SET: Detecting node clones in Sensor Networks

Heesook Choi, Sencun Zhu, Thomas F. La Porta
Department of Computer Science and Engineering
The Pennsylvania State University
E-mail:{hchoi,szhu,tlp}@cse.psu.edu

Abstract— **Sensor nodes that are deployed in hostile environments are vulnerable to capture and compromise. An adversary may obtain private information from these sensors, clone and intelligently deploy them in the network to launch a variety of insider attacks. This attack process is broadly termed as a *clone attack*. Currently, the defenses against clone attacks are not only very few, but also suffer from selective interruption of detection and high overhead (computation and memory). In this paper, we propose a new effective and efficient scheme, called *SET*, to detect such clone attacks. The key idea of *SET* is to detect clones by computing set operations (intersection and union) of exclusive subsets in the network. First, *SET* securely forms exclusive unit subsets among one-hop neighbors in the network in a distributed way. This secure subset formation also provides the authentication of nodes' subset membership. *SET* then employs a tree structure to compute non-overlapped set operations and integrates interleaved authentication to prevent unauthorized falsification of subset information during forwarding. Randomization is used to further make the exclusive subset and tree formation unpredictable to an adversary. We show the reliability and resilience of *SET* by analyzing the probability that an adversary may effectively obstruct the set operations. Performance analysis and simulations also demonstrate that the proposed scheme is more efficient than existing schemes from both communication and memory cost standpoints.**

I. INTRODUCTION

Unlike traditional network equipment, sensors have very limited resources: low processing capability, small memory size, and limited power supplies. Many sensor networks are self-configured with no centralized control. This enables unattended sensor deployment into inaccessible and hostile environments. These environments, however, often make sensors susceptible to capture and compromise. Once a sensor is compromised, the information inside is easily accessible. An adversary may replicate captured sensors and deploy them in the network to launch a variety of insider attacks. This attack process is broadly termed as a *clone attack*.

Several researchers have discussed attacks that may be launched using cloned nodes [15], [1], [4]. Since a cloned node has legitimate information (codes and key materials), it may participate in network operations in the same way as a non-compromised node. It can then launch a variety of attacks. For example, it may create a *black hole*, initiate a *wormhole attack* [7] with a collaborating adversary, inject false data [14] or incorrectly aggregate data to bias results [13], [14]. Cloned nodes may also leak data in an environment in which sensed data must be kept private.

A simple way to address clone attacks is to have every sensor send an authenticated report of itself and its neighbors to a base station using an existing data forwarding protocol. The symmetric key shared between the base station and sensor is used to authenticate the report. However, this simple method incurs high communication overhead due to the redundant information being reported. Furthermore, this approach is vulnerable to attacks in which a compromised node selectively drops reports on the way to the base station. Researchers have also proposed using witness-based schemes [11], but these rely on public key cryptography which may not be practical for sensor networks.

To overcome the limitations of existing solutions, we propose an effective and efficient scheme to detect clone attacks. We leverage the following observations to form the basis of our detection scheme. A sensor network can be modeled as a set of non-overlapping subregions. All nodes in the network have unique identifiers. Sensor nodes in each subregion form an exclusive subset. Since node identifiers are unique, the intersection of any two subsets should be empty. If an adversary replicates nodes and deploys them in the network, the intersection of subsets including these replicated nodes will not be empty, and hence a clone attack can be detected. We call this solution *SET*.

The first research challenge is to form exclusive subsets in a secure way. To address this challenge, we propose an algorithm in which an exclusive subset is securely formed among one-hop neighbors in a distributed way, while providing the authentication of nodes' subset membership. To efficiently and reliably compute set operations on these subsets, we employ a tree structure to compute non-overlapped set operations and integrate an interleaved authentication technique into the set operations on the tree to preserve the reliability of set operation results. We further fortify *SET* with randomization, which makes the exclusive subset and tree formation unpredictable to the adversary as well as makes the system more communication and memory efficient.

We provide a security analysis of *SET* considering both individual and colluding attackers. The analysis shows that attackers may only be effective in a very limited number of scenarios. Further, we show that these scenarios are very unlikely to occur. We also analyze the communication overhead of *SET* and verify the analysis by simulation. The analysis and simulation results confirm that *SET* is an appealing solution in sensor networks due to its efficiency and low communication overhead, while providing trustworthy detection of node replication.

II. NETWORK AND THREAT MODEL

A. Network Model and Assumptions

We deal with a sensor network which consists of a base station (BS) and a large number of low-end sensors. We model the wireless sensor network as an undirected graph $G = (V, E)$ where V and E are a set of nodes and edges, respectively. We use a unit disk graph model so that there exists an edge between nodes u and v, $(u, v) \in E$, if the Euclidean distance of u and

v satisfies $|u - v| \leq 1$. We assume that the sensor network is a connected graph, i.e., there exists a path between any two sensor nodes.

We assume that the base station has information (e.g., key materials) of all deployed sensors. Each sensor has a unique key shared with the base station which can be used for computing authentication codes or encrypting data to the base station. In addition, sensors are able to establish a pairwise key with other peers to support secure peer-to-peer communication. We employ an id-based pairwise key establishment scheme such as [3], [6], [8], [15] in which the keying material of a node is bound to its identity. Thus a node cannot lie about its id to neighbors, and a node knows the authenticated ids of all its neighbors. We also assume that a broadcast authentication protocol such as μTESLA [12] can be used to authenticate packets broadcast by the base station. We assume that the communication between the base station and sensors is made reliable by using one of many retransmission techniques.

B. Threat Model

We consider a setting in which the capability of the adversary is bounded such that only a limited number of sensors are compromised. Compromised nodes are totally in control of the adversary. An adversary may capture a few sensors, copy the information into its own sensors, and deploy the clones in places that are intelligently decided. Since cloned nodes have authenticated information extracted from the compromised sensors, they can be involved in network operations and launch various insider attacks.

The adversary may try to conceal the existence of cloned nodes. To conceal their existence, the adversary may interfere with the detection algorithm. For example, if sensors are required to report their presence (identifier) regularly, cloned nodes may not participate unless there exists a scheme to monitor sensors' regular reports which is difficult in sensor networks. An adversary may also drop or manipulate the reports of others that they are forwarding. Cloned nodes may also collaborate by eliminating cloned identifiers from reports.

III. SET: Clone Detection in Sensor Networks

In this section, we present a dependable and resilient defense scheme, *SET*, to detect clone attacks, i.e., to detect duplicated sensors. *SET* consists of five components: exclusive subset construction, authentication of subset covering, distributed set computation and interleaved authentication on subset trees, and verifiable random selection which further optimizes *SET*.

As described in the introduction, our solution is based on a set model of a sensor network in which exclusive subsets are formed and a report of each subset is transmitted to the base station.

