Distributed Detection of Node Replication Attacks Resilient to Many Compromised Nodes in Wireless Sensor Networks

Yuichi Sei
The University of Tokyo 7-3-1 Hongo,
Bunkyo-ku, Tokyo 113-8654, Japan
sei@nii.ac.jp

Shinichi Honiden
National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430,
Japan
The University of Tokyo 7-3-1 Hongo,
Bunkyo-ku, Tokyo 113-8654, Japan
honiden@nii.ac.jp

ABSTRACT

In large-scale sensor networks, sensor nodes are at high risk of being captured and compromised. Once a sensor node is compromised, all the secret keys, data, and code stored on it are exposed to the attacker. The attacker can insert arbitrary malicious code in the compromised node. Moreover, he can easily replicate it in a large number of clones and deploy them on the network. This node replication attack can form the basis of a variety of attacks such as DoS attacks, and Sybil attacks. Previous studies of node replication attacks have had some drawbacks; they need a central trusted entity or they become vulnerable when many nodes are compromised. Therefore, we propose a distributed protocol for detecting node replication attacks that is resilient to many compromised nodes. Our method does not need any reliable entities and has a high detection rate of replicated nodes. Our analysis and simulations demonstrate our protocol is effective even when there are a large number of compromised nodes.

Keywords

Security, Algorithm, Wireless sensor networks, Node compromising attack, Node replication attack

1. INTRODUCTION

A wireless sensor network consists of a large number of low-cost and low-power sensors. The sensor nodes collect information about their environment. Such a network is suitable for military surveillance, forest fire monitoring, etc. To join the network, new nodes should pass certain authentication phases of neighbor discovery protocols [3, 7]. These protocols determine whether the new nodes are legitimate or not by checking their IDs and corresponding keys. A problem is that attackers may capture and compromise sensors. They can obtain the secret keys stored in the compromised nodes and insert arbitrary malicious code in the nodes. Moreover, they can replicate a large number of clones and deploy them to the network. Because these replicated nodes have legitimate IDs and corresponding legitimate keys, neighbor discovery protocols cannot detect them.

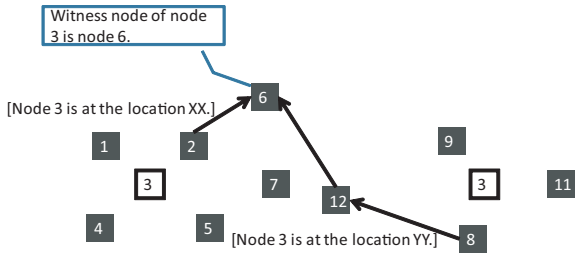


Figure 1: Replicated nodes and witness node (the numbers represent the node IDs)

Related studies [5, 15, 23] have proposed several protocols to detect node replication attacks. The basic idea is as follows. Every node broadcasts its node ID and location to one-hop neighbors. We call this message a claim and the node that broadcasts a claim a **claimer node**. The neighbors of the claimer node forward this claim toward a **witness node**. A witness node is determined by the node ID of the claimer node. If the witness node receives more than one claim containing the same node ID but from different locations, the witness node can conclude that more than one node with the same node ID exists (Fig. 1).

RED [5] can detect replicated nodes efficiently, but it needs a trusted entity (e.g., the base station). We cannot assume all wireless sensor networks have a trusted entity. For example, when a sensor network is deployed in a hostile environment such as a battlefield, the base station is an attractive target to the attackers. In this paper, we propose a protocol which does not need any trusted entities. B. Parno et.al. proposed two distributed detection protocols [15]. First, Randomized Multicast (RM) distributes node location information to randomly-selected witnesses. Second, Line-Selected Multicast (LSM) is an efficient distributed detection protocol. However, both have relatively low detection rates. SDC and P-MPC [23] are other distributed detection protocols. They do not require trusted entities and can detect replicated nodes with high probability. However, if an attacker compromises many nodes, the detection rates will greatly decrease. Our goals is to propose a replicated node detection protocol which does not need any trusted entities and is resilient to many compromised nodes attacks.

