Defending against Sybil Attacks in Sensor Networks*

Qinghua Zhang, Pan Wang, Douglas S. Reeves, Peng Ning
Cyber Defense Laboratory, Computer Science Department
North Carolina State University, Raleigh, NC 27695-8207
{qzhang2, pwang3, reeves, pning}@ncsu.edu

Abstract

Sybil attack is a harmful threat to sensor networks, in which a malicious node illegally forges an unbounded number of identities to defeat redundancy mechanisms. Digital certificates are a way to prove identities. However, they are not viable in sensor networks. In this paper, we propose a light-weight identity certificate method to defeat Sybil attacks. Our proposed method uses one-way key chains and Merkle hash trees. The method thereby avoids the need for public key cryptography. In addition, the method provides a means for authentication of all data messages. A variant of this method is presented that has lower computational requirements under certain conditions. The security of each method is analyzed, and is as good or better than previously-proposed approaches, with fewer assumptions. The overhead (computation, storage, and messages) is also shown to be acceptable for use in sensor networks.

1 Introduction

Sensor networking has become an exciting and important technology in recent years. It provides an economical solution to many challenging problems such as traffic monitoring, building safety monitoring and military applications. In a sensor network, thousands of battery-operated sensor devices (also known as *nodes*) which normally have limited computational, communication and storage resources [12], collectively monitor an area. They may transmit their sensor readings to a base station which is much more powerful, and connected to the outside world.

Sensor networks, however, are very vulnerable to security threats or attacks. For example, attackers could deploy unauthorized sensor nodes or take over a sensor node by hacking into it. To defeat such attacks, mechanisms based on redundancy are frequently proposed. That is, multiple nodes may perform a single function in order to cope with failures and errors. Examples of such functions might be data storage and network "health" monitoring.

Redundancy mechanisms are identity-based. They assume that each physical node is distinguished as one entity and presents only one single abstract concept of an identity. Accordingly, they are vulnerable to any method that allows identities to be forged or falsified. Such a method is the *Sybil attack*. In a Sybil attack, a single node illegally presents multiple identities to other nodes in the network by either forging new (false) identities, or stealing legal identities. A *Sybil node* is a misbehaving node's additional identity. Therefore, a single entity may be selected multiple times (based on multiple identities) to participate in an operation that relies on redundancy, thereby controlling the outcome of the operation, and defeating the redundancy mechanisms.

This paper proposes a new identity certificate scheme to defend against Sybil attacks in sensor networks. In this scheme, each sensor node is pre-assigned a unique secret key to derive one-way key chains. Each node is also distributed an identity certificate which associates its identity with its one-way key chains; this is accomplished by means of the Merkle hash tree cryptographic primitive. By employing an interactive node-to-node authentication method, any two nodes then can mutually authenticate each other and ensure the integrity of their communications. Since the proposed schemes are based on symmetric key cryptography, they satisfy the requirement for low power consumption and memory footprint. A variant of the basic method is also proposed which requires less computation under certain assumptions on sensor location.

The rest of this paper is organized as follows. Section 2 describes related work. Section 3 defines the problem more precisely. In section 4, we provide an overview of the approach. Section 5 describes the basic method which uses *identity certificates* and interactive node-to-node authentication based on one-way key chains. Section 6 describes an extended version which takes advantage of node deployment knowledge. Section 7 concludes the paper.

*This work is supported by the National Science Foundation (NSF) under grant CNS-0430223 and by the Army Research Office (ARO) under grant DAAD 19-02-1-0219.

2 Related Work

Douceur [3] first identified the problem of Sybil attacks, in the context of peer-to-peer distributed systems. He pointed out that in the absence of a logically centralized authority, Sybil attacks are always possible, except under extreme and unrealistic assumptions about the resources available to attackers and the coordination among entities. Karlof and Wagner [7] discussed the threat of Sybil attacks to sensor networks. They suggested using symmetric key cryptography to defeat Sybil attacks. In their approach, each node shares a unique symmetric key with a base station. Each pair of nodes that wishes to communicate then uses a Needham-Schroeder-like protocol to establish a shared key, via the base station. From this point, the two nodes can verify each other's identity by means of their established pairwise key. This approach is not scalable since the base station is a potential bottleneck.