Due to the typical random deployment of sensors, it is challenging to construct exclusive subsets in the network. We first present an *Exclusive Subset Maximal Independent Set* (ESMIS) algorithm by which exclusive subsets are formed in a distributed way in the network (subsection III-A). To ensure secure subset construction in a network with compromised nodes, we propose to integrate the ESMIS algorithm with an authentication scheme (subsection III-B). We optimize *SET* by applying randomization to the exclusive subset formation, without losing security and exclusiveness (subsection III-C). To perform efficient and reliable set computation in the network, we propose a multiple tree based set computation scheme so that intersection and union of subsets can be efficiently computed (subsection III-D). Finally, we present an interleaved authentication scheme, to preserve the reliability of set computation on a tree (subsection III-E). In the following subsections, we detail these components of *SET*.

A. Exclusive Subset Construction

The first component of our system creates exclusive subsets in the network. Each of these subsets will have a subset leader (SLDR). The SLDR reports the members of its subset and itself; if all reported subsets have no intersection, there are no clones.

A trivial way of constructing exclusive subsets is for each node to be a subset. Each node sends a report of itself either to the base station or to randomly selected nodes using various protection mechanisms [11]. This will incur high communication overhead. To reduce this cost, we can define a subset that covers nodes over multiple hops. However, it is difficult to ensure that every node within a defined number of hops is exclusively in a subset.

To reduce the communication overhead of a single node subset algorithm, and to ease the security design in a multi-hop covering subset, we propose to form a secure and exclusive subset. The scope of the subset is the transmission range of a node. The algorithm has two parts. First, set leaders (SLDRs) are determined such that no two SLDRs are within transmission range. Second, nodes that are not a SLDR associate with a single SLDR to form exclusive subsets.

This setting exactly matches the maximal independent set (MIS) problem [9]. By definition, in a graph $G = (V, E)$, a maximal independent set is a subset of V such that (1) there is no edge between nodes in the MIS, and (2) other nodes not in the MIS have at least one neighbor in the MIS [9]. However, the existing MIS algorithms are designed without considering security.

Protocol Description. Now, we present a secure distributed MIS approach to form exclusive subsets, preventing a specific node from being the target of an adversary. We use members of the MIS as SLDRs. Property (1) of a MIS guarantees that the SLDRs have no edges between them. The second property allows us to easily form an exclusive subset as we describe below.

We now describe the distributed MIS algorithm, *ESMIS* shown in Algorithm 1. We define three states (*Init*, *Ruler*, *Ruled*) of a node. The base station initiates detection by generating a random *seed* and broadcasting it to the network. On receiving the *seed*, every node sets its initial state as *Init*.

The basis of the ESMIS algorithm is a hash function $H_1 : (seed|x) \rightarrow y \in [1..d]$, where d is the average degree of a node in the network and x is a node ID. Since each node has a list of neighbors, a node locally computes H_1 for itself and its neighbors. The node having the largest result, H_{max}, among neighbors becomes a SLDR and changes its state to *Ruler*. If there exists more than one node with the same H_{max}, the node having the biggest identifier will be the SLDR. The SLDR announces itself by sending an ANN(ounce) message.

Other nodes that have a value smaller than H_{max} wait for an ANN message. If a node in *Init* state receives an ANN message,

Algorithm 1: ESMIS Algorithm (Code of node v)

Input: $N_v = \{v_1, v_2, ..., v_k\}$ // a set of neighbors of v;
   $seed$, $State = Init$, $SLDR = 0$
Procedure:
1: $H_v = H_1(seed|v)$;
2: $H_{n,max} = \max\{M_i = H_1(seed|v_i)|$for all $v_i \in N_v\}$;
3: $H_{max} = \max\{H_{n,max}, H_v\}$;
4: **if** ($H_{max} = H_v$) **then**
5: $State = Ruler$; Broadcast an ANN message; Stop;
6: **endif**
7: Start WAIT timer;
8: **while** State = Init **do**
9: **if** State = *Init* and COV messages received from all neighbors **then**
10:   $State = Ruler$; Broadcast an ANN message; Stop;
11: **endif**
12: **if** State = Init and ANN message received from v_i **then**
13:   $State = Ruled$; $SLDR = v_i$; send a COV message to neighbors; Stop;
14: **endif**
15: **if** WAIT timeout and there are at least one neighbor in *Init* **then**
16:   **if** H_v is the largest among these *Init* state neighbors **then**
17:    $State = Ruler$; Broadcast an ANN message; Stop;
18:   **else**
19:    Start WAIT timer;
20:   **endif**
21: **else**
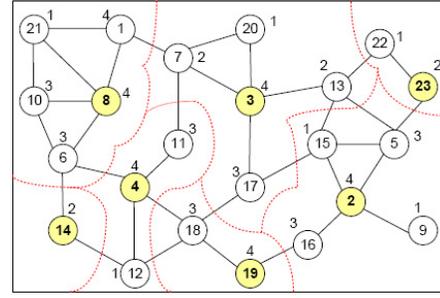22: $State = Ruler$; Broadcast an ANN message; Stop;
23: **endif**
24: **endwhile**

Fig. 1. An example network with seven subsets generated by the ESMIS algorithm

the node changes its state to *Ruled* and saves the SLDR identifier in the ANN message. This implies that the node becomes a member of the subset in which the SLDR node is the head. When a node transitions into the *Ruled* state it sends a "covered" message (COV) to its neighbors which includes its identifier and its SLDR identifier. *A subset is composed of a SLDR node and member nodes covered by the SLDR*. The SLDR node identifier is used to identify the subset.

Figure 1 shows an example with a small network which consists of 23 nodes and has $d = 4$; the value outside the circle of each node is the H_1 computation result. This shows one example of constructing seven subsets according to Algorithm 1. Nodes 2, 3, 4, 8, 14, 19, and 23 are selected as SLDR nodes of the following subsets respectively: $\{2,5,9,15,16\}$, $\{3,7,13,17,20\}$, $\{4,11,12\}$, $\{1,6,8,10,21\}$, $\{14\}$, $\{18,19\}$, $\{22,23\}$.

Since the algorithm is distributed and parallel, there exist race conditions leading to potential conflicts. For example, node 6 is located in an overlapped region and receives ANN messages from nodes 4 and 8. Without loss of generality, node 6 is covered by the SLDR from which it receives the first ANN message and simply ignores any other ANN messages received later.

Now, consider node 14 which is waiting for an ANN message because it does not have H_{max} amongst its neighbors. In this example, 14 only receives COV messages from neighboring nodes 6 and 12, i.e., it does not receive any ANN messages. To resolve this case, if a node that does not have H_{max} receives COV messages from all neighboring nodes, it becomes a SLDR. In this case node 14 becomes a SLDR (lines 9 and 10 in Algorithm 1).

Finally, a deadlock may occur if two or more adjacent nodes are in *Init* state, neither having H_{max}, and SLDR candidate neighbors are covered by other SLDRs out of transmission range of these nodes. Nodes 22 and 23 in Figure 1 are in this situation. The same rule is applied to break this deadlock. If a waiting timer (*WAIT*) expires and there exists more than one node in *Init* state within transmission range, a node (23 in the example) having the largest H_1 result among the nodes in deadlock becomes a SLDR

(lines 16 and 17 in Algorithm 1).