The rest of this paper is organized as follows. Section 2 presents a model sensor network and describes node replication attacks. Sec-

Digital Object Identifier: 10.4108/ICST.WICON2008.4796
http://dx.doi.org/10.4108/ICST.WICON2008.4796

tion 3 discusses the related methods and their problems. Section 4 presents the design of our algorithm of replicated node detection. In Section 5, we evaluate our method by using a mathematical analysis and simulations. Section 6 concludes the paper.

2. SYSTEM MODEL

In this section, we define our assumed sensor network model and the model of node replication attacks.

2.1 Sensor Network Model

We take into account a sensor network composed of a large number of small sensor nodes. Since they are designed to be inexpensive, the sensor nodes are assumed not to be equipped with tamper-resistant hardware. Once deployed, each node can determine its geographic location by using a localization scheme [2, 19].

The compromised nodes can launch many attacks. However, these threats are addressed in other research. We assume that sensor nodes can detect an attack involving dropping and fabricating messages by using detecting protocols such as [12, 22] and further assume that nodes cannot alter their IDs by detecting false IDs protocols [13].

These assumptions are the same as in other replicated node detection protocols [5, 15, 23].

We do not assume the existence of a trusted entity. In military applications, sensor nodes may be deployed in hostile regions. A base station would be an attractive target to the attackers. Many studies on wireless sensor networks devised various protocols that do not need trusted entities [11, 6]. The sensor nodes are loosely time synchronized as in μTESLA [16].

2.2 Node Replication Attack

An attacker may compromise multiple sensor nodes in a network. Once a sensor node is compromised, all the secret keys, data, and code stored on it are exposed to the attacker. The attacker can easily replicate it in a large number of clones and deploy them to the network. This node replication attack can be the basis for launching a variety of attacks such as DoS attacks [21, 14] and Sybil attacks [13]. If there are many replicated nodes in the network, they can multiply the damage to the network. Therefore, we should quickly detect replicated nodes.

3. RELATED WORK

We introduce five protocols: RED [5], RM and LSM [15], and SDC and P-MPC [23]. First, we explain their common method and goal. In all of these protocols, each node is preloaded with a unique node ID, a shared function which calculates the location of a witness node of an arbitrary node from the node ID, and a private key for creating a signature. They use ID-based public key crypto-schemes [18, 1] for creating and checking signatures. Each node can easily calculate a corresponding public key for a given ID. However, we cannot calculate the private key for an arbitrary ID because of its computationally complexity. When a detection process starts, each node n_i sends its node ID i, location, and signature to one-hop neighbor nodes (referred to as a **claim**). The neighbor nodes create a corresponding public key for the node ID i; then it checks the correctness of the signature. Next, each node calculates the location of the **witness node** for n_i. They forward the received claim toward the witness node by using a geographic routing protocol such as GPSR [8] with probability p. We call the node which forwards the claim a **reporter node**. If the witness node receives several claims that contain the same node ID but different locations, the witness node concludes that these sending nodes have been compromised and that an attacker replicated them.

If the location of the witness node is statically determined by the given node ID, an attacker who compromised one node can specify the witness nodes of all nodes. For example, if an attacker compromises a node and obtains a function that determines the location of the witness node from a given ID, he can compromise the witness node. Then the attacker can create and deploy a large number of replicated nodes without detection. Therefore, we need to avoid specifying the witness node from an attacker even if the attacker compromised many nodes.

In the RED protocol [5], a trusted entity broadcasts a one-time seed to the whole network. The location of the witness node of a node is determined from the node ID and the seed. Because the seed changes every time, an attacker cannot specify the location of a witness node in advance. The authors of RED said one can also use distributed protocol without a trusted entity such as a local leader election mechanism [4] to create a one-time seed. However, the authors did not mention how to create it; moreover, the local leader election mechanism creates a local leader from a small number of sensor nodes. Even worse, the method does not consider the existence of compromised nodes. Therefore, we cannot use it to create a global leader of a sensor network composed of a large number of nodes with some of them compromised.