Newsome et al. [11] proposed two methods of defending against Sybil attacks in sensor networks: (1) radio resource testing, and (2) random key pre-distribution. The first approach assumes each sensor node has only one radio, and cannot send and receive simultaneously on more than one channel. Therefore, by challenging a neighbor node (several times) on a radio communication channel which is exclusively assigned to this neighbor, a sensor node can detect Sybil nodes with a certain probability. However, how a sensor node assigns the radio channels to its neighbor nodes is an unsolved problem. In addition, this testing process may consumes a lot of battery power. The random key distribution approach associates the node's identity with the keys assigned to the node from a key pool [6]. Thus, a node's identity can be proved by verifying part or all of the keys that this node claims to have. A drawback of this approach is that if some keys are compromised, the adversaries may be able to falsely claim the identities of many non-compromised sensors. This drawback has been eliminated by recent work on unique random pairwise key establishment schemes [9, 5], based on *t-degree* polynomials. In this work, however, the choice of the threshold t is a challenge. If t is too small, the attacker only needs to compromise a small percent of sensor nodes to compromise the whole network. If t is too big, the storage overhead for each sensor node will be very high.

The identity-based encryption scheme of Boneh and Franklin[2], or any public key-based certificate schemes, can also be used to defend against Sybil attacks. However, these schemes rely on asymmetric cryptography. They require higher computational capability than may be available in a tiny sensor node [12]. Thus, they are not suitable for sensor networks.

3 Problem Statement

Network Assumptions. Our model of the sensor network is as follows. The network is potentially large and may consist of thousands of sensor nodes and is deployed by a single authority. There is a powerful setup server to pre-configure sensor nodes. Sensor nodes use wireless communication. Once deployed, each node is static and can only directly communicate with a fixed and limited number of neighboring nodes.

Sensor nodes and the wireless communication links are not secure. Attackers can inject or intercept messages; they can block the wireless communication links; they can hack into legitimate sensor nodes and control them to attack the network. Attackers can even deploy malicious nodes, but the fraction of them will be only a small percentage of the overall network.

Security Goals. In this paper, our goal is to detect Sybil attacks in sensor networks. Messages generated by Sybil nodes can therefore be disregarded, and the attempted Sybil attack can be reported for possible further corrective action.

To detect Sybil attacks, we should be able to securely verify the unique identity of each node. Ideally, once this has been done, a node should be able to verify the integrity and authenticity of messages received from any other node. If this goal is not achievable due to high costs (computation, memory, or communication), then a secondary goal is to allow each node to directly verify the identity of nodes in its vicinity, and indirectly verify the identity of more distant nodes.

4 Overview of the Proposed Method

We propose to utilize identity certificates to defend against Sybil attacks. The basic idea is very simple. The setup server, before deployment, assigns each sensor node some unique information. The server then creates an identity certificate binding this node's identity to the assigned unique information, and downloads this information into the node. To securely demonstrate its identity, a node first presents its identity certificate, and then proves that it possesses or matches the associated unique information. This process requires the exchange of several messages.

We use the Merkle hash tree [10] as the basic means of computing identity certificates. The *Merkle hash tree* is a vertex-labeled binary tree, where the label of each non-leaf vertex is a hash of the concatenation of the labels of its two child vertexes. The *primary path* of a leaf vertex is the set of vertexes on the path from the leaf to the root of the tree. The *authentication path* consists of the siblings of the vertexes on this primary path. Given a vertex, its authentication path, and the hash function, the primary path can then be computed, up to and including the root of the tree. This

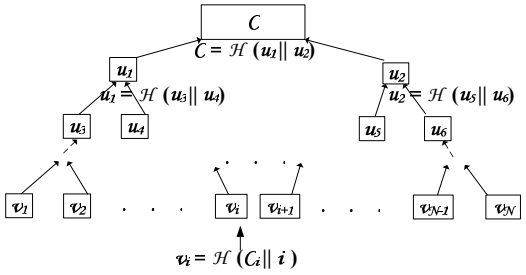

Figure 1. An example identity certificate tree. $\langle v_1, \{v_2, \cdots, u_4, u_2\}\rangle$ is an identity certificate $IDCert_1$ for node 1.

computed value of the root can then be compared with a stored value, to verify the authenticity of the label of the leaf vertex.

