On the Distribution and Revocation of Cryptographic Keys in Sensor Networks

Haowen Chan, Virgil D. Gligor, Adrian Perrig, and Gautam Muralidharan

Abstract—Key management has two important aspects: *key distribution*, which describes how to disseminate secret information to the principals so that secure communications can be initiated, and *key revocation*, which describes how to remove secrets that may have been compromised. Key management in sensor networks face constraints of large scale, lack of a priori information about deployment topology, and limitations of sensor node hardware. While key distribution has been studied extensively in recent work [1], [2], [3], [4], [5], the problem of key and node revocation in sensor networks has received relatively little attention. Yet, revocation protocols that function correctly in the presence of active adversaries pretending to be legitimate protocol participants via compromised sensor nodes are essential. In their absence, an adversary could take control of the sensor network's operation by using compromised nodes which retain their network connectivity for extended periods of time. In this paper, we present an overview of key-distribution methods in sensor networks and their salient features to provide context for understanding key and node revocation. Then, we define basic properties that distributed sensor-node revocation protocols must satisfy and present a protocol for distributed node revocation that satisfies these properties under general assumptions and a standard attacker model.

Index Terms—Sensor networks, security, revocation, key distribution, key management, distributed algorithms.

1 Introduction

As with all networks comprising geographically distributed nodes, communication security in sensor networks requires effective management of cryptographic keys. In contrast to traditional networks, key management in sensor networks is particularly complex due to the large numbers of sensor nodes, the lack of a priori information about the deployment topology of the network, the limited hardware capabilities of the nodes, and the constant exposure of nodes to capture by an active adversary who could obtain key material. Two important aspects of key management are *key distribution* and *key revocation*. Key distribution refers to the task of distributing secret keys between sensor nodes to provide communication secrecy and authenticity. Key revocation refers to the task of securely removing keys that are known to be compromised. If the cryptographic primitives themselves do not expose the secret keys, a reasonable and common assumption, then secret keys can only be exposed by compromising sensor nodes. The problem of sensor node revocation can thus be reduced to that of key revocation. By revoking all of the keys belonging to a known compromised sensor node, we can effectively remove the node's presence in the network.

In contrast to key distribution, which has been studied extensively in recent work [1], [2], [3], [4], [5], key revocation received relatively little attention. With the exception of the centralized revocation scheme proposed by Eschenauer and Gligor [3] and the distributed revocation scheme proposed by Chan et al. [1], no other schemes have been reported to date. Yet, key revocation is as important as key distribution in sensor network key management. A sensor network is generally designed for deployment in open, unmonitored environments exposing nodes to physical attacks. This requires that, in the event of node capture by an adversary, the sensor network have the ability to revoke the cryptographic keys of captured nodes. Otherwise, the entire network's operation may be compromised by an adversary that surreptitiously controls both the operation and communication of these nodes.

In this paper, we first review, in brief, several known methods for key distribution in sensor networks. This forms the background for our main discussion of the problem of distributed key revocation. Distributed node revocation is useful due to its ability to eliminate compromised nodes without requiring a central authority that might become an attractive attack target. Thus, distributed revocation improves reaction time after node capture and overall system resilience. However, distributed revocation protocols are more complex than centralized ones due to the fact that any of the nodes executing the protocol may be malicious and attempt to block or subvert the protocol. Thus, even if a distributed revocation protocol is correctly designed, specified, and formally verified in the absence of an active adversary, assurance of correct behavior would still be lacking. For example, captured nodes could circumvent or block protocol operation, or collude among themselves to execute the revocation protocol correctly against legitimate nodes to disconnect them from the network. So far, research in sensor net key management has been missing the following tools: 1) a rigorous specification of distributed-revocation properties that must hold in a sensor network

- *H. Chan and A. Perrig are with the Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA 15213. E-mail: {haowenchan, perrig}@cmu.edu.*
- *V.D. Gligor and G. Muralidharan are with the Department of Electrical and Computer Engineering, 2415 A.V. Williams Bldg., University of Maryland, College Park, MD 20742. E-mail: gligor@eng.umd.edu, mgautam@glue.umd.edu.*

Manuscript received 13 Oct. 2004; revised 9 June 2005; accepted 17 June 2005; published online 2 Sept. 2005.
For information on obtaining reprints of this article, please send e-mail to: tdsc@computer.org, and reference IEEECS Log Number TDSC-0149-1004.

even in the presence of an active adversary, 2) a precise definition of the adversary model, and 3) a distributed key revocation protocol that satisfies those properties in a general sensor-network setting.

The main contributions of this paper are a rigorous definition of distributed revocation properties for sensor networks, a general active-adversary model, and a protocol for distributed key revocation that satisfies the specified properties under the defined adversary model. However, distributed key revocation cannot be defined independently of the specific key distribution scheme used in a particular sensor network. This is the case because some key distribution methods are more suitable for specific key revocation methods (e.g., centralized or distributed), while others may prevent key revocation altogether. A secondary contribution of this paper is a succinct overview of key predistribution methods and their salient features that affect key revocation and overall sensor-network operation and resiliency.

2 Overview of Key Distribution Schemes for Sensor Networks

The problem of key distribution in sensor networks is as follows:[1] We wish to preload sensor nodes with cryptographic information such that, after deployment, the nodes are able to perform secure communications with each other and initiate a secure network. The scheme must be able to work without prior knowledge of the network deployment topology and also allow new nodes to be added to the network after deployment. A further constraint is that the protocol must be implementable on the nodes' limited hardware and thus it must have low computational and storage requirements; thus, in this paper, we only consider schemes that do not use asymmetric cryptography. We review several classes of known symmetric key distribution protocols suitable for sensor networks in this section.

2.1 Fully Pairwise-Shared Keys

In this approach, every node in the sensor network shares a unique symmetric key with every other node in the network. Hence, in a network of n nodes, there are a total of $\binom{n}{2}$ unique keys. Every node stores $n-1$ keys, one for each of the other nodes in the network.

This class of protocols achieves similar security properties to the class of asymmetric key-establishment schemes: nodes captured do not reveal information in the rest of the network, and central revocation is simple (just broadcast the revoked node's set of keys). However, these protocols require a large amount of memory storage space for all the keys, most of which are not actually used since nodes only communicate with their immediate neighbors and do not need to establish keys with every other node in the network.

2.2 Use of a Trusted Base Station as a KDC

This method of key distribution uses a secure base station as a trusted third party (or Key Distribution Center, KDC) to provide link keys to sensor nodes, e.g., similar to Kerberos [7], [8]. The sensor nodes authenticate themselves to the base station, after which the base station generates a link key and sends it securely to both parties.

An example of a base-station-mediated protocol is SPINS, which includes a protocol where two nodes A and B can establish a session key SK_{AB} by communicating with the base station [9]. The properties of this method of key establishment are that each node only requires preloaded storage of one single key, nodes captured do not reveal information in the rest of the network, and centralized revocation is simple via authenticated unicasts from the trusted base station. The main drawback of this scheme is that the trusted base station represents a single point of compromise for security information, and may also induce a focused communication load centered on the base station which may lead to early battery exhaustion for the nodes closest to the base station. Another concern is that certain networks do not have a suitable highly functional, tamper-proof device that can be used as a secure KDC.

2.3 λ-Secure $n \times n$ Key Establishment Schemes

Blom [10] and Blundo et al. [11] addressed the problem of key distribution and key establishment between all pairs of n principals. While these schemes were originally intended for group keying in traditional networks, and not for sensor networks, we include them here because of their relevance to the development of subsequent key distribution schemes for sensor networks. Both the Blom and the Blundo et al. schemes have an important resiliency property, called the the λ-secure property; i.e., the coalition of no more than λ compromised sensor nodes reveals nothing about the pairwise key between any two noncompromised nodes.

The main advantage of this class of schemes are that they allow a parameterizable trade-off between security and memory overhead. Whereas the full pairwise scheme involves the storage of $O(n)$ keys at each node and is n-secure, this class of schemes allows the storage of $O(\lambda)$ keys in return for a λ-secure property: It is perfectly resilient to node compromise until exactly $\lambda+1$ nodes have been compromised, at which point the entire network's communications are compromised.

2.4 The Basic Random Key Predistribution Scheme

Eschenauer and Gligor proposed the basic random key predistribution scheme [3]. In this scheme, let m denote the number of distinct cryptographic keys that can be stored on the key ring of a sensor node. The basic scheme works as follows: Before sensor nodes are deployed, an *initialization phase* is performed. In the initialization phase, the basic scheme picks a random pool (set) of keys Q out of the total possible key space. For each node, m keys are randomly selected from the key pool Q and stored into the node's memory. This set of m keys is called the node's *key ring*. The number of keys in the key pool, $|Q|$, is chosen such that two random subsets of size m in Q will share at least one key with some probability p. After deployment, neighboring sensor nodes then perform a challenge-response key discovery to find out if they happen to share keys with each other; if they do, then they establish a secure link. If the probability p were chosen correctly for the network's neighbor density (see the paper for details of how this calculation is made), then the resultant graph of secure

1. This section is adapted from an earlier paper [6].

links will be connected with some high probability. The remaining links in the graph are then filled in by routing key-establishment messages along this connected network of initial secure links.