In the RM and LSM protocols [15], the neighbor nodes of node n_i determine several witness nodes randomly. By exploiting the birthday paradox, at least one node is likely to receive conflicting location claims for a particular node. In the LSM protocol, each node which forwards claims also saves the claim. That is, the forwarding nodes are also witness nodes of a node which has the node ID in a claim. Therefore, LSM gives a higher detection rate than that of RM. However, both protocols have relatively lower detection rates compared with RED.

In the SDC and P-MPC protocols [23], the sensor network is considered to be a geographic grid. In the SDC protocol, witness nodes candidates of one node are all nodes of a grid. The grid is statically determined by the node ID, but which nodes in the grid actually become witness nodes is determined randomly. In P-MPC, to increase resiliency to many compromised nodes, the candidate witness nodes for one node are all nodes of several grids (three grids by default). The number of nodes in a grid was set to 100 by default. If an attacker compromises a node, learns the witness grids of the node, and compromises 100 nodes of one of the grids, the detection rate will decrease to about 1/8 (see the evaluation in section 5). Moreover, if the attacker compromises all nodes of the three grids (i.e., 300 nodes), these protocols can never detect node replication attacks. Many recent studies on security in wireless sensor networks have proposed methods that are resilient to many nodes being compromised (e.g., a thousand compromised nodes) [17, 10, 9, 20]. Therefore, we need to propose a replicated node detection protocol resilient to many compromised nodes.

4. DISTRIBUTED DETECTION PROTOCOL

This section presents our design for detecting replicated nodes.

4.1 Overview

Digital Object Identifier: 10.4108/ICST.WICON2008.4796
http://dx.doi.org/10.4108/ICST.WICON2008.4796

To prevent an attacker from learning the location of a witness node of a compromised node, our protocol uses a one-time seed for each replicated node detection process. In the RED protocol [5], a trusted entity gives it to all the nodes. In our protocol, each node sends a one-time seed to other nodes in turn. That is, each node has the role of starting a detection process and it is preloaded with the assigned turn number and seed for the turn.

When node n_i has a turn starting detection process, it sends the seed and its ID with a signature. Other nodes verify the signature and execute the detection process if the verification succeeds. Here, we should consider that sensor nodes break down easily and an attacker may compromise a large number of nodes. We divide nodes into groups to increase resiliency to fault nodes and compromised nodes. The role of the starting detection process is not assigned to each node but to each group.

We assign each node its node ID and its *role ID* and group nodes according to their role IDs. The reason why we introduce role IDs is to prevent attackers from learning the locations of the node that has a turn starting detection process. Attackers can learn a node ID and a location of the corresponding node because each claim contains a location of a node and its node ID. Therefore, nodes should use role IDs for starting detection process. $n_{i,j}$ denotes a node whose node ID is i and role ID is j.

In each group, nodes are sorted according to their role IDs. The number of nodes of a group is N_p. When a group G_i has a turn starting detection process (the turn is calculated from the elapsed time and the time interval of detection processes Δ), the first node of the group sends the seed and its role ID with its signature. If the process has not started after an internal interval time of detection process δ because of a failure, compromised node, or other reason, the second node of the group sends the seed and its role ID with its signature to the first node's place (see Fig. 2). Note that $N_p \cdot \delta$ can be less than but not greater than Δ.

If at least one node of a group survives, the group can start the detection process during its turn. An attacker must compromise the first node of a group which has the next turn starting detection process if he wants to learn the location of the witness node in the next detection process.