Our method has the following steps. For clarity, we use symbol *i* to represent the identity of the ith sensor node.

1. The setup server pre-distributes to each node *i* some secret and unique information I_i.

2. The setup server computes information *commitment* C_i for node *i*. The means of computing C_i from I_i must be computationally infeasible to invert, i.e., it must be difficult to compute I_i given C_i.

3. The server constructs a Merkle hash tree in which each leaf vertex corresponds to one node in the sensor network, and is labeled with its hash value, e.g. $\mathcal{H}(C_i\|i)$, where $\|$ stands for the concatenation operator. Figure 1 is an example.

4. The server pre-distributes to each node i the identity certificate for that node, denoted $IDCert_i$, and the label of the tree's root. $IDCert_i$ consists of the label of the leaf corresponding to that node along with its authentication path, denoted as $\text{AuthenticationPath}_i$.

5. After deployment, the ith node can prove its identity to another node j on demand. This requires the ith node to present its certificate to j, and to demonstrate it possesses the information I_i from which C_i was computed. This demonstration must also be unique to j, so that it cannot be reused or replayed between another pair of nodes.

5 The Basic Method

In this section, we describe our basic method in detail. The secret information assigned by the setup server to node i is a unique secret key which can be used to generate one-way key chains. From these key chains a *low-level* Merkle hash tree can be computed by node i (as described below). The label of the root of this low-level tree is the value of the commitment C_i.

Our method of creating certificates combines Merkle hash trees with one-way key chains. A *one-way key chain* [8] $(k_0, k_1, .., k_{l-1}, k_l)$ is an ordered list of cryptographic keys generated by iteratively applying a one-way hash function $\mathcal{H}$ to a randomly selected value or *key seed* k_l. That is, for $u = l \ldots 1$, $k_{u-1} = \mathcal{H}(k_u)$. Therefore, any key k_u is a commitment to all subsequent keys k_v, $v > u$; more specifically, k_0 is a *key chain commitment* for the entire chain. l is the length of the key chain and is a system parameter.

The setup server first pre-distributes to each node *i* a key seed, denoted $K_{i,l}$, as its unique information. Node *i* uses this to generate $N-1$ key seeds, where N is the number of sensor nodes in the network. The jth key seed for node i, denoted $K^j_{i,l}$, is simply computed as $K^j_{i,l} = K_{i,l} + j$. From each such key seed, node i can then generate a one-way key chain, with $K^j_{i,0}$ as the key chain commitment for node i's jth key chain. Node i uses a different key chain with a different node in the network. This waives time synchronization as an application condition as for μTESLA [12] (which also uses a one-way key chain for authentication).

The setup server computes the value of C_i for node i from the set $\{K^j_{i,0}\}$, $j = 1 \ldots N$. It does this by creating a *low-level* Merkle hash tree for node i. In this low-level tree for node i, the jth leaf is labeled with $K^j_{i,0}\|j$. The low-level tree is then constructed as previously described. The label of the root of this low-level tree for node i constitutes C_i. There are N such trees, one for each node in the network.

After computing the value of C_i for $i = 1 \ldots N$ in the fashion described above, the setup server can compute the single *high-level* Merkle hash tree. This high-level tree has leaves labeled $\mathcal{H}(C_i\|i)$, and corresponds to the Merkle hash tree described in the previous section. Figure 2 illustrates these two levels of trees. In this figure, only one of the low-level trees is shown (for node 4), and the single high-level tree is shown. The circles reflect some authentication paths

Every node i is given its identity (or high-level) certificate, $IDCert_i$, by the setup server. This certifies node *i*'s identity and its information commitment C_i. For the jth leaf in node i's low-level tree, there is an authentication path to the root of this low-level tree, denoted $AuthenticationPath^j_i$. Node i can create a *low-level certificate* for node j, denoted $IDCert^j_i$, $IDCert^j_i = \langle K^j_{i,0}\|j, AuthenticationPath_{i,j}\rangle$ as needed. This certifies the key chain commitment $K^j_{i,0}$ node *i* uses specifically for node *j*. For example, for node 4 in figure 2, $IDCert_4 = \langle v_4, \{v_3, u_3, u_2\}\rangle$. And node 4 uses $IDCert^2_4 = \langle K^2_{4,0}\|2, \{K^1_{4,0}\|1, m_4, m_2\}\rangle$ with node 2.