Correctness analysis. The ESMIS algorithm takes time proportional to the number of neighbors. Note that this does not depend on the total number of nodes in the network. After executing the ESMIS algorithm, each SLDR knows which neighbors it covers. It then sends a report of its subset to the base station. The following lemma shows that the ESMIS algorithm generates a MIS.

LEMMA 1: *A set of SLDRs generated by Algorithm 1 (ESMIS) is a MIS.*
Proof: (*Sketch*) To prove this, we need to show that two statements hold: 1) no two SLDRs have an edge between them and 2) there is no node to be added to the MIS after finishing the ESMIS, i.e., all nodes not in the MIS have at least one neighbor in the MIS.

1) Let us assume that there is an edge between two SLDRs in the MIS generated by Algorithm 1, i.e., two SLDRs are selected in transmission range. In Algorithm 1, there are three cases that a node becomes a SLDR: a node has the largest H_1 result within transmission range (line 5); it receives COV messages from all neighbors (line 10); a node has the largest H_1 among two or more *Init* state nodes in *deadlock* (line 17). First, the selection of a SLDR is based on the H_1 result and unique node identifier. Every node computes H_1 of itself and neighbors. For two nodes in transmission range to become a SLDR, the two nodes must have the same largest H_1 result and the same identifier. However, in the network, since every node has a unique identifier, this cannot happen. This contradicts the assumption. The second case does not occur since if there exists at least one SLDR, the other node receives at least one ANN message (i.e., it will not receive COV messages from *all* neighbors). The third case cannot occur for the same reasoning as the first case. Therefore, there cannot be an edge between SLDRs.

2) Under the assumption of network connectivity, every node in the network starts Algorithm 1 when it receives the *seed*. Once a node performs the ESMIS, its state is either *Ruled* or *Ruler*. Nodes in *Ruler* state are already in the independent set. Let us assume that there is a *Ruled* node which can be added to the independent set. The *Ruled* node is already connected to a SLDR which covers it. The addition of the *Ruled* node to the MIS, i.e., the node becomes a SLDR, creates an edge between two SLDRs. This cannot occur as shown in 1 above. Hence, once every node finishes the ESMIS algorithm, there is no more node to be added to the MIS. ∎

In the ESMIS algorithm, an adversary is unable to predict

which nodes will be SLDRs because of the use of the random *seed*. To further strengthen security, each time that the base station initiates the detection of node replication, it broadcasts a different *seed*. Hence, although compromised nodes may, in some specific scenarios, collaborate to successfully hide themselves, they can be detected with a significant probability in other rounds in which the base station generates a different seed. We will examine this probability in the next section.

B. Authenticated Subset Covering

The ESMIS algorithm presented above works well in a network in which no node is compromised. However, sensors may be deployed in a hostile environment in which an adversary may try to hide its existence by not participating in the ESMIS algorithm or by generating a bogus COV message with a fabricated SLDR identifier.

To curb this attack, we present a membership authentication approach. The basis of this approach is to have a *Ruled* node inform all its neighbor SLDRs of its covering SLDR. This way, the neighbor SLDRs can authenticate the veracity of the statement as follows. Suppose that a node i receives the first ANN message from SLDR g and a second from SLDR h. Node i is covered by g. As presented in the previous subsection, node i sends a COV message (ID=i, SLDR=g) to its neighbors. SLDR h determines that i is not a member of its subset. Node i then transmits the identifier of SLDR h to SLDR g so that SLDR g generates a membership authentication of i to SLDR h. In the remainder of this paper, $K_{ij}=K_{ji}$ denotes a pairwise key between nodes i and j. The message that g sends to h is as follows:

$$g \rightarrow h : [i, g, MAC(K_{gh}, i|g)]$$

The SLDR h can confirm that SLDR g covers the neighbor i by checking this message and saving $MAC(K_{gh}, i|g)$, referred to as *membership MAC*, which is used to vouch for SLDR g's covering of i.

Neither can a node generate a bogus COV message carrying a fraudulent SLDR identifier. For example, consider the case in which node i sends a COV message including SLDR f which does not exist. Node i should be able to generate the membership authentication computed by f to h with a pairwise key which has been established between h and f. However, node i can neither generate this key nor the membership authentication.

During this authentication process, a SLDR collects a set of neighbor SLDRs and shortest path information to them. The maximum distance of shortest paths between two SLDRs is three hops, i.e., at most two covered nodes may be on the path between two neighboring SLDRs.

After this authentication, each SLDR generates a report that includes a set of members, non-member nodes, and a MAC for the report. A report generated by SLDR g is as follows:

$$M_k = [k, S_{gk}]$$
$$g \rightarrow BS : [S_g, M_1, \ldots, M_m, MAC(K_{g,BS}, S_g|M_1|\ldots|M_m)]$$

, where non-member neighbors ($\cup_{k=1}^{m} S_{gk}$) are covered by m neighbor SLDRs, S_{gk} is a set of non-member neighbors covered by SLDR k, and S_g is a set of nodes covered by SLDR g. In summary, the subset report from a SLDR includes non-members and SLDRs covering the non-members, as well as a set of

members and a MAC for the report. Otherwise, corrupted SLDR and member nodes may collaborate and not report a member.

C. Verifiable Random Member Selection

In the basic scheme, every node's identifier is reported to the base station. This may cause the report packet size to become large near the top of the tree. To address this, we further optimize *SET* by using randomization for the subset formation, in which a SLDR generates a report of randomly selected members, instead of all members.

First, according to Algorithm 1, SLDRs are elected and exclusive subsets are constructed in the network. Each SLDR then selects a subset of members at random, based on a threshold σ which is a system parameter in $(0, \ldots, 1]$. For instance, if σ is 0.75, each SLDR generates a subset report of itself and randomly selected 75% of its members. However, this solution raises a question of how to verify whether a node is not selected or is not included intentionally to hide its existence by an adversary, if it is a clone.

To overcome the limitation of the simple randomization approach, we present a verifiable random selection scheme. The key idea of the verifiable random selection is to enable the base station to determine which sensors should be reported. To do this, the base station releases additional information (a bit array of $m+1$ bits, where m is the maximum value of $y=H_1(seed|x)$ computation [1]), referred to as a *mbins*, when broadcasting the *seed*. A SLDR generates a report of members whose y-th bit of the *mbins* is set to 1. For example, if m is seven and the *mbins* is set to "10101010" (left to right), a SLDR selects members whose y is 0, 2, 4, or 6. The base station can control the number of reported nodes by setting more (or less) bits of the *mbins* to "1". Since a SLDR already has a list of neighbors and their H_1 values, the decision does not incur any additional computation or communication. Moreover, due to the randomness of the *seed*, the adversary cannot predict which nodes will be selected.

Since the base station knows the list of all sensors, it can check if the reports include all the sensors to be reported. Except for the selection of members in each exclusive subset, all the operations are performed in the same way. Hence by using the verifiable random selection, we can reduce the message size of reports, while achieving the verifiable subset report of sensors. The limitation of this approach is that it may take longer (multiple rounds) to detect clones. However, as the analysis in the subsection IV-A.3 will show, the probability that a base station may run more than one round to detect clones is very low. We will show the detailed security analysis later in Section IV.