In the basic random key scheme, all nodes use the same key pool Q. This implies that the security of the network is gradually eroded as keys from Q are compromised by an adversary that captures more and more nodes. In this scheme, the number of exposed keys is roughly linear to the number of nodes compromised. This characteristic of the basic scheme motivated development of key predistribution schemes that have better resiliency to node capture. The basic scheme was extended by the q-composite scheme Chan et al. [1], and generalized by the more advanced probabilistic schemes discussed in Section 2.6.

In the q-composite keys scheme, instead of designing for a given probability p of sharing a single key, the parameters are altered such that any two nodes have a given probability p of sharing at least q different keys from the key pool. All q keys are used in the generation of the key which encrypts communications between sensor nodes; hence, in order to eavesdrop on the secured link, the adversary now has to compromise all q keys instead of just one. As q increases, the likelihood of the adversary having compromised all the keys necessary decreases geometrically. However, increasing the probability of overlap in this fashion naturally involves reducing the size of the key pool Q. The smaller key pool size thus makes the scheme more vulnerable to an adversary which is capable of compromising larger numbers of sensor nodes. This trade-off improves the initial resilience of the scheme toward low levels of node compromise, for a subsequent weakness in security once a larger number of sensor nodes have been compromised.

In general, random key predistribution presents a desirable trade-off between the insecurity of using a single network-wide key and the impractical high memory overhead of using unique pairwise keys. Its main advantage is that it provides much lower memory overhead than the full pairwise keys scheme, while being more resilient to node compromise than the single-network-wide-key scheme. Furthermore, it is fully distributed and does not require a trusted base station.

The main disadvantages of the approach are the probabilistic nature of the scheme, which makes it difficult to provide the guarantee of the initial graph of secure links being connected under nonuniform conditions or sparse deployments. Furthermore, since keys can be shared between a large number of nodes, this class of schemes does not provide very high resilience against node compromise and subsequent exposure of node keys.

2.5 Random Pairwise Keys Scheme

The Random Pairwise Keys scheme is a scheme proposed by Chan et al. which is a hybrid of the random key predistribution scheme and the full pairwise keys scheme [1]. Recall that, in the analysis for random key predistribution, it was deduced that as long as any two nodes can form a secure link with at least probability p, then the entire network will be connected with secure links with high probability. Based on this observation, Chan et al. note that it is not necessary to perform full pairwise key distribution in order to achieve a network where any two nodes can find a secure pathway to each other. Instead of preloading $n-1$ unique pairwise keys in each node, the Random Pairwise Keys Scheme preloads $m << n$ unique pairwise keys from each node. The m keys of a key ring are a small, random subset of the $n-1$ possible unique keys that this node could share with the other n nodes in the network. By the same reasoning as the random key predistribution scheme, as long as these m keys provide some sufficient probability p of enabling any two neighboring nodes to be able to establish a secure link, the resultant graph of initial secure links will have a high probability of being connected. The remaining links are then established using this initial graph exactly as in the random key predistribution scheme.

In their paper, Chan et al. present a preliminary initial distributed node revocation scheme that makes use of the fact that possessing unique pairwise keys allows nodes to perform node to node identity authentication. In their scheme, each of the m nodes which share a unique pairwise key with the target node (i.e., the node's *participants*) carries a preloaded *vote* which it can use to denote a message that the target is compromised. These m votes form a Merkle hash tree [12] with m leaves. To vote against the target node, a node performs a network-wide broadcast of its vote (i.e., its leaf in the Merkle hash tree) along with the $\log m$ internal hash values that will allow the other participants of the target to verify that this leaf value is part of the Merkle hash tree. Once at least t participants of a given target have voted and the votes have been verified by the other m participants using the Merkle hash tree, all m nodes will erase any pairwise keys shared with the target, thus revoking it from the network.

The Random Pairwise Keys scheme inherits both strengths and weaknesses from the full pairwise keys scheme (see Section 2.1) and the random key distribution scheme (see Section 2.4). Under the random pairwise keys scheme, nodes captured do not reveal information in the rest of the network, and central revocation can be accomplished by just unicasting to each of the nodes that share keys with the revoked node. It also involves a much lower memory overhead than the full pairwise keys scheme. Unfortunately, like the random key predistribution schemes, it is probabilistic and cannot be guaranteed to work in nonuniform or sparse deployments.

2.6 Multispace Key Schemes

This class of schemes is a hybrid between random key predistribution and the λ-secure $n \times n$ key establishment schemes. These schemes were first proposed by Du et al. [2] and by Liu and Ning [4].

Recall that in random key predistribution, a key pool is first selected from the universe of possible keys. Each sensor node is then given a set of keys from the key pool such that any two nodes possess some chosen probability p of sharing enough keys to form a secure link. Multispace key schemes use the same basic notion of random key predistribution, but use *key spaces* where individual keys are used in random key predistribution. Hence, the key pool is replaced by a pool of key spaces, and each node randomly selects a subset of key spaces from the pool of key spaces such that any two

nodes will have some common key space with probability p. Each key space represents a unique instance of a different λ-secure $n \times n$ key establishment scheme (for example, Blom's scheme [10], see Section 2.3). If two nodes possess the same key space, they can then perform the relevant λ-secure $n \times n$ key establishment scheme to generate a secure session key.

The main advantage of of multispace schemes are that node compromise under these schemes reveals much less information to the adversary than for the random key predistribution schemes. However, they retain the disadvantage of being probabilistic in nature (no guarantee of success in nonuniform or sparse deployments) and furthermore they experience the threshold-based sudden security failure mode that is a characteristic of the λ-secure schemes (see Section 2.3). Other schemes have combined λ-secure schemes with other constructions than random key-space selection; Liu and Ning [4] in particular describe a deterministic grid-based construction where key-spaces are used to preform intermediary-based key establishment between nodes.

2.7 Deterministic Key Predistribution Schemes

One drawback of the random key distribution approach is that it does not guarantee success; Lee and Stinson [13], as well as Camtepe and Yener [14], both propose using combinatorial design techniques to allocate keys to nodes in such a way as to always ensure key sharing between any two nodes. The amount of memory required per node is typically some fractional power of the overall supported network size (e.g., $O(\sqrt{n})$). The main drawback of these schemes is that the same keys are shared between many nodes leading to weaker resilience to node compromise. Chan and Perrig have proposed a deterministic scheme using peer nodes as intermediaries in key-establishment with similar memory overheads [15]; compared with the combinatorial design approach, this scheme trades off increased communication cost for greater resilience against node compromise.

3 The Node Revocation Problem

Key revocation for captured sensor nodes poses new design challenges that do not arise in key predistribution. Key revocation protocols are carried out in the presence of active adversaries. These adversaries can both monitor and modify network messages and, more importantly, can pretend to be legitimate participants in the protocols themselves. Captured (and, hence, compromised) nodes may act as an adversary's surrogates within a revocation protocol and may collude to subvert its execution (e.g., they could block the operation of the protocol by exhausting resources of legitimate nodes, or refuse to carry out key protocol steps). Thus, a specific challenge in the design of revocation protocols is to achieve revocation of sensor nodes that are compromised by an adversary despite the active participation of that adversary in the protocol. An additional challenge, which is shared with key predistribution protocols, is that of using only limited computation and communication resources in the protocol design; i.e., revocation protocols must rely on very simple cryptographic primitives and achieve their goal with a limited number of messages. For example, effective primitives include hash functions such as SHA-1 [16] and hash trees [12], authenticated encryption of protocol messages in one pass over the message data using only the block cipher [17], [18], [19], and evaluations of low-degree polynomials. In contrast, energy or memory intensive primitives such as public-key based primitives or consensus protocols that reach agreements in the presence of malicious adversaries and require multiple network-wide broadcasts are much less desirable.

Recent research on key revocation in sensor networks illustrates two different approaches with orthogonal properties; i.e., a centralized approach (e.g., for the basic random keys scheme [3]) and distributed approach (e.g., for the random pairwise scheme [1]). In the centralized approach, upon detection of a compromised node, a base station broadcasts a revocation message to all sensor nodes that need to remove the copies of keys to be revoked from the compromised node.