We now describe the protocol by which a node proves its identity on demand, using the above information. This protocol is similar to the interactive guy fawkes protocol [1], originally designed for authentication by computationally-limited devices in ad hoc wireless networks. The essential

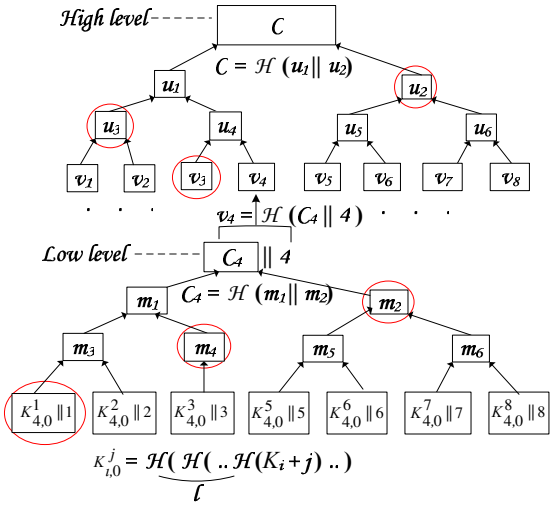

Figure 2. A one-way key chain and merkel-hash tree based Identity Certificate Tree with 8 nodes.

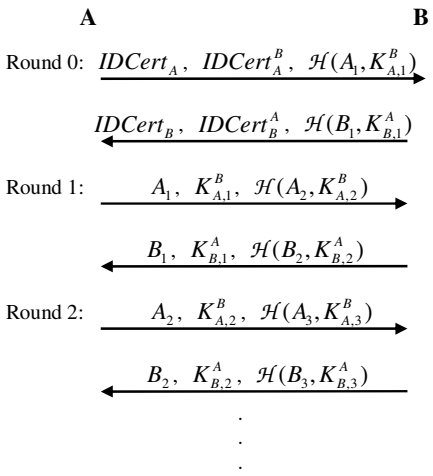

Figure 3. Interactive node-to-node authentication. A_r(B_r) is a messages sent by A (resp. B) in round r.

difference of our method is that our protocol does not require the use of a location-limited side channel and each node i will use keys from its jth one-way key chain as authenticators, rather than randomly-generated values.

Figure 3 is a simple illustration. Two nodes *A* and *B* begin communicating in round 0 by exchanging their identity certificates (*IDCert*$_A$, *IDCert*$_A^B$ and *IDCert*$_B$, *IDCert*$_B^A$, respectively) together with the hashes of their first messages ($\mathcal{H}(A_1,K^B_{A,1})$, $\mathcal{H}(B_1,K^A_{B,1})$). *B* can verify $\mathcal{H}(C_A\|A)$ included in *IDCert*$_A$ with respect to the publicly known tree root *C*. And then it can verify $K^B_{A,0}$ provided by *IDCert*$_A^B$ with respect to C_A.

$K^B_{A,1}$ is the first key of *A*'s one-way key chain, e.g $K^B_{A,0}$ = $\mathcal{H}(K^B_{A,1})$. $K^B_{A,1}$ serves as the authenticator of *A*'s first message A_1, since only *A* knows the secret $K^B_{A,1}$ before it is released. Likewise A can verify the first message from B. *A* and *B* continue to exchange messages authenticated with successive keys in their key chains. In round *r*, *A* reveals the message A_r, shows the authenticator $K^B_{A,r}$ and gives the next message's hash $\mathcal{H}(A_{r+1}, K^B_{A,r+1})$, and B does the same. *B* can verify A_r and $K^B_{A,r}$ immediately since it received $\mathcal{H}(A_r, K^B_{A,r})$ and $K^B_{A,r-1}$ in the previous round. In this approach, both *A* and *B* prove their identities, and prove the authenticity of the messages they exchange at each step. If B fails to respond when A sends a message in the rth round, A can resend the entire message, or just $\mathcal{H}(A_{r+1}, K^B_{A,r+1})$. A won't disclose $K^B_{A,r+1}$ until it receives a valid $K^A_{B,r}$ from B.

