks. In *NDSS*, 2003.

[13] D. Liu and P. Ning. Establishing pairewise keys in distributed sensor networks. In *ACM CCS*, 2003.

[14] D. Liu, P. Ning, S. Zhu, and S. Jajodia. Practical broadcast authentication in sensor networks. In *Mobiquitous*, 2005.

[15] UC Berkey The EECS department. Cotbots: The mobile mote-based robots. `http://www-bsac.eecs.berkeley.edu/projects/cotbots/`.

[16] R. Merkle. Protocols for public keycryptosystems. In *IEEE Symposium on Research in Security and Prvacy*, 1980.

[17] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and D. Tygar. Spins: security protocols for sensor networks. In *ACM MobiCom*, 2001.

[18] B. Przydatek, D. Song, and A. Perrig. SIA: Secure information aggregation in sensor networks. In *ACM SenSys*, 2003.

[19] G. T. Sibley, M. H. Rahimi, and G. S. Sukhatme. Robomote: A Tiny Mobile Robot Platform for Large-Scale Sensor Networks. In *Proceedings of ICRA2002*, 2002.

[20] I. C. Technology. Mica2: Wireless measurement system, http://www.xbow.com/product_pdf_files/wireless_pdf/6020-0042-0%4_a_mica2.pdf.

[21] Y. Tirta, Z. Li, Y. Lu, and S. Bagchi. Efficient collection of sensor data in remote fields using mobile collectors. In *ICCCN*, 2004.

[22] Y. Xu, J. Heidemann, and D. Estrin. Geography informed energy conservation for Ad Hoc routing. In *ACM MobiCom*, 2001.

[23] F. Ye, H. Luo, J. Cheng, S. Lu, and L. Zhang. A two-tier data dissemination model for large-scale wireless sensor networks. In *ACM MobiCom*, 2002.

[24] W. Zhang, G. Cao, and T. L. Porta. Data Dissemination with Ring-Based Index for Wireless Sensor Networks. In *IEEE ICNP*, 2003.

[25] W. Zhang, H. Song, S. Zhu, and G. Cao. Least privilege and privilege deprivation: towards tolerating mobile sink compromises in wireless sensor networks. In *MobiHoc*, pages 378–389, 2005.

[26] L. Zhou, J. Ni, and C. Ravishankar. Supporting Secure Communication and Data Collection in Mobile Sensor Networ. In *INFOCOM*, 2006.