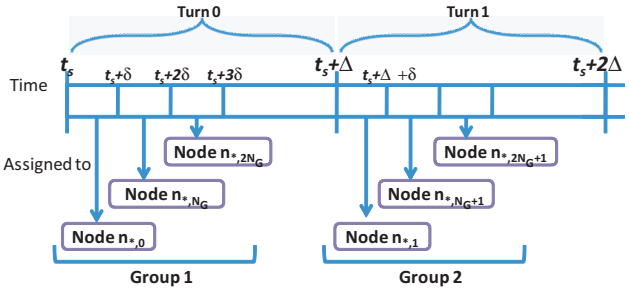
The following subsections describe the details of our proposal. We use the notation presented in Table 1.

4.2 Preparation of sensor nodes

We preload each node $n_{i,j}$ with a hash function H, a time interval of detection processes Δ, an internal time interval of detection process δ, a function that calculates a location of a witness node from a seed and a node ID F_L, a function that calculates a public key for node IDs F_E, a private key for node ID K_i^{-1}, a function that calculates a public key for role IDs $\hat{F}_E$, and a private key for role ID $\hat{K}_j^{-1}$.

4.3 Tasks of node $n_{i,j}$

- Calculate the current turn number at fixed intervals of δ. Let the reference time be t_s and the current time be $t_{current}$. The current turn number is $T_c = \lfloor (t_{current} - t_s)/\Delta \rfloor$. The current internal turn number is $t_c = \lfloor [(t_{current} - t_s)\%\Delta]/\delta \rfloor$.
- Calculate the role ID of the node that will start the detection process during this internal turn. The role ID is $R_{ID} = T_c \mod N_G + t_c \cdot N_G$.
- If $j = R_{ID}$ and if in this turn T_c, the detection process has not been started, node n_i starts process A.
- If node n_i receives message M_A, it starts process B.
- If node n_i receives message M'_A, it starts process C'.



(a) Role of the starting detection process is assigned to each group ($N_p = 3$. '*' represents an arbitrary node ID.)

At time t_s,
node n_0 starts the detection process.
At time t_s+δ, if node n_0 has not started the detection process,
node n_{N_G} starts the detection process.
At time t_s+2δ, if nodes n_0 and n_{N_G} have not started the detection process,
node n_{2N_G} starts the detection process.

At time t_s+Δ,
node n_1 starts the detection process.
At time t_s+Δ+δ, if node n_1 has not started the detection process,
node n_{N_G+1} starts the detection process.
At time t_s+Δ+2δ, if nodes n_1 and n_{N_G+1} have not started the detection process,
node n_{2N_G+1} starts the detection process.

(b) Tasks of each node

Figure 2: Overview of our protocol

4.3.1 Process A

- Node n_i broadcasts a message

$$M_A(i) = \langle i, l_i, S_{K_i^{-1}}(H(T_c||i||l_i)) \rangle$$

to one-hop neighbor nodes where l_i represents the location of node n_i. The symbol || represents a stream concatenation.

- Node n_i then broadcasts a message

$$M_{A_2} = \langle S_{\hat{K}^{-1}_{R_{ID}}}(H(T_c)) \rangle$$

to one-hop neighbor nodes. This value is the seed of the detection process in this turn.

4.3.2 Process B

Assume that node n_i has received a message $M_A(j)$.

- Node n_i verifies the location l_j.

Digital Object Identifier: 10.4108/ICST.WICON2008.4796
http://dx.doi.org/10.4108/ICST.WICON2008.4796

Table 1: Notation

N	Number of nodes in the network
n_i	Sensor node whose node ID is i
$n_{i,j}$	Sensor node whose node ID is i and role ID is j (We can identify a node by node ID or role ID only.)
K_a	Public key of a node whose node ID is a (calculated by a function $F_E(a)$)
K_a^{-1}	Private key of a node whose node ID is a
$\hat{K}_a$	Public key of a node whose role ID is a (calculated by a function $\hat{F}_E(a)$)
$\hat{K}_a^{-1}$	Private key of a node whose node ID is a
N_G	Number of groups
N_p	Number of nodes per group ($N = N_G \cdot N_p$)
C_n	Number of compromised nodes
$S_K(M)$	Signature of string M by key K
$F_L(s,i)$	Location of witness node obtained from seed s and node ID i
$H(M)$	Hash of M
Δ	Time interval of detection processes
δ	Internal time interval of detection process
r	Average number of reporter nodes of one claimer node