Security Analysis. The proposed method provides a way to build initial trust between a pair of nodes. The high-level certificate allows node i's identity to be proved to any other node. A node creating a false identity will not be able to easily forge an identity certificate for which C is the commitment, according to the properties of the Merkle hash tree. The low-level identity certificate makes this proof specific to a single receiving node j. Any other node k receiving this same low-level certificate will be able to determine the low-level certificate is not intended for it. Receiving node j is expected to record the fact that it has received an identity certificate from node i; another node presenting or replaying i's identity certificate can easily be detected.

The low-level certificate that node i sends to node j certifies node i's one-way key chain commitment for node j. The use of the one-way key chain authenticates each message exchanged between nodes i and j after this initial exchange of identity certificates.[1] The disclosure of a key follows it use, and the validity of each key can be determined from the initial, certified key-chain commitment. Key forgeries or reuse are therefore prevented.

Our method does not prevent colluding nodes from sharing keys and posing as each other. This is outside the scope of our method, and we leave it as a future work.

The original interactive guy fawkes protocol was susceptible to a man-in-the-middle attack. In this attack, a malicious node X intercepts a message sent from one party A to another party B. X tampers with or modifies the authenticator in this message before sending it on to B. While B can detect that tampering occurred, it loses synchronization with A, and must resort to the location limited side channel to negotiate a new starting authenticator.

A similar situation with our protocol is illustrated as follows. Malicious node X has tampered with the hash value $\mathcal{H}(A_{r+1}, K^B_{A,r+1})$ in the message sent from A to B in round r. This prevents the contents of the message, A_{r+1}, sent

[1] While the keychain is of a fixed length l, it is not difficult to negotiate a replacement keychain for an existing one before all of its keys have been disclosed; details are omitted.

from A to B in round $r+1$ from being authenticated. In our protocol, a node never discloses a new valid key unless it receives a new valid key from its counterpart node. That being the case, B will refuse to disclose further keys $B_{r+1}, \cdots$ until it receives a verifiable key from A. B further maintains the last verifiably correct key, A_r, received from A, and therefore does not lose synchronization, regardless of how many forged or altered messages are received. Malicious node X is not able to forge a verifiable hash from A, because the key needed to do so is not disclosed by A until the following round, and each key in the sequence is used only once. Further, in the event of a communication link failure, two nodes can resume communication once the link is back to normal, without loss of synchronization.

A malicious node may yet send forged or tampered messages to receivers, just to waste the computation and communication bandwidth of the receiving nodes. The overhead of message verification in our method is low (discussed further below), however, so only a high volume of such messages would have a significant impact. A means of detecting and mitigating such denial of service attacks is outside the scope of this paper.

Cost Analysis. Before the overhead is analyzed, we propose a simple modification of the method to minimize this cost. Previously, it was implied that the setup server downloaded to each node i all of the low-level Merkle hash tree for that node. This would enable node i to construct any one of the $N-1$ low-level certificates needed to communicate with other nodes in the system. This hash tree may, however, require an excessive amount of storage for a resource-limited node that is part of a very large network.

We instead propose that the setup server only download the low-level tree from the root (depth 0) to depth D ($D \in [1, \ldots, \log_2 N]$). Any part of the tree below this that is needed to construct a low-level certificate can be computed "on the fly" by i as needed. This is because the information from which the ith low-level tree is computed, $K_{i,l}$, is downloaded to i by the setup server. The parameter D provides a means to flexibly trade off the amount of storage and the amount of computation needed by any sensor node to apply the method we have proposed.

Figure 2 illustrates this process. Suppose D is 1, and node 4 needs the low level certificate $\langle K^2_{4,0} \| 2,\ \{K^1_{4,0} \| 1,\ m_4,\ m_2\}\rangle$ in order to begin communication with node 2. The setup server initially downloads the three hash values C_4, m_1, m_2 to node 4. Node 4 determines that $K^1_{4,0} \| 1$ and m_4 are needed to complete the low level certificate authentication path. It must therefore compute $K^1_{4,0}$ and $K^3_{4,0}$ to derive these values.

The overhead, with this modification, now is summarized in figure 4. Let c represent the amount of storage needed for one hash value and s represent the number of maximum number of neighbors that a node actively communicates with. Due to page limit, we only briefly discuss the first two items. The storage overhead associated with the basic scheme mainly includes one high level certificate and parts of its low certificates and as many one-way key chains as there are nodes with which it actively communicates. To compute an arbitrary identity certificate on the fly, each node i needs to derive all the labels of leaf vertexes used to derive the unknown part of its AuthenticationPath$_i$ and then derive the AuthenticationPath$_i$ accordingly.