D. Distributed Set Computation on Subset Trees

The ESMIS algorithm outputs a set of unit exclusive subsets in the network. To detect node replication, each SLDR can send its report to the base station. Due to the random deployment of sensor nodes, the base station may not know which nodes are selected as SLDRs. This may enable the adversary to drop reports from subsets on the way to the base station.

To support more efficient and attack-resilient detection of set operations, we employ a random tree structure. A tree enables the

---

[1]The maximum value of $y=H_1(seed|x)$ computation indicates the degree of a node as described in III-C. Therefore, the size of m will be only a few bits

At Root
  Send CTREE ($SID_1 = r$) to all neighboring SLDR nodes
Intermediate node i
  :: Receive CTREE($\{SID_k\}_{k=1,...,i-1}$) from SLDR SID_{i-1}
  if (Root = 0 & Parent = 0) **then**
      Root = SID_1
      Parent = SID_{i-1}
      Ancestors = $\{SID_k\}_{k=1,...,i-1}$
      Add myself (SID_i) to $\{SID_k\}_{k=1,...,i-1}$
      for all neighbor SLDR N_k, except SID_{i-1} **do**
          Send CTREE($\{SID_l\}_{l=1,...,i}$) to N_k
      endfor
      Send CTREE_RESP to SID_{i-1}; done;
  endif
  Send CTREE_RESP to SID_{i-1}; done;

hierarchical construction of subsets, guaranteeing exclusiveness because of its no-cycle condition. A subset generated by the ESMIS algorithm is a node of the subset tree.

The tree construction is initiated by a root SLDR. The root sends a final report collected from the tree to the base station. This leaves open a question of which SLDR will be the root. A compromised node may try to be a root to control the set computation. To overcome the vulnerability in a single-rooted and fixed tree, we present a multiple tree based approach in which multiple roots are decided randomly in the network. Each root initiates tree construction so that multiple trees of subsets are constructed independently and in parallel. As a result of multiple tree construction, the network can be divided into non-overlapping regions. The intersection and union of subsets are performed on each tree in a distributed way. If the intersection of all subsets computed in the base station is empty, there are no clones detected in the network.

The *seed* that the base station broadcasts is also used for choosing roots randomly in the network. Roots are determined by computing another hash function, $H_2 : seed \rightarrow M$, i.e., $M = H_2(seed) \in [1, \ldots, N]$. If a subset has at least one node whose identifier is in $[M, M + B)$, the SLDR of the subset becomes a root. For example, if nodes in $[M, M + B)$ are in different subsets in the network, B trees will be constructed. The value B is a system parameter which decides the maximum number of trees constructed in the network. This way, the adversary is unable to determine roots and the tree structure.

The tree construction starts from each root by discovering the SLDRs neighboring the root; the SLDRs that are neighbors of the root set the root as their parent and become the root's children; for each child, the same tree extension is performed recursively. Discovering neighbors is carried out by sending a message, *CTREE*, carrying the root identifier to neighbors. Table I shows the tree construction pseudo code on a tree. During the tree construction, SLDRs obtain path information on the tree, i.e., the ancestor SLDRs on the tree. We present how this path information is used to authenticate set operations performed on the tree subsequently.

Unless a SLDR j is a root, it becomes a child of another SLDR i from which it receives the first CTREE message. The SLDR j sets i its parent and sends to i a CTREE_RESP message (ID=j, parent=i, root=SID_1). When SLDR i receives the CTREE_RESP message in which a parent field is set i, it

adds j to its children list. A SLDR may receive multiple CTREE messages which have a different root SLDR or parent. A SLDR becomes a leaf on the tree in the following situations: either it does not have any neighbor SLDR except its parent or all neighboring SLDRs except the parent already joined a tree.

After the tree construction, the leaf SLDR sends its subset report to its parent. The parent collects its children's subsets, and computes the intersection of these subsets to check if a clone exists. If not detected, it generates a union of its children's subsets and sends this new report to its parent. Each root forwards its final report (union of all subsets in a tree) to the base station.

If the intersection in the midst of the tree is not empty, this implies that clones exist in the subtree. The notification of node replication will be transmitted to the base station which will take further action.

E. Interleaved Authentication on a Subset Tree

An important issue we must address in the tree-based set operation is the dependability of the set operation results (intersection and union of subsets) on the tree. A parent collects its children's subsets, and computes the intersection and union of the collected subsets, with its own subset. If the parent is a corrupted node, it may delete cloned identifiers from the union.

Therefore, the set operations should be verified. To address this, we leverage the previous work [16] to propose an interleaved authentication scheme on the tree. During the tree construction, a SLDR obtains the path information from the root to itself. When a SLDR sends a report to its parent, it computes a keyed MAC, such as an HMAC [2], not only for its parent, but also for its grandparent (interleaved MAC).

If a corrupted parent changes the results of set operations, the grandparent can detect the inconsistency by computing and checking the interleaved MACs. We quantify the security afforded by this approach in the next section.

The report is different based on whether a SLDR is a leaf or an intermediary on the tree. Let us denote i, p, and g as child, parent, and grandparent SLDRs, respectively, and S_i as a subset of child i.

Leaf: Leaf SLDR i computes MACs for its parent and its grandparent. A subset report that leaf SLDR i generates is composed of its subset, MAC of the subset report, and interleaved MAC as follows:

$$i \rightarrow p : [S_i, MAC(K_{ig}, S_i), MAC(K_{ip}, S_i | MAC(K_{ig}, S_i)]$$

A parent SLDR verifies the reports from its leaves by checking the $MAC(K_{ip}, S_i | MAC(K_{ig}, S_i))$s of all children i.

Intermediate: An intermediate SLDR on a tree receives subsets from its children and computes the intersection and union of the received subsets and its own subset. It then generates a report to its parent. The report generated by an intermediate node also carries interleaved MACs from the children and an interleaved MAC computed by itself for its grandparent. Suppose that SLDR i has k children. A report that SLDR i generates is as follows:

$$
\begin{aligned}
M_1 &= (S_i, S_1 || \ldots || S_k, \{MAC(K_{jg}, S_j)\}_{j=1,k}) \\
M_2 &= (S_1 || \ldots || S_k || S_{k+1}), \text{ where } S_{k+1} = S_i \\
M_3 &= (M_1 | MAC(K_{ig_i}, M_2)) \\
i \rightarrow p_i &: [M_1, MAC(K_{ig_i}, M_2), MAC(K_{ip_i}, M_3)]
\end{aligned}
$$

where p_i and g_i denote the parent and grandparent of SLDR i respectively, and $||$ indicates the concatenation of subsets. The grandparent g_i uses M_2 and $MAC(K_{ig_i}, M_2)$ to check the operation of node i and p_i.