- If node n_i does not know the value of $S_{\hat{K}_{R_{ID}}^{-1}}(H(T_c))$, it starts process B'. If node already knows the value of $S_{\hat{K}_{R_{ID}}^{-1}}(H(T_c))$ or the verification in process B' succeeds, it starts process C.

- If node n_i succeeds with the verification in process B', it starts process A only once.

4.3.3 Process B'

- Node n_i receives a message M_{A_2} from node n_j.

- Node n_i calculates the public key of R_{ID}; $\hat{K}_{R_{ID}} = \hat{F}_E(R_{ID})$ and calculates $H(T_c)$. Then it verifies M_{A_2}.

4.3.4 Process C

Assume that node n_i has received a message $M_A(j)$.

- Node n_i calculates the location of the witness node of node n_j;

$$l_{dst} = F_L(S_{\hat{K}_{R_{ID}}^{-1}}(H(T_c)), j).$$

- Node n_i forwards a message

$$M'_A(j) = \langle j, l_j, l_{dst}, S_{K_j^{-1}}(H(T_c||j||l_j))\rangle$$

to an applicable node.

4.3.5 Process C'

Assume that node n_i has received a message $M'_A(j)$.

- If node n_i is the destination of l_{dst}, it starts process D. If not, it forwards the message to an applicable node for l_{dst}.

4.3.6 Process D

Assume that node n_i has received a message $M'_A(j)$.

- If node n_i does not know the value of $S_{\hat{K}_{R_{ID}}^{-1}}(H(T_c))$, it suspends this process until it receives a message M'_A.

- Node n_i calculates $F_L(S_{\hat{K}_{R_{ID}}^{-1}}(H(T_c)), j)$ and checks whether or not the calculated value is the same as the value of l_{dst}.

- Node n_i calculates a public key of the node whose node ID is j; $K_j = F_E(j)$, and calculates $H(T_c||j||l_j)$. It then verifies $S_{K_j^{-1}}(H(T_c||n_j||l_j))$.

- Node n_i saves the message $M'_A(j)$. If in this internal turn it receives another message $M'_A(j)$ which contains node ID j, it checks whether or not l_j is also the same. If it is not the same, it concludes that it detected a replication attack. It broadcasts these two messages to all the nodes in the network.

4.4 ANALYSIS

If an attacker compromises some nodes, he may learn the locations of witness nodes from them. To learn the locations of witness nodes, he needs to compromise the first node of a group if no nodes are broken. We call a turn in which an attacker compromises the first node of a group a **compromised turn**. If an attacker knows the locations of witness nodes of several consecutive turns, he can perform replicated node attacks from the first compromised turn of the sequence of compromised turns to the last compromised turn of the sequence. Therefore, we need to analyze the expected maximum number of consecutive compromised turns for the number of compromised nodes.

We also need to analyze another feature. If some of the nodes are faulty (i.e., compromised or broken), some turns may not be able to start a detection process. We call a turn that cannot start the detection process a **faulty turn**. We analyze the expected average number of consecutive faulty turns.

4.4.1 Maximum number of consecutive compromised turns.

Assume that C_n nodes are compromised. The probability that number of groups of which the first node compromised is C_G is

$$P_c(N, N_G, C_n, G_c) = \frac{\binom{N_G}{C_G}\binom{N - N_G}{C_n - C_G}}{\binom{N}{C_n}} \quad (1)$$

Each group has an ID. The ID of the first group is 1, and the ID of the last group is N_G.