In distributed revocation for random pairwise predistributed keys [1] (see Section 2.5), revocation decisions are made by the neighbors of a compromised node. These neighbors vote to decide whether to revoke a given node and, if the vote tally exceeds a specified threshold, revocation takes effect. In contrast with centralized revocation, distributed revocation should be faster, as it requires predominantly local broadcast messages that are inexpensive, and avoids a single point of failure. These features are important since compromised nodes must be sealed off and effectively disconnected from the rest of the network expeditiously. However, the distributed revocation protocol proposed for the random pairwise scheme solely uses network-wide broadcasts of long messages, which is slow, consumes communication energy, and makes the network prone to denial-of-service attacks. Furthermore, its operation requires each node to keep a record of which votes have been heard since the beginning of the network's lifetime. Not only is this memory-intensive, but it also can lead to stale state and incorrect results. For example, suppose that on average, once a month each legitimate sensor node may mistakenly detect a neighbor as having malicious characteristics, causing a revocation vote to be released against it. Further suppose that the threshold number of votes for revocation is three. Since each vote, once cast, can never be retracted, this means that on average any given legitimate node will be revoked from the network in less than three months. These serious flaws make the current scheme impractical.

In general, distributed revocation is inherently more complex than centralized revocation. Such protocols are inherently prone to design error, and the verification of their correctness becomes essential. Correctness verification requires precise definition of both revocation properties and adversary attack model. Both have been lacking to date. In this article, we present the first precise definition of desired properties for the design of distributed revocation protocols. Furthermore, we present the first distributed revocation protocol that can be shown to fulfill the list of desired properties. It is hoped that this new protocol can demonstrate the usefulness and viability of the provided framework, thus facilitating further research into distributed node revocation schemes.

4 Attacker and Communications Model

In this section, we list the assumptions that are general to the node revocation problem. This defines the standard attacker and communications model described by in the literature for key distribution in sensor networks [1], [2], [3], [4]. These assumptions are independent of the details of our proposed protocol (for protocol-specific assumptions, please see Section 6.2).

1. **Adversary has universal communication presence**. We assume that the adversary can simultaneously send and receive an arbitrary number of messages in any part of the network at any time.
2. **Adversary can perform chosen node compromise**. The adversary can selectively compromise a small fraction of the nodes in the sensor network. All data on a compromised node becomes known to the adversary. Furthermore, compromised nodes are controlled by the adversary and can perform active network functions as part of the network until they are revoked. In particular, compromised nodes can launch revocation attacks (see Definition 1 in Section 5.2).
3. **Compromised nodes collaborate**. We assume a single adversary performs all the node compromise in the network; hence, all leaked information in the compromised nodes is collectively known to the adversary and compromised nodes can coordinate to perform collaborative attacks on the sensor network.
4. **Adversary cannot block or significantly delay communications**. We assume that compromised nodes can selectively drop packets which they have received, but the adversary is unable to jam or delay local (single hop) communications in the network whose source and destinations are both uncompromised nodes. The adversary cannot block or delay multihop broadcast messages, either neighborhood-wide or network-wide (i.e., compromised nodes can refuse to forward broadcasts but we assume that there are sufficient legitimate nodes performing the forwarding to ensure complete coverage). The adversary is also unable to partition the network via node compromise. Note that implicit in this assumption is that since an adversary is unable to perform these disruptions, random failures also do not affect the assumed connectivity and communication properties of the network. Under these assumptions, we can assume a bound on the time for broadcasts to propagate over the network. Neighborhood-wide broadcasts take at most time Δ_c to fully propagate over a neighborhood. Network-wide broadcasts or node-to-base-station communications take at most time Δ_d to propagate to every node, where $\Delta_d > \Delta_c$. It may seem that this is a strong assumption since providing efficient reliable broadcast in the presence of active adversaries is a challenging technical problem in itself. However, we note that without this assumption, no revocation protocol (distributed or centralized) could be feasible since the adversary would always be able to interrupt the revocation messages to some part of the network and remain active there. Hence, this is a required assumption if we are to discuss the revocation problem at all. Note that, even in the absence of an efficient solution, simple flooding still provides an inefficient but adequate solution to the reliable broadcast problem. Given that revocation events are rare, it may be possible that the costs associated with simple flooding are acceptable compared with the security benefits of node revocation.

5 Basic Properties of Distributed Revocation

To date, there has not been a succinct definition of distributed revocation properties. It is clear that a distributed revocation protocol should enable a set of nodes to make a decision to exclude another node from the network. However, the problem contains many complicating factors, such as the need to be resistant to attempts by an adversary to block or subvert the protocol. In this section, we describe a set of desired properties, thus providing a precise definition of the problem. Fig. 1 is available as a reference which summarizes the notation used in this part of the paper.

5.1 Correct Operation

The following properties ensure the correct operation of the distributed sensor node revocation protocol:

Property 1 (Completeness). *If a compromised node is detected by t or more uncompromised neighboring nodes, then it is revoked from the entire network permanently (i.e., its subsequent reinsertion into another part of the network is not possible).*

Property 1 ensures that if compromise is detected by sufficient nodes, then the protocol always operates correctly in permanently removing the compromised node, and the adversary is unable to prevent such a revocation from taking place, or circumvent it by reinserting the compromised node elsewhere.

Property 2 (Soundness). *If a node is revoked from the network using this scheme, then at least t nodes must have agreed on its revocation.*

Property 2 ensures that the protocol always requires the agreement of at least t nodes to perform the distributed revocation of any single node. The threshold t provides a mechanism for verifying that the compromise was detected by at least t nodes before a revocation commitment can be made. Note that the property does not state that the revoked node must be actually compromised, nor that all the collaborating revoking nodes must be legitimate. In particular, this property allows an uncompromised node to be revoked by a set of t malicious compromised nodes. This is allowed because it is not realistic to assume that node compromise is

$a \gg b$	a is "much greater" than b, we define this to mean: we can always safely assume $a > b$ regardless of network topology, or changes due to node death, revocation, or compromise. Specifically, the adversary does not have the ability to make $a \leq b$ at any point in time.
Δ_c	maximum time taken for a local-neighborhood broadcast to fully propagate.
Δ_d	maximum time taken for a network-wide broadcast to fully propagate. $\Delta_d > \Delta_c$.
Δ_s	length of time in the revocation session when a node is listening to votes and recording them (i.e. the length of time a session is in the *active* state). $\Delta_s > 2\Delta_c$
Δ_t	maximum time duration of a revocation process from the time the first vote is sent until the node is revoked, or the entire sensor network knows that the session is over
d_{max}	maximum number of nodes that have established a local 1-hop keyed link with any node i.e. the maximum number of local participants for any node
$E_k\{M\}$	message M encrypted with key k using an Authenticated Encryption (AE) mode
$H(x)$	cryptographic hash of the value x.
$H(p)$	cryptographic hash of coefficients of the random polynomial p. e.g. if $p = a_0 + a_1x + a_2x^2 + \ldots + a_{t-1}x^{t-1}$ where a_i are randomly selected then $H(p) = H(a_0 \| \| a_1 \| \| a_2 \| \| \ldots \| \| a_{t-1})$
m	number of participants of each node (i.e. nodes which share pairwise keys with the node)
n	number of nodes in the network
q_{Bs}	t-degree random polynomial used for revocation session s against node B
s_{total}	total number of revocation sessions available against each node
t	threshold number of votes needed to revoke a node

Fig. 1. Notation used in this paper.

always detected with 100 percent effectiveness; hence, if t undetected compromised nodes in the network collude to revoke an uncompromised node, this action is necessarily indistinguishable from having t uncompromised nodes agree to revoke a compromised node.

Property 3 (Bounded Time Revocation Completion). *Revocation decision and execution occur within a bounded time period (let this bound be Δ_t) from the time of sending of the first revocation vote.*

Property 3 ensures that the revocation decision is completed in a timely fashion. This means that an adversary cannot prolong the lifetime of a detected compromised node by delaying the outcome of the revocation decision and, thus, slowing down the revocation process; in order to circumvent the revocation, the adversary has to force the revocation decision to return a negative result (i.e., a decision not to revoke). This property also means that sensor nodes do not have to carry revocation state information for long periods of time, which is attractive since sensor nodes generally have limited memory. This condition also implies that, within the time period Δ_t, if there is an insufficient quorum of sensor nodes that agree that a node is to be revoked, then the revocation decision correctly returns a negative result, instead of waiting indefinitely for the threshold number of votes to be reached. This property resolves the problem of stale state in the original random pairwise revocation scheme, where erroneous votes can accumulate over the network's lifetime and result in the revocation of a legitimate node.

Property 4 (Unitary Revocation). *Revocations of nodes are unitary (all-or-nothing) in the network. Specifically, if a node is revoked in one part of the network, then it will be revoked in the whole network within time Δ_d, where Δ_d is the time taken for any message to propagate across the entire network. If it is not revoked in one part of the network, then it was not revoked in any part of the network in the time prior to the last Δ_d time period.*

Property 4 ensures that revocation is universal within the given limits of communication delay. In particular, the adversary cannot block or delay part of the revocation such that when a compromised node is revoked in one part of the network, it can still operate in a different part of the network for a substantial length of time.