If the child is not a leaf, the intermediate SLDR i checks the concatenated subsets and MACs from its grandchildren ($MAC(K_{jg}, S_j)$ for all grandchildren j), as well as MACs from its children ($MAC(K_{kp}, M_k)$ for all children k). This interleaved authentication verifies the set operations of intermediate SLDRs and detects if they have removed identifiers. If all subset reports successfully pass the verification, the intermediate SLDR sends its parent a report which consists of a concatenation of children's subsets, its subset, interleaved MACs, and MAC for the report. Figure 2 shows an example of how the interleaved authentication works.
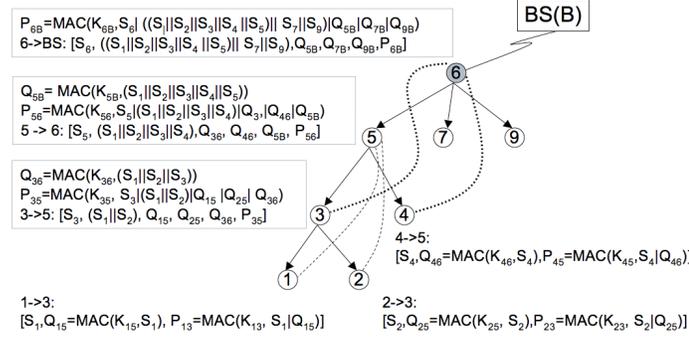

Fig. 2. An example of interleaved authentication on a tree with height 3: SLDR 3 sends to SLDR 5 a report which includes its own subset S_3, concatenation of S_1 and S_2 ($S_1||S_2$) and their interleaved MACs (Q_{15} and Q_{25}), SLDR 3's interleaved MACs (Q_{36}) and P_{35}. SLDR 5 computes the interleaved MACs for the received subsets (S_1 and S_2) and check with the received MACS (Q_{15} and Q_{25} from SLDR 1 and 2).

F. Detection at the Base Station

Each root forwards its final report (union of all subsets in a tree) to the base station. The base station verifies the reports from roots by checking MACs generated by roots and interleaved MACs by their children. The base station detects the clone attack by computing the intersection of any two received subsets from roots. If it detects cloned nodes, it may revoke the corresponding nodes by broadcasting the list of cloned identifiers to the network.

A compromised SLDR may lie by inserting a nonexistent neighbor node into the report. The base station will then receive multiple identifiers for this non-compromised node. To deal with this case, the base station may run multiple rounds of *SET* since the probability that the compromised node consecutively becomes a SLDR is very low. This basic scheme may be augmented with a reputation scheme to determine which nodes are inserting false reports. The details of this issue are outside the scope of this paper.

Based on the received reports, the base station can detect clones unless compromised nodes collaborate to hinder the detection. There are limited scenarios under which the collaboration of compromised nodes can successfully hinder detection. We quantify the likelihood of these circumstances in the next section.

IV. SECURITY ANALYSIS

In this section, we qualify the impact of compromised nodes on our detection scheme. We find that adversaries can only be effective in limited scenarios. We quantify the effectiveness of

an adversary by analyzing the probability that the conditions for these scenarios hold true. As we demonstrate, there are two colluding scenarios where the adversary can hide cloned members with a very low probability. However, in *SET*, the concealed clones in a specific round can be detected in the subsequent rounds since subset and tree structures may change based on the new *seed* broadcast by the base station. Unless compromised, we assume that sensor nodes operate properly according to the defined protocols.

The following notation is used in the remainder of the analysis.

- n: Number of clones that have the same identifier.
- N: Number of nodes in the network.
- T: Number of trees constructed in the network.
- B: Parameter value used to choose multiple roots in the network.
- L: Average number of neighbor subsets.
- P_s: Probability that a node is selected to be reported.
- P_a: Probability that an adversary avoids detection.
- $P(j=\text{Comp.})=P_c$: Probability that node j is compromised.
- $P(\text{Child}(i) = j)$: Probability that SLDR j becomes a child of SLDR i.
- P_{cchild}: Probability that SLDR j is compromised and becomes a child of a SLDR.
- P_{pc}: Probability that parent and child SLDRs are compromised and the parent is not a root.
- $P_{r,pc}$: Probability that parent and child SLDR nodes are compromised and the parent node is a root of the tree.

A. Node compromise on the subset construction

1) Single Node Compromise: In performing the subset construction, a compromised node may strive to be a head of the subset by falsely claiming that it has the largest H_1 result. In the ESMIS algorithm, however, every sensor locally computes H_1 for itself and all neighbor nodes. Therefore, a compromised node cannot falsely claim that it has the largest result, since other nodes also have the computation results.

The compromised node may also attempt to avoid being reported as a member of a subset by either claiming that it is covered by a non-existent SLDR or by keeping silent. A covering SLDR provides the membership authentication to verify that it covers a node. Since the compromised node does not have a pairwise key for its non-existent covering SLDR, it cannot provide this membership authentication. This prevents a node from claiming coverage by a non-existent SLDR.

In the ESMIS algorithm, the protocol precludes a node from keeping silent while it has a neighbor relationship with other nodes. If a node attempts to hide itself by not responding an ANN message, the non-compromised SLDR and other neighbors will detect this.

2) Nodes in collusion: In our setting, a node may play one of two roles: member or SLDR. Thus, various colluding scenarios may take place: member nodes may collude (members in the same subset or members in the different subsets), or a member node and SLDR may collude.

Two types of attacks may be attempted by colluding member nodes. First, a colluding node may try to help its partner claim to be the SLDR. As discussed above, given that other non-compromised nodes have the H_1 results of all neighbors, this attack is not effective. Second, colluding nodes may try to be silent, thus going undetected. Since colluding members have

neighbor relationship with their SLDRs, they cannot be silent, nor can they claim to be covered by a non-existent SDLR as discussed above.

Now consider the case of a compromised member i and SLDR j in collusion. If they are within transmission range of each other, one possible attack is for node i to claim itself as covered by j, and for j to eliminate i from its subset. In *SET*, SLDRs send their neighbor list (members and *non-members*) to the base station when a subset report is transmitted through the tree as described in Section III-D. Thus, the base station will check if the reported non-members are completely covered by a SLDR when it receives all reports from root SLDRs. Since the corrupted SLDR j does not include i in its subset, the base station will detect this missing node. It will query the legitimate neighbor SLDRs which reported the non-members (including i) for the membership MACs and the SLDRs of the missing non-members.

However, suppose that the compromised member j has only one SLDR within range which is also compromised. Since there is no other SLDR to report j as a non-member, the collusion of these two nodes may work to hide the compromised member j. To investigate the effectiveness of this colluding attack, we performed simulation of a network of 800 nodes in which two random nodes, x and y, are cloned. We generated 2, 6, 12, and 20 clones of each compromised node and deployed them as pairs (x,y) within transmission range of each other in different parts of the network. For the case in which an adversary generates two clones of each compromised node, the attack was successful 0.3% of the time. In the other cases (6, 12, and 20 compromised pairs), the adversary was never observed to be successful. Note, for this attack to be successful, *all* cloned pairs must be in a covered-SLDR relationship with no neighbor SLDRs; as the number of cloned pairs grows, this becomes a highly improbable scenario. This demonstrates that our subset construction and membership authentication schemes successfully mitigate the effectiveness of this colluding attack.