We require the expected number of maximum consecutive compromised groups. For example, if an attacker compromises nodes in groups $3, 4, 5, 7$, and 10, the number of maximum consecutive compromised groups is three (groups $3, 4, 5$). Here, the next group of group N_G is group 1. The maximum number of consecutive compromised groups is the maximum number of turns in which

Digital Object Identifier: 10.4108/ICST.WICON2008.4796
http://dx.doi.org/10.4108/ICST.WICON2008.4796

the attacker can learn the locations of witness nodes from compromised nodes. We assume that the number of executing the detection processes is V.

First, we require the number of combinations such that the maximum number of consecutive compromised groups is equal to M_S when an attacker compromises C_G groups from N_G groups. Here, let us assume that the next group of group N_G is nonexistent. The number of combinations $f_{sc}(N_G, C_G, M_S)$ is

$$f_{sc}(N_G, C_G, M_S) = \begin{cases} \text{if } (C_G = 0 \,\&\, M_S = 0) \,||\, N_G = C_G = M_S \text{ then } 1, \\ \text{else if } N_G \leq 0 \,||\, N_G < C_G \,||\, C_G < M_S \text{ then } 0, \\ \text{else } \sum_{i=0}^{M_S} f_{sc}(N_G - i - 1, C_G - i, M_S) \\ + \sum_{i=0}^{M_S - 1} f_{sc}(N_G - M_S - 1, C_G - M_S, i). \end{cases} \tag{2}$$

We can get the expected number of maximum consecutive compromised groups when an attacker compromises C_g groups from N_G groups. Here, let us assume that the next group of group N_G is group 1. The expected number of maximum consecutive compromised groups $E_m(N_G, C_G)$ is

$$E_m(N_G, C_G) = \frac{1}{\binom{N_G - 1}{C_G - 1}} \times \begin{cases} \text{if } N_G = C_G \text{ then } V, \\ \text{else } (C_G)^2 \\ + \sum_{j=1}^{C_G - 1} \sum_{i=1}^{C_G} \left[j \cdot \max(i, j) \cdot f_{sc}(N_G - j - 2, C_G - j, i) \right]. \end{cases} \tag{3}$$

Finally, we can get the expected number of maximum consecutive compromised turns when an attacker compromises C_n nodes from equations 1 and 3.

$$E_M(N, N_G, C_n) = \sum_{i=1}^{C_n} P_c(N, N_G, C_n, i) \cdot E_m(N_G, i). \tag{4}$$

4.4.2 *Average number of consecutive faulty turns.*

Assume that F_n nodes are faulty (broken or compromised) of N nodes. The probability that the number of groups whose nodes are all faulty is F_g is

$$P_f(N, N_G, F_n, F_g) = \frac{\binom{N_G}{F_g} C_f(N_G - F_g, N/N_G, F_n - F_g \cdot (N/N_G))}{\binom{N}{F_n}},$$

where $C_f(N_G, N_p, F_n)$

$$= \begin{cases} \text{if } F_n = 0 \text{ then } 0, \\ \text{else } \binom{N_G \cdot N_p}{F_n} \\ - \sum_{i=1}^{\min(N_G, F_n/N_p)} {}_{N_G}C_i \cdot C_f(N_G - i, N_p, F_n - i \cdot N_p). \end{cases} \tag{5}$$

When the number of fault turns is F_g, the average number of consecutive faulty turns is

$$E_f(N_G, F_g) = \begin{cases} \text{if } N_G = F_g \text{ then } V, \\ \text{else } \sum_{j=0}^{F_g} j \cdot \left[\frac{N_G - F_g}{N_G - j} \prod_{i=0}^{j-1} \frac{F_g - i}{N_G - i} \right]. \end{cases} \tag{6}$$

Finally, we can get the expected number of average consecutive faulty turns when F_n nodes from equations 5 and 6.

$$E_F(N, N_G, F_n) = \sum_{i=1}^{\min(N_G, F_n \cdot N_G/N)} P_f(N, N_G, F_n, i) \cdot E_f(N_G, i). \tag{7}$$

5. EVALUATION

We conduct mathematical analysis and simulations of our protocols.