Overhead	Basic Method	Extended Method
Storage (Bytes)	$c\cdot\log N + c\cdot 2^{D+1} + s\cdot l\cdot c$	$c\cdot\log_2 N + c\cdot 2^{D+1} + s\cdot l\cdot c + \min(m-g, 9g-m)\cdot\log_2 g$
Computation for Node (# of hash)	$\frac{(l+1)\cdot N}{2^D} + D - l - \log_2 N - 1$	$\frac{(l+1)\cdot m}{2^D} + D - l - \log_2 m - 1$
Computation for Server (# of hash)	$(l+1)\cdot N^2 + N - 1$	$(l+1)\cdot mN + N - 1$
Direct Comm. for certificate (Bytes)	$2c\cdot\log_2 N$	$c\cdot(\log_2 N + \log_2 m)$
Indirect Comm. for certificate (Bytes)		$25c\cdot t + 2c$

Figure 4. Overhead.

As we can see, the computation cost for the server is $O(N^2)$ and $O(N)$ for each node (although the coefficient $\frac{l+1}{2^D}$ may be very small). The overhead is acceptable when N is modest, e.g. N = 1000. However, it may take a very long time for the server to compute the two-level hash tree, for large N, e.g., when $N = 10^6$. Also it is not practical to require that each sensor node do large numbers of hash operations to generate each low-level certificate. The basic method as presented therefore has scalability problems.

Summary. This basic method has most of the qualities we desire. It can provide full scale node-to-node authentication in the network by employing unique one-way key chains between each pair of nodes. The method is robust against man-in-the middle attacks, and does not require clock synchronization between nodes (unlike μTESLA [12]). It can defeat Sybil attacks launched by non-colluding nodes without using expensive public key operations. The drawback just described is non-scalability, due to computational overhead. This is a result of the generality of our assumptions. Under more limiting assumptions about communication patterns, the computational overhead can be significantly reduced. The next section describes such a method, and analyzes the results.

6 Extended Version

In a sensor network, sensor nodes are used to collect data which are delivered to and integrated at a data sink node or base station. These are the major communication patterns. Most communications occur between neighboring nodes and are localized. In this section, we propose an extended version which assumes deployment knowledge can be approximately available to reduce the overhead associ-

ated with the basic method. Deployment knowledge was previously used in key management in sensor network [4].

We assume a group-based deployment model and Gaussian deployment distribution similar to that stated in [4]. The *target field*, or the field of interest, is a two dimensional area and is partitioned into small regular areas (e.g. rectangle) which can be represented by their center's coordinates (L_x,L_y). Sensor nodes are deployed in groups and each group is deployed in one different area following a Gaussian distribution. We assume the pdf $f(x,y)=\frac{1}{2\pi\sigma^2}e^{-\frac{(x^2+y^2)}{2\sigma^2}}$ of each group is identical. The setup server knows the group that a node belongs to, and thereby knows the node's *expected area* or *expected location*. A node has a signal range *R*. After deployment, two nodes are *neighbors* if they are located within each other's signal range.

Description of the Extended Version. The extended version continues to use two-level identity certificates and the interactive node-to-node authentication protocol to defend against Sybil attacks as the basic version. The difference is that each low level certificate tree is now computed based on at most m selected nodes instead of the whole network, where $m \ll N$. Each node then has at most m low level certificates.

This extension reduces the computational cost for both nodes and the setup server. On the other hand, it sacrifices the capability of direct verification and authentication between any pair of nodes. The impact of this sacrifice is a function of the accuracy of deployment knowledge, and the value m. Under favorable conditions, nodes that communicate will have the appropriate certificates.

We first investigate how the m predicted neighbors should be chosen. The setup server selects the *m* nodes whose expected locations are closest to that of node *i*. This consists of a randomly selected set of nodes in i's group. If m is larger than the size of i's group, denoted g, the remaining $m-g$ nodes are selected uniformly and randomly from the areas adjacent to i's area. There are at most 8 such adjacent areas. The setup server then computes the low level certificate tree for node i based on these selected nodes using the basic method described previously.