3) Effectiveness of Verifiable Random Selection: Now, we explore the effectiveness of an adversary in the verifiable random selection scheme. As discussed above, a single compromised node is not able to hide itself in *SET*. The random selection affects only the unit subset formation. Thus as a special case of node collusion, we separately analyze the effectiveness of an adversary to avoid detection during the subset construction in the random selection scheme.

First, we investigate the probability that an adversary successfully avoids detection. We need to consider two scenarios: either the identifier of clones is selected for reporting or not. Assume that n clones exist in the network. For an adversary to be successful in the first case, only a single clone node is reported; the other $(n-1)$ clones must be hidden. To do this, the adversary needs $(n-1)$ compromised SLDRs to collude with and hide the $(n-1)$ clones. For the later case, although the clone identifier is not selected, an adversary can avoid detection only when at most one clone is elected to a SLDR, because if multiple clones become a SLDR, each must generate a report and the base station will receive duplicate identifiers and detect the clone attacks. Let us denote $P_{1,SLDR}$ as the probability that
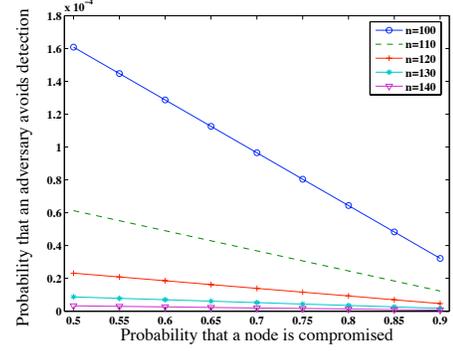

Fig. 3. The probability that an adversary successfully avoids detection

at most one node becomes a SLDR among n nodes located outside transmission range of each other. Since every node within transmission range has the same probability ($\frac{1}{d}$, where d is the average degree) to be a SLDR, the probability, $P_{1,SLDR}$, is defined by $P_{1,SLDR} = \binom{n}{1}(\frac{1}{d})(1-\frac{1}{d})^{n-1} + (1-\frac{1}{d})^n$.

Therefore, the probability P_a that an adversary successfully avoids detection using verifiable random selection is

$$P_a = P_s(P_c(1 - \frac{1}{d}))^{n-1} + (1 - P_s)P_{1,SLDR} \qquad (1)$$

, where $(P_c(1 - \frac{1}{d}))^{n-1}$ indicates that each of the $(n-1)$ clones that has not been selected as a SLDR is covered by a SLDR that is compromised. Equation 1 gives an upper bound of the probability that an adversary successfully avoids detection in the verifiable random selection scheme. This analysis assumes that if a SLDR and member are both compromised, the collusion will always be successful. However, this is not true as discussed in the previous analysis. Figure 3 shows that the adversary's effectiveness plummets as the number of clones (n) increases.

Now, we analyze the expected number of rounds that the base station must run *SET* for detecting clones. With verifiable random selection, the base station may run multiple rounds to detect clone attacks because in a round, a portion of a total set is reported. Based on the probability distribution (1), the expected number of rounds is defined by $E(X{=}k){=} \sum\limits_{k=1}^{\infty} kP_a^{(k-1)}(1 - P_a)$, which can be simplified to $E(X{=}k){=}\frac{1}{1-P_a}$. Note that the expected number of rounds converges to one quickly as n increases. Thus we can argue that the verifiable random selection makes *SET* more communication efficient, without sacrificing security. Furthermore, notice that the adversary must compromise at least $(n-1)$ other nodes to conceal clones.

B. Node compromise on the tree

In order to attack set operations on a tree, the compromised node must be a SLDR. In this subsection, we examine the impact of SLDR compromise.

1) Single Node Compromise: Once a compromised node becomes a SLDR, it also joins the tree construction and participates in set operations on the tree. The compromised SLDR may try to change the results of set operations on a tree by removing cloned node identifiers. The proposed interleaved authentication on a tree enables the detection of the changes by the compromised SLDR: if the corrupted SLDR removes identifiers from the subsets of its

children, its parent SLDR will detect inconsistency by computing interleaved MACs for the received subsets and checking with the received interleaved MACs which are computed for the original subsets by the grandchildren.

2) Nodes in Collusion: The main goal of colluding SLDRs is to hamper the set operations so that cloned nodes may not be detected. There are four colluding scenarios: 1) compromised SLDRs belong to different trees; 2) compromised SLDRs are on different paths of the same tree; 3) compromised SLDRs are on the same tree, but not in an immediate parent-child relationship; and 4) compromised SLDRs are in a parent-child relationship.

For the former two cases, if the compromised SLDRs are on the different paths or different trees, they cannot effectively collaborate. Their misbehavior on its tree or path is detected in the same way as the case of a single compromised SLDR.

As an example of the third case, suppose that a SLDR and its grandparent are compromised. The corrupted SLDR will send its subset to the parent SLDR which is not compromised. This SLDR collects subsets from all its children. The legitimate parent SLDR sends the result to its parent (children's grandparent SLDR which is compromised). This grandparent may remove elements from the collaborating grandchild. In Figure 2, let us assume that SLDRs 2 and 5 are compromised. SLDR 3 sends a report to SLDR 5 which includes subsets S_1, S_2, and S_3. Corrupted SLDR 5 may remove elements from S_2 so that these nodes are not detected. However, the SLDR 6 will detect the inconsistency by computing $M_{36} = MAC(K_{36}, (S_1||S_2||S_3)_{rx})$ and checking if M_{36} is the same as the received MAC (Q_{36}) computed by the SLDR 3, where $(S_1||S_2||S_3)_{rx}$ is the received subsets from 5.

Therefore, an adversary can only be effective if the compromised nodes have a parent-child relationship (case 4). Since the adversary may be able to hide clones in this scenario, during the tree construction, a compromised SLDR may strive to have a parent-child relationship with another compromised SLDR with which it is a neighbor. In *SET*, however, due to the randomness in the subset construction, this cannot be planned, and if it occurs, is unlikely to occur repeatedly in subsequent rounds.

The parent-child scenario can be divided into two cases: the compromised parent is a root or the compromised parent is not a root. Note that a corrupted root can have a significant impact on the detection of clones. For these two operative scenarios, we quantify the effectiveness of the adversary by evaluating the probability that they can be in this relationship.

We first determine several probabilities that are used in the remaining analysis. A SLDR becomes a root if it has at least one identifier in $[M, M+B)$ in its subset. Hence, the probability that a SLDR becomes a root is

$$P_r = 1 - (1 - \frac{B}{N})^d$$

, where d is the average number of neighboring nodes in a subset.

In order for SLDR j to be a child of SLDR i, it should receive the first CTREE message from SLDR i, and it should not be a root because although it receives the first CTREE message from i, it cannot be a child of i if it is a root. Hence, the probability that SLDR j becomes a child of SLDR i is

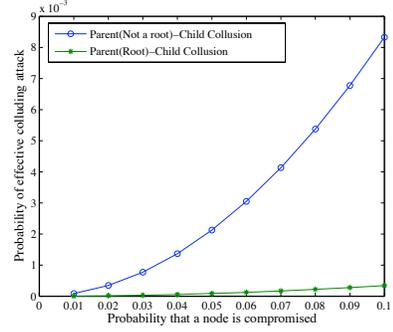$$P(\text{Child}(i) = j) = \frac{1}{L} * (1 - P_r). \quad (2)$$

Fig. 4. Effectiveness of an adversary ($L = 17$, $d = 10$, and $B = 4$)

The first term comes from the fact that a SLDR can receive the first CTREE message from L neighbor subsets.