5.2 Resistance to Active Abuse

The properties in Section 5.1 ensure that the protocol operates correctly. However, correctness is insufficient. Since compromised sensor nodes can actively participate in the distributed revocation of other nodes, an adversary could abuse the distributed revocation protocol to further its own agenda without actually interfering in the correct operation of the protocol. In particular, the adversary could use the compromised nodes under its control to launch a *revocation attack*:

Definition 1 (Revocation Attack). *An attack where an adversary uses the distributed node revocation protocol to selectively revoke uncompromised nodes from the network.*

Hence, we require an additional property:

Property 5 (Revocation Attack Resistance). *If c nodes are compromised, then they can only revoke at most αc other nodes where α is a constant and $\alpha \ll \frac{m}{t}$.*

A distributed revocation scheme that satisfies Property 5 restricts the adversary's ability to perform a revocation

attack. Based on the property, when an adversary has compromised some number of nodes, it is only able to successfully revoke a number of nodes that is much less than the total number of nodes it would have been able to revoke if every compromised node had cast a vote against each of the nodes which share a key with it.

6 Our Protocol for Distributed Sensor Node Revocation

In this section, we describe our protocol for distributed node revocation. Although for simplicity we present our protocol in the context of the Random Pairwise Key Distribution scheme, our protocol can be extended for implemention with other key distribution protocols, for example with the generalized random key predistribution mechanisms proposed by Du et al. [2] and Liu and Ning [4].

Our distributed revocation protocol is a significant improvement of the distributed revocation scheme presented by Chan et al. [1]. First, we add the idea of *revocation sessions*, which is a mechanism by which a revocation decision can be completed in bounded time, thus resolving the issue of stale votes causing eventual erroneous revocation of legitimate nodes. Second, we improve the efficiency of the scheme by performing the voting and revocation decision process only using hop-limited local broadcast messages that cover only the target's local neighborhood, thus eliminating the extremely high communication overhead of the original scheme. After the voting process is complete, a single short cryptographic message is then broadcast into the entire network to finalize the revocation outcome. This is in contrast to the original scheme where all voting and communications are full network-wide broadcasts involving large amounts of cryptographic information, which can be very expensive in a large sensor network. Finally, the protocol that we describe is the first distributed node revocation protocol that provides rigorous proofs of high-level desired properties.

The addition of sessions is necesary to facilitate the bounded-time completion of the revocation process. Without sessions, each note could only cast a revocation vote at most once against any other node, and, once cast, it would be unable to withdraw the vote. Given that intrusion detection may yield some level of inaccuracy, this implies that each node can only estimate that one of its neighbors is malicious with some level of certainty. If we set the nodes to trigger their votes at a low certainty level, then inaccurate false-positive votes could accumulate and cause the revocation of legitimate nodes. If we set the nodes to trigger only at a high level of certainty, then we lose the advantage the distributed revocation is fast reacting. The use of sessions avoids this dilemma by allowing nodes to react immediately with high sensitivity to intrusion events and, yet, not have to worry about accumulated votes from false positives being a factor. Note that votes, once cast, become public and cannot be withdrawn. Hence, session information cannot be reused for a subsequent voting process—once voting is complete for each session, the session must be closed regardless of the outcome of the vote.

In the description of the protocol, we will assume that the Random Pairwise Key Distribution is the underlying key establishment protocol, and we perform node revocation actions only among the m nodes which share pairwise keys with the target (i.e., the m *participants* of the target). The set of m participants does not include the target itself. We note that our scheme is general in the sense that it can be directly adapted for other key distribution mechanisms and not just the Random Pairwise protocol. For example, if SPINS [9] was used for key establishment, the set of a participants could be defined as the set of neighbors of a node, and the required revocation data could be downloaded from the base station during the key-establishment process. In the general case, we could set $m = n$ and allow the set of participants of a node to be the entire sensor network. Hence, any other key distribution scheme such as the generalized random key predistribution schemes proposed by Du et al. [2] or Liu and Ning [4] could also be used with our protocol. We further assume that each sensor node has a unique symmetric key that it shares with only the base station. This key is used for authenticated, confidential communications between sensor nodes and the base-station.

6.1 Definitions of Terms Used

In this section, we will define the terms we will use in the description of the distributed revocation protocol. As mentioned, Fig. 1 summarizes the notation we will be using in this paper.

Definition 2 (Neighborhood). *The **neighborhood** of a node is the set of nodes that are within communication range of it.*

Definition 3 (Target, Participants). *A node to be revoked is called a **target** node and any of the m nodes that has a shared pairwise key with a target node is called a participant. A participant is a **local participant** if it has established a direct (1-hop) communication link with the target (hence, it is must be located within the neighborhood of a target); otherwise, it is a **nonlocal participant**. All other nodes are called the nonparticipant nodes. The target node is a nonparticipant of itself, since it cannot participate in key-agreement or node-revocation activities with respect to itself.*

Definition 4 (Local Neighborhood Broadcast). *A **local-neighborhood broadcast** is a multihop broadcast that originates within a given neighborhood and reaches all the nodes inside that neighborhood. Generally, this refers to broadcasts limited to the neighborhood of a revocation target. Local-neighborhood broadcasts take at most Δ_c time to propagate over the entire neighborhood (based on Attacker Model Property 4).*

6.2 Assumptions of Our Protocol

Before we describe our protocol, we list the assumptions under which our protocol operates. Each of these assumptions was necessary to facilitate the proofs that our protocol fulfils the set of properties we described in our problem framework in Section 5. Hence, the strength of these assumptions is closely tied to the strictness of the requirements in our problem framework: A weaker set of properties would require fewer assumptions. Instead of opting for proving weaker properties with few assumptions in our

initial protocol, we chose instead to show that with some number of reasonable assumptions, we are able to describe the first protocol that fulfills all the stringent properties we described in our problem statement. It is hoped that future research will be able to pick up on this direction, and provide new protocols that require fewer assumptions while fulfilling the same set of stringent requirements.

1. **Deployment Atomicity**. Deployment of new nodes is atomic (i.e., it appears to happen instantaneously from the point of view of the network). Deployments do not occur while there are active revocation sessions in the network. All communications in the network are received at their final destinations before deployment begins. In practice, this can be achieved by shutting down the network in an orderly fashion prior to physical deployment of new nodes; after the new nodes have been physically deployed, then the network is turned on again.
2. **Locality Restriction of Compromised Nodes**. The problem of controlling replication of a single node identity or the generation of multiple Sybil node identities across the network is an important technical challenge that is independent of distributed node revocation—examples of techniques to provide such functionality has been described by Newsome et al. [20] and Parno et al. [21]. Since this problem is not the focus of our protocol, we assume the functionality for addressing this problem is already present. More specifically, we assume that node replication, Sybil node identities, or node movement can be detected and the offending node centrally revoked; hence, each malicious node is confined to a single neighborhood. However, all malicious nodes can share information arbitrarily. We assume that all sensor nodes are immobile. Immobile sensor networks constitute a majority of known sensor network applications to date. We design our protocol only for this subset of applications since mobility creates complications that this initial protocol is not meant to address.
3. **Node Degrees**. Each node i has $d_i \gg t$ local participants with which it has successfully performed key-establishment, where $m \gg d_i$ (hence, $m \gg d_{max}$, where d_{max} is the maximum number of local participants for any node). We assume that this is enforced by some degree-counting mechanism, where nodes with low degrees are centrally revoked (i.e., it is enforced that $d_i > d_{min}$ for some d_{min}). An example is described by Chan et al. [1]. An adversary could attempt to exploit the degree-counting mechanism by reducing the degrees of legitimate nodes (e.g., by causing malicious nodes to refuse to complete key exchange with legitimate nodes). This is a very inefficient attack since, in order to disable a single node in this manner, an adversary has to capture or disable a large number of the nodes in its neighborhood. For example, typical values for d_{min} should not exclude legitimate nodes that simply happen to be deployed in a sparse area. Since t is typically also not large compared with deployment density, the typical degree of any node is usually much greater than $d_{min} + t$. Hence, if an adversary wishes to cause the removal of a node, for the vast majority of the nodes it would be more efficient to simply capture t local participants and perform a coordinated revocation attack than to attempt to drive the target node's degree below d_{min} in order to cause automatic central revocation.
4. **Events that can cause a node to start a revocation against another node are visible to the node's entire neighborhood**. If we allow nodes to trigger revocation sessions based on events that are only observable to themselves, then it is difficult to ascertain if this is a legitimate response to misbehavior, or if a malicious node is performing spurious revocations, or if a malicious node is deliberately inducing a legitimate node to perform legitimate revocation actions that are guaranteed to fail. Hence, we only trigger revocation based on events that are visible to all the target node's neighbors. By the previous Assumption 3, this also means that such events are always observed by some $d_i \gg t$ legitimate nodes. Hence, nodes only react to events which they know can also be observed by many other nodes and, thus, there is assurance of reaching the threshold t whenever a voting session is started legitimately. In practice, the legitimate nodes observing the event may encounter false positives or false negatives, which may cause the initiation of failed revocation sessions. We assume that the rate of false positives and false negatives is small. In fact, the development of distributed intrusion detection mechanisms is a challenging research problem. As a current working example, we can suppose that our detection mechanism only reacts to highly visible, egregious misbehaviors such as repeatedly performing spurious transmissions, or complete absence of communication over a long period of time. As intrusion detection mechanisms become more advanced and more accurate, the range of detectable behaviors supported by this scheme will also increase.
5. **Revocation Sessions Are Always Available**. We assume that revocation attempts by legitimate nodes are infrequent enough that only s sessions need to be stored on the sensor nodes such that no node ever runs out of revocation sessions during its lifetime. We assume that $s > t$ but it is small (e.g., less than 10). If all the available sessions against a given target node B are exhausted, all of B's participants will be able to detect this since each of them is aware of how many sessions are remaining for B. Hence, each participant simply suspends communications with B until new sessions arrive, thus temporarily excluding B from the network as long as it cannot be revoked. An adversary may attempt to exhaust all available revocation sessions against a given node