If two neighbor nodes don't hold certificates for each other, they must first authenticate their key chain commitments through a *multi-path* message authentication process. Following that, they can employ the basic interactive node-to-node authentication scheme to authenticate all future data messages.

A sketch of this indirect authentication is as follows. First, two nodes *i* and *j* each generate a one-way key chain, using the secret keys which were pre-distributed to them ($K_{i,l}$ and $K_{j,l}$, respectively). From this i can generate a one-way key chain for j with commitment $K^j_{i,0}$, and j can do likewise for i. Second, one of the nodes (assume i) broadcasts a message to its neighbors asking which node

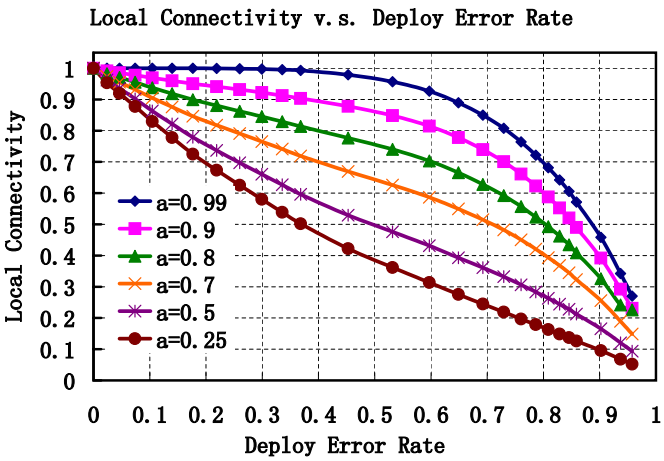

Figure 5. Local Connectivity v.s. Deployment error rate.

can authenticate both node i and j's messages (i.e., this neighbor must hold certificates for both *i* and *j*). Any neighboring nodes which can serve this purpose reply to node *i*. Third, the originator *i* randomly chooses *t* nodes from these replies and sends the commitment $K^j_{i,0}$ to them, and these *t* nodes in turn send $K^j_{i,0}$ to node *j*. Node *j* then sends its key chain commitment $K^i_{j,0}$ to node *i* in the same way and in the reverse order. Each message exchanged is authenticated by applying the interactive node-to-node authentication method. Finally, if both nodes get more than $\frac{t}{2}$ identical commitments from the other, the two nodes can employ the interactive node-to-node authentication scheme in their next communication. The requirement that multiple neighbors must assist in this process, with identical results, is to reduce the likelihood of man-in-the-middle attacks.

Evaluation. We simulated the local connectivity under different deployment error rates. The local connectivity is the probability of two neighboring nodes having certificates for each other. The error rate is the probability a node is not located in its expected area. We use similar configuration to that of [4]. In addition, we configure each group has the same number of group members and set it to be g = 100.

Figure 5 shows the simulation results. Each curve represents a result for a different m value, and $a = \frac{m-g}{8g}$ (i.e., the probability that node i has a certificate for a random node in a neighboring area). As expected, performance improves as m and a increase, and as the deployment error rate decreases. When the deployment error rate is less than 10%, nearly all the local connectivity values are above 90%. When a is high, the connectivity can be close to 100%.

Cost Analysis. As seen from figure 4, the computational cost for each node to re-compute an identity certificate is substantially reduced from $O(N)$ to $O(m)$, $m \ll N$. The cost for the server is also reduced greatly from $O(N^2)$ to $O(mN)$. The overhead of an exchanging identity certificates

is reduced $c{\cdot}(\log_2 N - \log_2 m)$ bytes.

Although the storage cost increases $min(m-g, 9g-m){\cdot}\log_2 g$ bytes for memorizing the identities for which a node has or hasn't low identity certificates, this additional cost can be potentially evaluated to be 0. The case appears when m is close to $9g$ and the local connectivity is high accordingly. $\log_2 g$ is the number of bytes needed to represents an identity within a group. The message overhead associated with indirect authentication process includes a broadcast message ($\leq 2c$ bytes), up to t reply messages ($\leq c$ bytes each), and up to $8t$ data messages ($3c$ bytes each) exchange. The indirect communication may happen less frequently if the local connectivity is very high.