Parent-Child (P_{pc}): We first analyze the probability that SLDR j is compromised and that it is a child of SLDR i which is also compromised and not a root. Since the node compromise and tree construction events are independent, the probability of being a child of SLDR i and compromised is

$$\begin{aligned} P_{cchild} &= P(j = \text{Comp.}, \text{Child}(i) = j) \\ &= P(j = \text{Comp.}) * P(\text{Child}(i) = j) \\ &= P_c * \frac{1}{L} * (1 - P_r) \end{aligned} \quad (3)$$

by applying equation 2.

For the adversary SLDR i to be effective, it must have at least one compromised child SLDR. Thus, the probability that the adversary is effective is the same as the probability that SLDR i is compromised and it has at least one compromised child. In the case we examine here, SLDR i is not a root. Therefore, the probability is:

$$\begin{aligned} P_{pc}(\text{effective}) &= (1 - P_r) * P(i = \text{Comp.}) * \sum_{k=1} P_{cchild} \\ &= (1 - P_r) * P_c * (1 - (1 - P_{cchild})^{L-1}) \end{aligned} \quad (4)$$

Root ($P_{r,pc}$): If the compromised parent is a root of the tree, then the attack may have a large impact on the final set result. Equation 4 is the probability that the compromised parent SLDR i is not a root. Now, we examine the case in which SLDR i is compromised and is a root.

The probability that a parent SLDR i is a compromised root, and it has at least one compromised child is

$$P_{r,pc}(\text{effective}) = P_r * P(i = \text{Comp.}) * (1 - (1 - P_{cchild})^L) \quad (5)$$

Figure 4 shows the probabilities of $P_{pc}(\text{effective})$ and $P_{r,pc}(\text{effective})$. When the compromised nodes occupy 10% of the network, the scenario of collusion of a non-root parent and child SLDR has a probability 0.008 of occurring. For the scenario of a compromised root and its child in collusion, the probability of occurrence is $(0.3 * 10^{-3})$. In these cases, the adversary can hide clones only if all clones belong to the subtree rooted at the colluding parent-child SLDRs. Furthermore, in *SET*, since the base station generates a different *seed* each time, the probability that this attack may occur consecutively is statistically insignificant (0.008^2 and $(0.3 * 10^{-3})^2$ for P_{pc} and $P_{r,pc}$ respectively). We can conclude that our detection scheme based on a set model can support a reliable and secure detection of the clone attack.
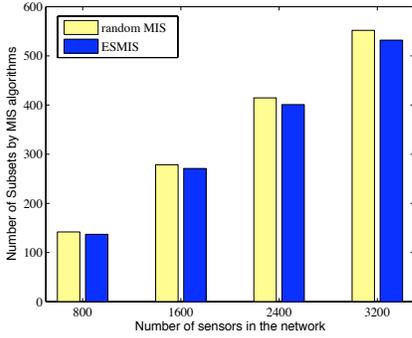
Fig. 5. Comparison of the number of subsets.

V. PERFORMANCE ANALYSIS

Based on requirements of a large sensor network, we performed simulations on networks of 800, 1600, 2400, and 3200 nodes which are deployed randomly. The area of the network is varied to achieve an average degree of 10. In this section, we study the performance of the ESMIS algorithm and analyze the overhead of *SET*, comparing with existing solutions. The analysis and simulation results show that *SET* outperforms the existing solutions.

A. Performance of ESMIS algorithm

As a reference point, we used a random MIS algorithm in which a node is randomly selected from V to join a MIS and its neighboring nodes are removed from V. This operation is performed continuously on V until V becomes empty.

We conducted simulations on the ESMIS algorithm and the random MIS algorithm 50 times to get the average size of the MIS. The results are shown in Figure 5. The results show that the ESMIS algorithm creates fewer subsets than the random MIS algorithm, while it can be executed in a distributed and parallel way. Since communication cost is proportional to the number of subsets, we claim that the ESMIS algorithm is appealing due to its distributed nature and its lower communication overhead, which we quantify below.

B. Communication Overhead Analysis

In this subsection, we examine the overhead incurred by *SET*. We perform a worst case analysis and simulation.

In *SET*, the subset construction, membership authentication, tree construction, and subset reporting incur packet transmissions. The communication cost can be measured by the expected number of messages transmitted during these operations. Broadcast (μTESLA) by the base station costs O(N) message transmission in the network. During the construction of subsets and membership authentication, selected SLDRs broadcast an ANN message and send membership MACs to the neighbor SLDR if some of its members are overlapped. The covered nodes send a COV message to their neighbors, relay the received neighbor COV messages to their SLDRs, and relay the membership MACs. Therefore, the subset construction and membership authentication costs $O(N)$ message transmissions in the entire network.

In constructing a subset tree, the CTREE message is propagated through to the leaf SLDRs. The CTREE_REP message is transmitted to the neighbor SLDR which transmitted the CTREE message. The transmission of the CTREE and CTREE_REP messages depends on the tree size. Suppose that T trees are constructed in the network and a tree has an average degree T_d. The total message transmission on a tree is $\sum_{k=1}^{h} T_d^k = \frac{T_d^{h+1}-1}{T_d-1}$, where h is the height of the tree. The height h of the tree is $\lfloor log_{T_d}(\frac{N}{ST}) \rfloor$, in which S is the average size of a subset and $\frac{N}{S}$ subsets may be constructed in the network. Therefore, the message transmission of the tree construction on each tree takes $O(\frac{N}{ST})$ which results in total $O(\frac{N}{S})$ message transmissions in the network. The constant factor 2 (at most 2 nodes between two SLDRs) is canceled out in the big O notation.

The transmission of reports in a tree is the same as the reverse of the tree construction, with an additional cost for each root to transmit a final subset report to the base station. The message transmission overhead for reports in the network is $O(T\sqrt{N} + \frac{N}{S})$. Therefore, the total message transmission overhead in *SET* takes $O(N) + O(N) + O(\frac{N}{S}) + O(T\sqrt{N} + \frac{N}{S}) = O(N)$ in the entire network.

Table II shows the comparison of the message transmission overhead with existing broadcast and multicast schemes, individual node report, and subset report schemes. The analysis results demonstrate that *SET* is the most efficient in terms of communication.

TABLE II
COMMUNICATION COST COMPARISONS

Schemes	Communication Cost
Broadcast	$O(N^2)$
Randomized Multicast [11]	$O(N^2)$
Line-Selected Multicast [11]	$O(N\sqrt{N})$
Individual Node Report	$O(N\sqrt{N})$
Individual Subset Report	$O(\frac{N}{S}\sqrt{N})$
SET	$O(N)$

Simulation: To verify the theoretical analysis, we implemented three schemes in a simulation environment: single node reporting, individual SLDR reporting, and *SET*. We conducted our simulations on four different size networks mentioned above. For each network, we generated 50 topologies to get the average message transmission overhead of each scheme in the entire network. The base station is located at a random place in the network.