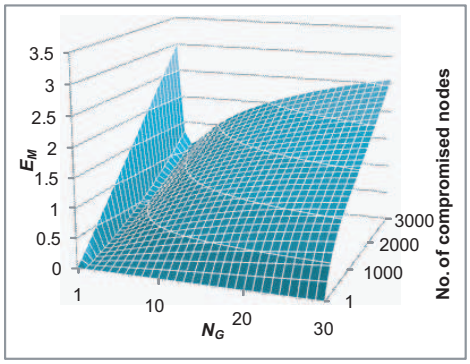
5.1 Mathematical analysis

We conduct mathematical analyses of the detection process (1. maximum number of consecutive compromised turns and 2. average number of consecutive faulty turns) and our reporter node determination protocol. We set the number of nodes of a network to 10,000.

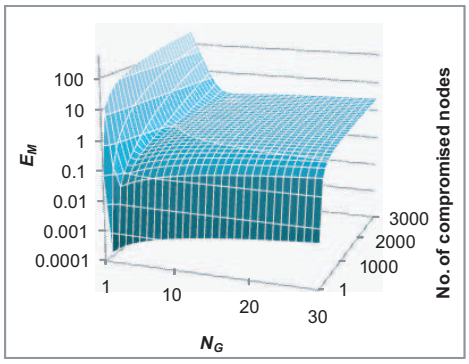
5.1.1 *Maximum number of consecutive compromised turns.*

Figure 3 represents the relationship among the number of compromised nodes (C_n), the number of nodes of a group (N_p), and the expected maximum number of consecutive compromised turns (E_M). In Fig. 3, the number of the detection processes V was set to 10, and in Fig. 3, V was set to 1,000. Assume that an attacker compromises more than N_G nodes. Although the probability that the number of compromised turns (E_M) is equal to N_G is very small, it is not zero. If E_M is equal to N_G, the attacker can learn all witness nodes in all turns. Therefore, N_G should not be very

Digital Object Identifier: 10.4108/ICST.WICON2008.4796
http://dx.doi.org/10.4108/ICST.WICON2008.4796



(a) $V = 10$



(b) $V = 1,000$

Figure 3: Expected maximum no. of consecutive compromised turns ($N = 10,000$)

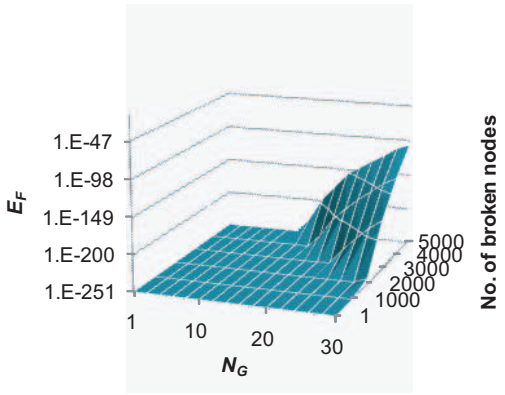


Figure 4: Expected average no. of consecutive faulty turns ($N = 10,000, V = 100$)

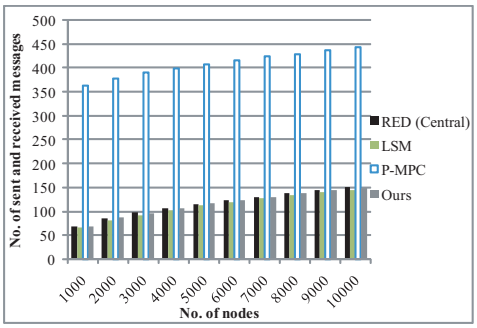


Figure 5: Communication overhead

small. If V is greate, a relatively greater N_G is better. However, N_G should not also be very greate value. If N_G is greate, the number of nodes per group is small. Therefore, the probability that an attacker compromises the first node of a group increases.