Impact on Security. This extended version achieves scalability at the expense of some increase in complexity, and a potential reduction in security. An attacker can exploit the indirect authentication process to falsely claim the identity of another node. In this scheme, at least $\frac{t}{2}$ malicious nodes must collude to accomplish this. In the following, let E represent the number of malicious nodes, p represent the local connectivity and d represent the number of neighbors of a node. We use the term *verifier* to be the node checking whether another node is a Sybil node.

Claim 1. *A Sybil node has a low probability of proving its (false) identity if* $\frac{E}{N} < \frac{p}{1+p}$.

Proof. For a Sybil node i to prove its identity, the verifier must indirectly authenticate it through more than half of the nodes that respond to i's broadcast. The number of non-compromised nodes that won't verify i's identity is $d{\cdot}p^2{\cdot}(1-\frac{E}{N})$. The number of compromised nodes which may verify i's identity is $d{\cdot}\frac{E}{N}{\cdot}p$. For the Sybil node's identity to be proved, it is necessary that $d{\cdot}\frac{E}{N}{\cdot}p > d{\cdot}p^2{\cdot}(1-\frac{E}{N})$. Simplifying, $\frac{E}{N} > \frac{p}{1+p}$. $\frac{p}{1+p}$ is an increasing function of p. The higher the probability that two neighboring sensor nodes each have certificates for the other, the greater the number of sensor nodes an attacker needs to compromise to (falsely) prove the Sybil node's identity. □

In our previous simulation, the local connectivity can be as high as almost 100%. In this case, attackers need to compromise more than 50% of the sensor nodes to successfully launch a Sybil attack. If more than 50% sensor nodes could be compromised, defending against Sybil attacks becomes meaningless. From this sense, we can conclude this extended version sacrifices little in the way of security.

7 Conclusions

In a Sybil attack, a single node illegally presents multiple identities to other nodes. This paper proposes a method of defending against such attacks in sensor networks, The method uses a two-level Merkle hash tree to create certificates. An interactive node-to-node authentication protocol employs one-way key chains for message authentication. This method does not require clock synchronization between nodes, and is robust against man-in-the middle attacks. An extension of this method exploits node deployment knowledge to reduce the computational overhead at each node. This extended method is substantially more scalable, with a small reduction in the security it provides.

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

References

[1] D. Balfanz, D. Smetters, P. Stewart, and H. C. Wong. Talking to strangers: Authentication in ad-hoc wireless networks. In *Proceedings of NDSS'02*, Feb 2002.

[2] D. Boneh and M. Franklin. Identity based encryption from the weil pairing. In *Proceedings of Crypto 2001 (LNCS 2139), pp. 213-229*, May 2001.

[3] J. R. Douceur. The sybil attack. In *First International Workshop on Peer-to-Peer Systems (IPTPS'02)*, Mar 2002.

[4] W. Du, J. Deng, Y. S. Han, S. Chen, and P. K. Varshney. A key mangagement scheme for wireless sensor networks using deployment knowledge. In *Proceedings of the IEEE INFOCOM'04*, Mar 2004.

[5] W. Du, J. Deng, Y. S. Han, and P. K. Varshney. A pairwise key pre-distribution scheme for wireless sensor networks. In *Proceedings of the CCS'03*, Oct 2003.

[6] L. Eschenauer and V. D. Gligor. A key-management scheme for distributed sensor networks. In *Proceedings of the CCS'02*, Nov 2002.

[7] C. Karlof and D. Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. In *First IEEE International Workshop on Sensor Network Protocols and Applications*, May 2003.

[8] L. Lamport. Password authentication with insecure communication. In *Communications of the ACM, 24(11):770-772*, Nov 1981.

[9] D. Liu and P. Ning. Establishing pairwise keys in distributed sensor networks. In *Proceedings of the CCS'03*, Oct 2003.

[10] R. C. Merkle. Protocols for public key cryptosystems. In *Proceedings of the 1980 IEEE Symposium on Security and Privacy*, 1980.

[11] J. Newsome, R. Shi, D. Song, and A. Perrig. The sybil attack in sensor networks: analysis & defenses. In *Proceedings of 3rd International Symposium on Information Processing in Sensor Networks (IPSN ?4)*, April 2004.

[12] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. Tygar. Spins: Security protocols for sensor networks. In *Wireless Networks Journal (WINET)*, Sep 2002.