such that this behavior occurs, but such an attack is easily detectable and counteracted. For example, if a node A repeatedly initiates revocation sessions against node B that subsequently fail, then this behavior is reported by its fellow participants to a central base station. By assumption 4 above, we know that such behavior is not typical of a legitimate node because a legitimate node should only react to events visible to the entire node neighborhood. If this happens repeatedly, the base station may then detect if this is an attempt to perform a session exhaustion attack, and the offending node A will be centrally revoked. An adversary may attempt to circumvent this mechanism by using s nodes in turn to attempt the session exhaustion attack. However, since we required that $s > t$, in this case, the adversary could simply revoke the target node permanently instead of performing such an attack. Hence, we can assume that revocation sessions are always available for the revocation of any target node. Node sessions may be slowly expended as false positives and random events accrue over time. However, new sessions can be refreshed via encrypted transmissions from the base station to each relevant participant. While such a refresh is very costly in terms of communication overhead, we note that since revocation attempts are infrequent, a node would probably only experience a very small number of such refresh events in its lifetime.

We further note the following nonassumptions of our protocol:

1. **We do not assume time synchronization between sensor nodes**. We do not make use of any internal clock information in sensor nodes. We also do not expect sensor nodes or the network to react instantaneously to any event.
2. **We do not assume any asymmetric-key cryptography capabilities on the sensor nodes**. We present our protocol using only symmetric-key primitives, thus trading off an increased memory requirement for lower energy and computational requirements. We note that it is simple to convert our symmetric-key scheme to one based on asymmetric cryptography. In particular, note that such a conversion would assure that revocation sessions are always available since each node would be able to generate more revocation sessions for any number of neighbors indefinitely.

6.3 Protocol Overview

A high-level overview of our revocation protocol is as follows: Local participant nodes perform voting in sessions to agree to revoke a neighbor. In each voting session, we use a secret-sharing scheme to tally revocation votes from each participant. Each vote is a secret share, and voting is performed by broadcasting the secret share to all the participant nodes in the target node's neighborhood. Once a certain time has elapsed since the first vote of the session was broadcast, each local participant tallies the votes that it heard; if sufficient votes were heard, then it can prove this fact by broadcasting the secret of the secret-sharing scheme. Such a broadcast indicates that the target was successfully revoked and causes all nodes in the network to erase the keys associated with the target, thus eliminating it from the network. In the subsequent sections, we describe the protocol in detail.

6.4 Cryptographic Primitives

In the revocation protocol, we make use of *random polynomials*. Polynomial $q(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_{t-1}x^{t-1}$ is random if all its coefficients $a_0, a_1 \ldots a_{t-1}$ are random uniformly distributed values in a certain range $[0, l-1]$ (e.g., $l = 2^{64}$). We define the cryptographic hash of a random polynomial $q(x)$ to be $Hq(x) = Ha_0||a_1||a_2|| \ldots ||a_{t-1}$, where H is a hash function and $a_0, a_1 \ldots a_{t-1}$ are the coefficients of $q(x)$.

The protocol also uses Authenticated-encryption (AE) modes. AE modes detect 1) ciphertext forgeries (e.g., ciphertext messages produced by manipulation of encrypted-message blocks or simply by arbitrary message strings that are not obtained by encryption) and 2) false (or inauthentic) decryption keys (i.e., decryption keys not used by the corresponding encryption operation), during ciphertext decryption. AE modes specially designed to save power and energy are particularly well-suited for sensor networks (e.g., AE modes that use a single pass over the data using a single cryptographic primitive, namely, the block cipher [17], [18], [19]).

6.5 Offline Node Initialization

For node initialization, we first compute s_{total} random polynomials of degree t for each of the n nodes in the network, where s_{total} is the number of revocation sessions (attempts) against any target node in the network. For example, if the size of the sensor network is $n = 10,000$ and the number of revocation sessions is $s_{total} = 6$, this would require 60,000 polynomials of degree t. This is not a very large number considering that they are generated offline, efficiently. For the purposes of discussion, we number each session against a given target from $1 \ldots s_{total}$, with session 1 starting first and proceeding sequentially until the last session s_{total}. The voting sessions are necessary to fulfil Property 3 (bounded-time revocation decision completion).

Second, on each node A, for each node B of A's m participants, and for each revocation session s against target B, based on our random polynomial q_{Bs}, we load the revocation vote from A against B. This revocation vote consists of the secret share $(q_{Bs}(x_{ABs}), x_{ABs})$, AE encrypted with the activation mask $Mask_{ABs}$ that B gives to A. The points at which the secret-sharing polynomial are evaluated (e.g., x_{ABs}) are generated such that no two participants have the same revocation secret share. The preloaded data is represented as $(E_{Mask_{ABs}}(q_{Bs}(x_{ABs}), x_{ABs}))$. The purpose of the masks is to ensure that each node is only able to revoke nodes within its immediate neighborhood. Since we assume nodes are unable to move or replicate (see Assumption 2), each malicious node can only collect masks from one neighborhood, thus limiting their revocation power. For each vote, we also load the $\log m$ authenticating hash values for the Merkle tree with leaves $(q_{Bs}(x_{iBs}), x_{iBs})$ for each node i in B's participants (a total of m leaves). The root R_B of this Merkle tree is also stored. When the revocation vote

Stored on each node A:
For each node B in A's m participants,
For each revocation session s against node B (total of s_{total} sessions):
1. $E_{Mask_{ABs}}\{(q_{Bs}(x_{ABs}), x_{ABs})\}$
Revocation secret share (vote) from A against B
(AE encrypted with the activation mask $Mask_{ABs}$ that B gives to A)
2. $\log m$ authenticating hash values, for the Merkle tree with leaves $(q_{Bs}(x_{iBs}), x_{iBs})$ for each node i in B's participants (total m leaves)
3. R_B
The root of the Merkle tree described above.
4. $H^2(q_{Bs})$
The hash of the hash of the revocation polynomial of B

For each node B in A's m participants,
For each revocation session s against node A:
$Mask_{BAs}$ for B to decrypt votes against A.

Fig. 2. Information preloaded on each sensor node.

is cast, these $\log m$ authenticating values are also attached to the message. This allows fellow participants of B to verify the authenticity of the vote by computing the hashes up the tree and comparing it to the known root value R_B. Finally, we load $H^2 q_{Bs}$, which is the hash of the hash of the revocation polynomial of B. This will allow nonlocal participants to verify the authenticity of a revocation decision against B.

Third, on each target node A, for each node B in A's m participants, for each revocation session s against target node A, we load the value $Mask_{BAs}$ that A will give to B to allow B to decrypt votes against A.

The information that is preloaded on each sensor node is summarized in Fig. 2. The total storage overhead per node is $O(s_{total} m \log m)$. Note that the overhead of the random pairwise keys scheme itself is already $O(m)$, hence this revocation scheme is only a small factor larger than the basic overhead necessary for key-establishment.

6.6 Connection Establishment

The masks for the current revocation sessions are exchanged at connection time. Masks are verified if votes can be decrypted—due to the authenticated encryption property, an incorrect mask would cause decryption of the vote to fail. If the mask exchange protocol is not completed (e.g., by a malicious node refusing to reveal its mask, or revealing an incorrect mask), and still does not succeed after some number of retries, then the link is dropped. Malicious nodes which repeatedly refuse to perform mask exchange will have a low degree and this will be detected and centrally revoked by the degree-counting scheme assumed in Assumption 3. Note that, by Definition 3, if the unmasking protocol fails and the link is dropped between two neighboring nodes, we consider them to be *nonlocal participants* even though they are in physical communication range of each other.

A malicious node M_1 may attempt to circumvent the unmasking process by having another malicious node M_2 act as its proxy in a distant location to obtain extra masks. However, since all communications are authenticated (node-to-node identity authentication is a prerequisite for any distributed revocation protocol), this is exactly equivalent to M_1 producing a Sybil replication at M_2's location. By Assumption 2, functionality to address this kind of replication is already present; hence, such an attack is not feasible.