The shortest path is used between all nodes, including SLDRs, and the base station. As the network size increases, *SET* shows scalable performance; it is linearly proportional to N as the analysis shown in Table II. *SET* is also advantageous in that it detects clones as the intersection and union of subsets are performed on a tree. Figure 6 shows our simulation results including the benchmarking cost $O(N\sqrt{N})$ which is the best performance of the existing solutions. We note that in particularly energy sensitive settings, the message size, not just the number of messages, is important. We leave this analysis for future work.

C. Memory Overhead

Available sensor products have limited memory. For example, Mica 2 motes, have 4 KB RAM [5]. In *SET*, each root maintains approximately $\frac{N}{T}$ identifiers, each of which is 12 bits. In the four networks used in our simulation, if we construct four trees in the networks, the subset at a root occupies only 0.3 KB, 0.6 KB, 0.9 KB, and 1.2 KB, respectively. This can be further reduced by using the verifiable random selection.

Moreover, roots do not need to keep this subset continuously. Each time that the base station initiates detection, roots and
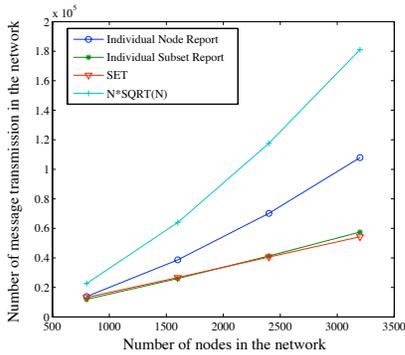
Fig. 6. Message transmission overhead: Comparing *SET* with individual node and individual subset report schemes.

SLDRs are reselected based on a new *seed*. Hence, during the network lifetime, sensors will share the overhead of being a SLDR and a root.

VI. RELATED WORK

In this section, we review the existing research efforts to combat clone attacks. Capkun et al. [4] proposed a secure positioning and distance measuring scheme. In particular, they considered an attack in which cloned nodes appear to be a single node that is changing location. To address this attack, the base station uses a unique fingerprint to identify each device. However, this may not be applicable to communication among peer sensors since each sensor will be required to keep the unique fingerprint information of other sensors. If a node is captured and cloned, it will then have the fingerprints it requires to go undetected.

Zhu et al. [15] proposed a key management protocol for sensor networks that provides a defense against the clone attack. The idea is to remove the master key once a sensor established pairwise keys so that although the adversary may generate clones of a node and deploy them, the clones cannot establish pairwise keys with the new neighbors.

A Sybil attack [10] is orthogonal to the clone attack. An adversary compromises nodes and deduces important information of different identifiers so that it can make clones to appear as different nodes. Newsome et al. [10] explored the Sybil attack in sensor networks and proposed several defenses.

In addition to efforts on preventive technologies, it is imperative to provide an efficient clone detection system. Few works, however, have addressed clone detection. Parno et al. [11] proposed random multicast and line-selected multicast schemes in which neighbor nodes of a sensor select random multiple witness nodes in the network and send a report (identifier and location) of the node to them. The distributed nature of this solution is interesting. This scheme incurs higher communication overhead ($O(N\sqrt{N})$) and memory usage to accommodate public keys of enough other nodes to achieve a high probability of detection.

VII. CONCLUSIONS

Considering that sensors may not be equipped with tamper-resistant hardware, it is crucial to provide a detection system against clone attacks. In this paper, we presented *SET*, a detection scheme based on a set model of the sensor network. *SET* is composed of four components: formation of exclusive subsets, authentication of subset covering, distributed set computation on subset trees, and preservation of reliable set operations on the tree. The randomization schemes used in *SET* enable resilient and efficient detection, while providing distributed load sharing among nodes in the network.

We provided a detailed security analysis for several types of attacks. Our probabilistic analysis showed that *SET* provides a resilient and dependable detection under colluding attacks. We also evaluated the performance and overhead of the proposed algorithm. The results showed that our solution has low transmission overhead, while using reasonably small memory space.

REFERENCES

[1] A. Becher, Z. Benenson, and M. Dornseif. Tampering with Motes: Real-World Physical Attacks on Wireless Sensor Networks. *International Conference on Security in Pervasive Computing*, 2006.

[2] M. Bellare, R. Canetti, and H. Krawczyk. Keying Hash Functions for Message Authentication. *Lecture Notes in Computer Science Vol. 1109*, 1996.

[3] C. Blundo, A. D. Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung. Perfectly-Secure Key Distribution for Dynamic Conferences. *Lecture Note in Computer Science*, 1995.

[4] S. Capkun and J.-P. Hubaux. Secure Positioning of wireless devices with application to sensor networks. *IEEE Infocom*, 2005.

[5] Crossbow. Wireless Sensor Networks (http://www.xbow.com/Products), July 2006.

[6] W. Du, J. Deng, S. Han, and P. Varshney. A pairwise key predistribution scheme for wireless sensor networks. *In Proceedings of ACM Conference on Computer and Communications Security*, 2003.

[7] Y.-C. Hu, A. Perrig, and D. Johnson. Packet Leashes: A Defense against Wormhole Attacks in Wireless Networks. *IEEE Infocom*, 2003.

[8] D. Liu and P. Neng. Establishing pairwise keys in distributed sensor networks. *In Proceedings of ACM Conference on Computer and Communications Security*, 2003.

[9] T. Moscibroda and R. Wattenhofer. Maximal Independent Sets in Radio Networks. *PODC'05*, 2005.

[10] J. Newsome, E. Shi, D. Song, and A. Perrig. The Sybil Attack in Sensor Networks: Analysis & Defenses. *3rd International Symposium on Information Processing in Sensor Networks*, 2004.

[11] B. Parno, A. Perrig, and V. D. Gligor. Distributed Detection of Node Replication Attacks in Sensor Networks. *IEEE Symposium on Security and Privacy*, 2005.

[12] A. Perrig, R. Szewczyk, J. Tygar, and etal. SPINS: Security Protocols for Sensor Networks. *Wireless Networks*, 8(5):521–534, 2002.

[13] B. Przydatek, D. X. Song, and A. Perrig. SIA: secure information aggregation in sensor networks. *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems, SenSys (SenSys)*, 2003.

[14] Y. Yang, X. Wang, S. Zhu, and G. Cao. SDAP: A Secure Hop-by-Hop Data Aggregation Protocol for Sensor Networks. *ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2003.

[15] S. Zhu, S. Setia, and S. Jajodia. LEAP: Efficient Security Mechanisms for Large-Scale Distributed Sensor Networks. *ACM conference on Computer and communications security*, 2003.

[16] S. Zhu, S. Setia, S. Jajodia, and P. Ning. An Interleaved Hop-by-Hop Authentication Scheme for Filtering of Injected False Data in Sensor Networks. *Proceedings of IEEE Symposium on Security and Privacy*, 2004.