Once the masks are exchanged, they are used to decrypt the relevant votes.

6.7 Stages of a Revocation Session

We now describe the process of an entire revocation session s for some given participant A. Each node keeps a state variable for each session. Before any voting occurs in a current session, node A's state for session s is the pending state. In this state, A is awaiting the first vote in the session s. When the first vote of the session is cast or received by A, A starts its timer for the new revocation session and changes its state for the current revocation session to active. Only when A is in the active state does it record and verify other votes cast in this session by fellow participants. When A enters the active state, if A believes that B is compromised, it will cast its vote if it has not already done so. The active state lasts for exactly Δ_s time for each node, after which the node transitions to the **completed** state for this session, and starts the *pending* state for the next session. Δ_s is a precomputed time duration based on the time that a revocation decision is expected to take. In order to ensure full dissemination of all messages within a session, we require that $\Delta_s > 2\Delta_c$ where Δ_c is the maximum time that a message needs to completely propagate in a local neighborhood broadcast.

Definition 5 (Current Session). *The **current session** of a node for the revocation of a target is the session with the smallest session number that has not yet completed (i.e., it is either active or pending).*

Any session that comes after the current session is considered to be in the **not-current** state. Any votes received for a session that is *not-current* is buffered and only acted upon after the current session is completed.

6.8 Voting in a Revocation Session

When a node A detects compromise of a node B, it votes *both* in the current session *and* on the next session. Voting in the next session occurs immediately upon completion of the current session. This voting process is to ensure that A's vote is actually counted because a vote in an already-active session cannot be guaranteed to disseminate in time for all nodes to receive it. For example, suppose both node A and C are currently near the end of an active revocation session s against B (i.e., both of them have been in the *active* state for session s for nearly Δ_s time). Suppose that both of them have recorded $t-2$ votes against B. Simultaneously, both A and C detect that B is compromised. If they vote only in the current session, since the session is almost over, it is possible that neither of their votes reach each other within the remaining time for the session, so both A and C tally $t-1$ votes and fail to reach the threshold needed to revoke B, even though t votes were cast in session s. Hence, it is necessary that the nodes also vote in the next session $s+1$. Voting only in the next session (and not the current session) is insufficient since in the interest of speed, we would like the current revocation session to succeed if at all possible.

When A votes against B, it performs an unencrypted hop-limited broadcast of A's against node B in the current revocation session s, $(q_{Bs}(x_{ABs}), x_{ABs})$ along with the $\log m$ Merkle authentication values that verify that this vote is a valid vote. This is a local neighborhood broadcast, i.e., the broadcast only needs to go far enough to ensure complete dissemination in the neighborhood of B (four-six hops can cover the area with high probability [3]). Only nodes that can verify the authenticity of the vote using the Merkle authentication values (i.e., the participants of B) will disseminate the broadcast. This ensures complete coverage with high probability [1].

6.9 Completing the Revocation Process

When A's state for the session has transitioned to *complete*, it counts the number of votes it has received while it was in the active state.

If A has at least t revocation votes (including its own if it detected the compromise, otherwise not), then it computes the revocation polynomial of B for this session, q_{Bs}. From this, A computes the hash of the polynomial, $H(q_{Bs})$. This value is then broadcast through the entire network. Note that we broadcast only the hash of the polynomial q_{Bs} instead of the polynomial itself; thus, we only need to transmit a single cryptographic value instead of a lengthy set of large polynomial coefficients. All participants of B verify this preimage against the value stored in their memory, $H^2(q_{Bs})$. If the verification is successful, then all shared keys with B are deleted and B is marked as revoked. The broadcast is then disseminated to the other participants of B until the entire network is covered.

If A does not have at least t verified revocation votes, then the revocation session has failed. Each local participant privately notifies the base station of the failed revocation session, thus ensuring that future deployed nodes will be deployed with the most current revocation session in the correct state (i.e., in the *pending* state since deployments should not occur if there are any active revocation sessions in the network). Local participants of B then proceed to request the masks for the new session $s+1$, i.e., $Mask_{AB(s+1)}$ (if B does not respond correctly then its degree is reduced and it may be centrally revoked due to insufficient degree (Assumption 3). To save memory, any state regarding the old revocation session s is cleared.

6.10 Proofs of Properties

In this section, we shall prove that Distributed Sensor Node Revocation satisfies the properties outlined in Section 5.

Lemma 1. *Every node is deployed with the correct current revocation session for its participants.*

Proof. Immediate from Deployment Atomicity (Assumption 1). We assume that the base station keeps track of the current revocation session of each node and updates newly deployed nodes with the correct session information; since deployment is atomic, the base station always has a correct notion of the current session of every node in the network. □

Definition 6 (Session Agreement). *Two nodes are in* ***session agreement*** *with respect to a target node at some instant in time if, for some session s, either*

1. *session s is pending for both nodes,*
2. *session s is active for both nodes,*
3. *session s is active for one node B and session s is completed for another node A, but session s is completing within time Δ_c for node B, or*
4. *session s is active for one node A and pending for the other node B, but node B is activating session s within Δ_c time.*

Lemma 2. *At any given point in time, any two uncompromised local participants are in session agreement for any target node.*

Proof. Let the initial session of node that was deployed later be r. By Correct Deployment (Lemma 1), when the later node was deployed, the earlier node also had session r pending since 1) deployment only occurs when there is no active revocation session and 2) the deployed nodes are always deployed with the correct current session. Hence, the two nodes are session agreement on session r when the later node was deployed.

We now show the inductive step that if the two nodes are in session agreement in all time instants $T' \leq T$ (after both nodes have been deployed), then they are still in agreement in time $T+\epsilon$ where $\epsilon < \Delta_s$. If the states of the nodes are unchanged between the two times, or if both nodes changed states into the same states at time $T+\epsilon$, then we are done. Otherwise, we have five nondegenerate cases. Recall that Δ_s was chosen at design time such that $\Delta_s > 2\Delta_c$.

Case 1: Session s is pending for both nodes at time T, and at time $T+\epsilon$, node A activated session s. Since A is uncompromised, it will perform a local broadcast of the first vote of session s. By Attacker Model Property 4, local communications cross the local neighborhood in time Δ_c. Since node B is in pending mode for session s, it will receive the first vote of session s in time Δ_c and activate the session. Hence, the nodes are in agreement by option 4 of the definition of agreement.

Case 2: Session s is active for both nodes at time T. At time $T+\epsilon$, node A completed session s, but node B still has the session active. By the induction hypothesis, the nodes are in agreement when the first one of them started session s; hence, they must have both started session s within Δ_c of each other. Since each session takes a fixed amount of time to complete (Δ_s), and A completed session s at time $T+\epsilon$, we know that B will complete session s by $T+\epsilon+\Delta_c$; hence, they are in agreement by option 3 of the definition of agreement.

Case 3: Session s is active for node B and session s is complete for node A at time T. At time $T+\epsilon$, session s has completed for node B. Hence, session $s+1$ must be either active or pending for node B. If session $s+1$ is active or pending for both nodes, then by options 1 or 2 of the definition, we have agreement. Otherwise, one node has session $s+1$ active and the other has session $s+1$ pending. By Attacker Model Property 4, local communications cross the local neighborhood in time Δ_c. Hence, the node with session $s+1$ pending will receive the first vote for session $s+1$ within time Δ_c and start the session. Hence, we have agreement by option 4 of the definition.

Case 4: At time t, session s is active for A and pending for B. Two subcases: Subcase 4a: At time $T+\epsilon$, session s is completed for A and session s is still pending for B. This subcase is impossible. By the induction hypothesis, the nodes are in agreement when A started session s; hence, B must be starting session s within Δ_c of that time. In order for B to be still pending session s at time $T+\epsilon$, it must be that A started session s after time $T+\epsilon-\Delta_c$. However, we know that A needs at least Δ_s time to complete session s, so the earliest time it can complete the session is at $T+\epsilon-\Delta_c+\Delta_s$, but since $\Delta_s > \Delta_c$, this time is after $T+\epsilon$ (contradiction). Hence, this subcase is impossible.

Subcase 4b: At time $T+\epsilon$, session s has completed for A and session s is active for B. By the induction hypothesis, the nodes are in agreement when A started session s; hence, they must have both started the session within Δ_c of each other. Since each session takes a fixed amount of time to complete (Δ_s), and A completed session s at time $T+\epsilon$, we know that B will complete session s by $T+\epsilon+\Delta_c$; hence, we have agreement by option 3 of the definition.

Hence, we have shown by induction that A and B are in session agreement for all times T. □

Property 1 (Completeness). *If a compromised node is detected by t or more uncompromised neighboring nodes, then it is revoked from the entire network permanently (i.e., its subsequent reinsertion into another part of the network is not possible).*

Proof. Suppose a compromised node B is detected by a set S of t or more uncompromised neighboring nodes. By Lemma 2, all t nodes are in mutual session agreement on some session for the target B. We proceed by two possible cases. Let node C be the node with the *lowest* current session in the set S. Recall that Δ_s was chosen such that $\Delta_s > 2\Delta_c$.

Case 1: The current session s is pending for node C. Consider an arbitrary node A in S. By the definition of session agreement (Definition 6), A either (1a) has session s pending, or (1b) has session s active. A cannot have session $s' > s$ pending or active since in that case it could not be in session agreement with C. For Case 1a, we have that A will vote in session s and change its state to session s active. Since session s is starting at this instant for A, it has at least time $\Delta_s > \Delta_c$ to receive all t votes in session s. For Case 1b, node A will vote in session s. We know by that session s could not have been active for node A for more than Δ_c time, otherwise within that time the first vote would have reached node C and activated session s. This is due to Attacker Model Property 4 which states that local broadcasts take at most Δ_c time. Hence, since session s has not been active for more than Δ_c time, so there is at least $\Delta_s - \Delta_c > \Delta_c$ time remaining for it to receive all the t votes in session s. In both Cases 1a and 1b, we have that all nodes in S will vote in session s and every node has at least Δ_c time to receive all the votes from the other nodes. Since the votes are local broadcasts and, so, need at most Δ_c time to propagate and always reach their destinations (Attacker Model Property 4), we know that all $|S| \geq t$ votes in session s will be received by all members of S.

Case 2: The current session $s-1$ is active in node C. Consider an arbitrary node A in S. By the definition of session agreement (Definition 6), A either 2a has session $s-1$ pending 2b has session $s-1$ active, 2c has session s pending, or 2d has session s active. It cannot be that session s is completed for A since that would mean that session s would also complete for C within time Δ_c, which cannot be true since C will take at least $\Delta_s > 2\Delta_c$ time to complete session s which has started at this point. Case 2a is covered by Case 1 above. Hence, we only have three cases (2b, 2c, and 2d). For Case 2b, both nodes A and C have detected compromise while a session is active, so they will vote in both session $s-1$ and session s. We know by Lemma 2 that A and C were in agreement when the first one of the started session $s-1$; hence, they must have started session $s-1$ within Δ_c of each other. Since they will both start session s immediately after they complete session $s-1$, they will also start session s within Δ_c of each other. Hence, each node has at least $\Delta_s - \Delta_c > \Delta_c$ time to receive the votes in session s. For Case 2c, A will vote in session s (as will C). Since session s has not started for node A, it will have $\Delta_s > \Delta_c$ time to receive all $|S|$ votes in session s. For Case 2d, node A will vote in both session s and session $s+1$. Since node C still has session $s-1$ active, it could not be that node A completed session $s-1$ and, then, started session s more than Δ_c time ago. This is because by Definition 6, node C must be completing session $s-1$ within time Δ_c of the time when node A completed session $s-1$. Since this has not happened yet, node A must have completed session $s-1$ (and started session s) within the last Δ_c. Hence, node A still has at least $\Delta_s - \Delta_c > \Delta_c$ time to receive the votes in session s. Hence, in all cases (2b, 2c, 2d), we have that all the nodes in S will vote in session s, and every node has at least Δ_c time to receive the votes of the other nodes in S. Attacker Model

Property 4 states that local broadcasts always cover the local neighborhood and take at most Δ_c time. Hence, we know that all the nodes in S will have received all the $|S| \geq t$ votes in session s by the end of the session.

Hence, all $|S| \geq t$ nodes will receive and correctly verify (via the Merkle hash tree mechanism) all unique votes in session s against B. By the preimage resistance of the Merkle hash tree, the adversary cannot fabricate and inject invalid votes, and it has no way of reconstructing a valid vote without first being able to produce a preimage for the cryptographic hash function. Replayed votes have no effect since only unique votes are recorded by the nodes. Hence, each node in S is able to record at least t valid votes and, thus, generate the revocation secret q_{Bs} for session s. The verifying revocation value $H(q_{Bs})$ is the broadcast throughout the network. By Attacker Model Property 4, this broadcast reaches every participant of B and, thus, induces a permanent network-wide revocation of node B. □

Property 2 (Soundness). *If a node is revoked from the network using this scheme, then at least t nodes must have agreed on its revocation.*

Proof. Suppose a node B is revoked in the network using this scheme. Then, the correct verifying revocation value $H(q_{Bs})$ must have been broadcast for some session s. Since the cryptographic hash is preimage resistant, this means that some party must have computed the revocation secret q_{Bs}. Since q_{Bs} is not stored in its entirety on any node in the network, by the threshold property of the random polynomial, the only way to reconstruct q_{Bs} is to obtain at least t secret shares of q_{Bs}. Hence, at least t nodes must have colluded to perform this computation. □

Property 3 (Bounded Time Revocation Completion). *Revocation decision and execution occur within a bounded time period (let this bound be Δ_t) from the time of sending of the first revocation vote.*

Proof. Let the time when the first vote is cast against some target node B be time T. By Attacker Model Property 4, all local participants will have received this vote by time $T + \Delta_c$. By the protocol, each node makes its revocation decision within time Δ_s of receiving the first vote. If the outcome is positive, the verifying revocation value is then broadcast to the rest of the network. Otherwise, the base station is notified of the failure of the session. By Attacker Model Property 4, the broadcast takes at most time Δ_d to reach the rest of the network (as does the base station notification). The total time taken is thus at most $\Delta_t = \Delta_c + \Delta_s + \Delta_d$. □

Property 4 (Unitary Revocation). *Revocations of nodes are unitary (all-or-nothing) in the network. Specifically, if a node is revoked in one part of the network, then it will be revoked in the whole network within time Δ_d, where Δ_d is the time taken for any message to propagate across the entire network. If it is not revoked in one part of the network, then it was not revoked in any part of the network in the time prior to the last Δ_d time period.*

Proof. Case 1 (If a node is revoked in one part of the network, it will be revoked in the entire network in time Δ_d): If a node is revoked in one part of the network, then it must be that the correct verifying revocation value $H(q_{Bs})$ must have been received in that part of the network. The nodes that receive $H(q_{Bs})$ will rebroadcast it to the rest of the network; by By Attacker Model Property 4, the entire network will receive this broadcast in time Δ_d and, thus, the node will be completely revoked in time Δ_d.

Case 2 (If a node is not revoked in some part of the network, then it was not revoked in any part of the network in the time prior to the last Δ_d): We proceed to prove the contrapositive, i.e., if the node was revoked in any part of the network in the time prior to the last Δ_d, then it must be revoked in this part of the network. From Case 1, we can see that if the node was revoked in any part of the network in the time prior to the last Δ_d, then it must be revoked in the entire network by now. Hence, it must be revoked in this particular part of the network.□

Property 5 (Revocation Attack Resistance). *If c nodes are compromised, then they can only revoke at most αc other nodes where α is a constant and $\alpha \ll \frac{m}{t}$.*

Proof. Suppose c nodes are compromised. By Assumption 2, each of these nodes identities are fixed in one location and the adversary is unable to create other points of presence elsewhere in the network. Hence, by Assumption 3, each compromised node i can only establish connections with $d_i \ll m$ other nodes. Thus, each compromised node can unmask at most d_i votes each. The total number of unmasked votes is thus $\sum_{i=1}^{c} d_i$. Hence, the maximum possible number of nodes revocable by these votes is $\sum_{i=1}^{c} \frac{d_i}{t} < \frac{cd_{max}}{t} \ll \frac{cm}{t}$. □

7 Conclusions

In this article, we provide an overview of the key management problem for sensor networks. In the first part of our paper, we provide a brief summary of existing key distribution techniques for sensor networks. These techniques address only the key-establishment part of our key-management problem. A comprehensive key-management protocol suite must also possess the ability to revoke the secret keys that have been compromised by an adversary. To date, this important research problem has been insufficiently pursued. To address this, we have presented a precise formulation of the distributed revocation problem as well as an initial protocol that has been shown to satisfy the requirements of this problem formulation.

Distributed revocation protocols have several advantages over centralized revocation that come into play when compromised nodes must be disconnected from the network. The first advantage is speed, due to the fact that they require only broadcast messages of a few hops that reach their local destinations reliably. The second advantage is the avoidance of single points of failure. However, distributed revocation protocols are inherently more complex than centralized protocols and, hence, more prone to design error since compromised sensor nodes can participate in the

revocation protocol and attempt to block or circumvent it. Thus, the precise specification and verification of the revocation-protocol properties and of the attack model are essential to the secure operation of sensor networks deployed in hostile environments.

In this paper, we defined a set of high-level properties for distributed sensor-node revocation and presented a protocol that satisfies these properties under general assumptions and a standard attacker model. In particular, we showed that, unlike most other cryptographic protocols, distributed-revocation protocols can be executed while the adversary exercises *complete* control of compromised nodes which take the role of active participants in the protocol, but which have malicious objectives such as attempting to block revocation, or selectively revoke noncompromised nodes and disrupt network operation. Hence, due to the complex nature of distributed sensor node revocation, it is important to obtain rigorous proofs of our set of high-level properties in order to show that the revocation protocols cannot be subverted or abused by compromised nodes. We have also described a distributed node revocation protocol that we have proven to have the essential set of high-level properties that ensure correct functioning and resistance to abuse by a malicious attacker. Our scheme has stronger properties and is also more efficient and faster than the previous distributed revocation scheme described by Chan et al. [1].

Several research problems are opened by our work in distributed revocation. First, our distributed revocation protocol is described for networks in which keys are predistributed in a random pairwise manner. While the protocol can be extended to other types of key predistribution, straightforward extension (e.g., by setting the number of participants to n, the size of the network) may not be the most ideal method of distributed revocation for all key predistribution protocols such as probabilistic predistribution of random keys [3], or hybrid probabilistic and random pairwise predistribution [2], [4]. Hence, distributed revocation schemes that are specially designed for other key distribution protocols are needed. Second, we present a specific metric of protocol resistance to active attacks, i.e., the ratio of the number of uncompromised nodes that can be revoked by a group of colluding compromised nodes under the control of an active adversary, versus the number of colluding nodes. Other distributed revocation protocols may be more resilient under our metric, or may suggest other useful metrics for resiliency. Third, design space of policies for distributed revocation is substantial: We only explored a policy based on local neighborhood decisions. Other policies may be equally useful, for example, those that involve all key-connected neighbors of a revocation target, and not just the local neighbors. Finally, we note that we established the proof of our revocation properties only under some strong assumptions; future research to develop protocols that operate under fewer, weaker assumptions (such as allowing sensor nodes to be mobile) may be fruitful.

ACKNOWLEDGMENTS

The views and conclusions contained in this paper are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of Bosch, Carnegie Mellon University, the US National Science Foundation, the US Army Research Office, The University of Maryland, the US Government or any of its agencies. V.D. Gligor's research was supported in part by the US Army Research Office under Award No. DAAD19-01-1-0494, and by the US Army Research Laboratory under Cooperative Agreement DAAD19-01-2-0011 for the Collaborative Technology Alliance for Communications and Networks. A. Perrig's research was supported in part by CyLab at Carnegie Mellon University under grant DAAD19-02-1-0389 from the US Army Research Office, and grant CAREER CNS-0347807 from the US National Science Foundation, and by a gift from Bosch.

REFERENCES

[1] H. Chan, A. Perrig, and D. Song, "Random Key Predistribution Schemes for Sensor Networks," *Proc. IEEE Symp. Security and Privacy,* May 2003.

[2] W. Du, J. Deng, Y. Han, and P. Varshney, "A Pairwise Key Pre-Distribution Scheme for Wireless Sensor Networks," *Proc. 10th ACM Conf. Computer and Comm. Security (CCS 2003),* pp. 42-51, Oct. 2003.

[3] L. Eschenauer and V. Gligor, "A Key-Management Scheme for Distributed Sensor Networks," *Proc. Ninth ACM Conf. Computer and Comm. Security,* pp. 41-47, Nov. 2002.

[4] D. Liu and P. Ning, "Establishing Pairwise Keys in Distributed Sensor Networks," *Proc. 10th ACM Conf. Computer and Comm. Security (CCS 2003),* pp. 52-61, Oct. 2003.

[5] D. Liu and P. Ning, "Location-Based Pairwise Key Establishments for Static Sensor Networks," *Proc. ACM Workshop Security in Ad Hoc and Sensor Networks (SASN '03),* Oct. 2003.

[6] H. Chan, A. Perrig, and D. Song, "Key Distribution Techniques for Sensor Networks," *Wireless Sensor Networks,* T. Znati et al., eds., 2004.

[7] S.P. Miller, C. Neuman, J.I. Schiller, and J.H. Saltzer, "Kerberos Authentication and Authorization System," *Project Athena Technical Plan,* section E.2.1, 1987.

[8] J. Kohl and C. Neuman, "The Kerberos Network Authentication Service (V5)," RFC 1510, Sept. 1993, ftp://ftp.internic.net/rfc/rfc1510.txt, ftp://ftp.math.utah.edu/pub/rfc/rfc1510.txt.

[9] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J.D. Tygar, "SPINS: Security Protocols for Sensor Networks," *Proc. Seventh Ann. Int'l Conf. Mobile Computing and Networks (MobiCom 2001),* pp. 189-199, July 2001.

[10] R. Blom, "An Optimal Class of Symmetric Key Generation Systems," *Advances in Cryptology: Proc. Eurocrypt '84,* pp. 335-338, 1984.

[11] C. Blundo, A.D. Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung, "Perfectly-Secure Key Distribution for Dynamic Conferences," *Proc. Conf. Advances in Cryptology (Crypto '92),* E.F. Brickell, ed., pp. 471-486, 1992.

[12] R. Merkle, "Protocols for Public Key Cryptosystems," *Proc. IEEE Symp. Research in Security and Privacy,* pp. 122-134, Apr. 1980.

[13] J. Lee and D. Stinson, "Deterministic Key Predistribution Schemes for Distributed Sensor Networks," *Selected Areas in Cryptography,* 2004.

[14] S. Camtepe and B. Yener, "Combinatorial Design of Key Distribution Mechanisms for Wireless Sensor Networks," *Proc. Ninth European Symp. Research Computer Security,* 2004.

[15] H. Chan and A. Perrig, "PIKE: Peer Intermediaries for Key Establishment in Sensor Networks," *Proc. 24th Conf. IEEE Comm. Society (Infocom 2005),* Mar. 2005.

[16] C.S. Laboratory, "Secure Hash Standard," pp. 180-182, Aug. 2002.

[17] V. Gligor and P. Donescu, "Fast Encryption and Authentication: XCBC Encryption and XECB Authentication Modes," *Fast Software Encryption (FSE),* M. Matsui ed., Apr. 2001.

[18] C. Jutla, "Encryption Modes with Almost Free Message Integrity," *Proc. Advances in Cryptology—EUROCRYPT 2001,* pp. 525-542, May 2001.

[19] P. Rogaway, M. Bellare, J. Black, and T. Krovetz, "OCB: A Block-Cipher Mode of Operation for Efficient Authenticated Encryption," *Proc. Eighth ACM Conf. Computer and Comm. Security,* Nov. 2001.

[20] J. Newsome, E. Shi, D. Song, and A. Perrig, "The Sybil Attack in Sensor Networks: Analysis and Defenses," *Proc. Third Int'l Workshop Information Processing in Sensor Networks (IPSN),* 2004.
[21] B. Parno, A. Perrig, and V. Gligor, "Distributed Detection of Node Replication Attacks in Sensor Networks," *Proc. IEEE Symp. Security and Privacy,* 2005.

Haowen Chan received the BSc degree in computer science from Stanford University. He is a PhD student at Carnegie Mellon University. His research interests include distributed algorithms for sensor systems and sensor network security.

Virgil D. Gligor received the BSc, MSc, and PhD degrees from the University of California at Berkeley. He has been at the University of Maryland since 1976, and is currently a professor of electrical and computer engineering. Over the past three decades, his research interests ranged from access control mechanisms, penetration analysis, and denial-of-service protection to cryptographic protocols and applied cryptography. He was a consultant to the Burroughs (1977-1981) and IBM (1984-1999) Corporations, and is currently serving on Microsoft's Trusted Computing Academic Advisory Board. He served the profession as the chair of cochair of several conferences and symposia including IEEE Security and Privacy Symposium, Internet Society's Network and Distributed Systems Security Symposium, IEEE Dependable Computing for Critical Applications, and IEEE-ACM Symposium on Reliability in Distributed Software and Databases. He received the outstanding paper award at the 1988 IEEE Symposium on Security and Privacy. He was a member of several US Government INFOSEC Study Groups that set research agendas in information security, and served on a National Research Council panel on information security (1987-1988). He was an editorial board member of *Information Systems* (1984-1994), the *Journal of Computer Security* (1991-2000), and is currently an editorial board member of the *ACM Transactions on Information System Security*, *IEEE Transactions on Dependable and Secure Computing*, and *IEEE Transactions on Computers*.

Adrian Perrig received the BSc degree in computer engineering from the Swiss Federal Institute of Technology in Lausanne (EPFL)and the PhD degree in computer science from Carnegie Mellon University, and spent three years during his PhD degree at University of California at Berkeley. He is an assistant professor of electrical and computer engineering, engineering and public policy, and computer science at Carnegie Mellon University. His research interests revolve around building secure systems and include Internet security, security for sensor networks, and mobile applications.

Gautam Muralidharan received the BE degree in electrical and electronics engineering from the University of Madras, India, in 1997 and the MS degree in computer engineering from the University of Maryland, College Park, in 2004. In 2004, he joined the Identity Management Practice at PricewaterhouseCoppers LLP, Minneapolis. His work includes designing and implementing authentication, access control, and enterprise level user management infrastructures.