We know from the Fig. 3 that E_M is only 1.0 - 1.7 even if the attacker compromises 3,000 of 10,000 nodes.

5.1.2 Average number of consecutive faulty turns.

Figure 4 represents the relationship among the number of faulty nodes (F_n), the number of groups (N_G), and the expected average number of consecutive faulty turns (E_F). In Fig. 4, V was set to 100. We conducted analyses of varying V from 10 to 1,000, however, the results are almost the same as each other. To reduce E_F, we know from the figure that N_G should be small. This is because a faulty turn means a turn whose N/N_G nodes are all faulty. Even if 5,000 of 10,000 nodes are faulty, the number of consecutive faulty turns remains very small.

5.2 Simulations

We conducted simulations to demonstrate the effectiveness of our detection protocol of replicated node attacks. Table 2 lists the resources required by RED [23], LSM [15], P-MPC [5], and our protocol. RED uses a central trusted entity (e.g., a base station). The authors of [15] proposed the LSM and RM protocols. Because LSM is a modification of RM, we only compared LSM with our protocol. By the same token, the authors of [5] proposed the SDC and P-MPC protocols, but we only compared P-MPC.

The number of nodes was $10,000$. The orders in Table 2 are theoretical values. The numeric values represent the simulation results. The values of s and p_s in the P-MPC protocol are the protocol's system parameters. We used the default values for the related studies if they had system parameters. The results of our protocol are almost the same as that of RED. In our protocol, each node must do a signature check and save $S_{K^{-1}_{R_{ID}}}(H(T_c))$. Therefore, our protocol needs one more memory occupancy and signature check compared with RED.

Next, we conducted a simulation of communication overhead. Figure 5 shows the results. We varied the number of nodes from 1,000 to 10,000. We found that LSM needed a large overhead and the other protocols had almost equal overheads.

Next, we conducted a simulation of the detection rate with no compromised nodes. The results are shown in Fig. 6. We know that the detection rates of RED and ours are almost the same. That of P-MPC was a little smaller than ours, and that of LSM was relatively

Digital Object Identifier: 10.4108/ICST.WICON2008.4796
http://dx.doi.org/10.4108/ICST.WICON2008.4796

Table 2: Comparison with related work

	Memory Occupancy	Sent Message	Received Message	Signature Check	Detection rate
RED (Central)	$O(r)$ 1.00	$O(r \cdot \sqrt{N})$ 20.49	$O(r \cdot \sqrt{N})$ 49.35	$O(r)$ 6.02	0.914
LSM	$O(r \cdot \sqrt{N})$ 19.62	$O(r \cdot \sqrt{N})$ 20.55	$O(r \cdot \sqrt{N})$ 48.36	$O(r \cdot \sqrt{N})$ 12.52	0.338
P-MPC	$O(r \cdot s \cdot p_s)$ 7.60	$O(r \cdot \sqrt{N}) + O(s)$ 167.21	$O(r \cdot \sqrt{N}) + O(s)$ 195.62	$O(r \cdot s \cdot p_s)$ 13.62	0.866
Ours	$O(r)$ 2.00	$O(r \cdot \sqrt{N})$ 20.67	$O(r \cdot \sqrt{N})$ 48.59	$O(r)$ 7.00	0.912

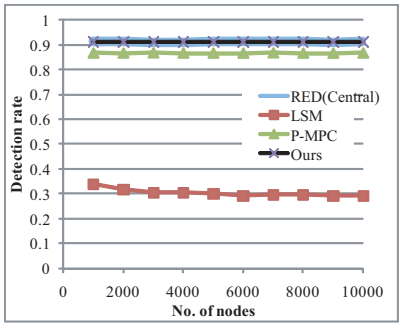


Figure 6: Detection rate with no compromised nodes

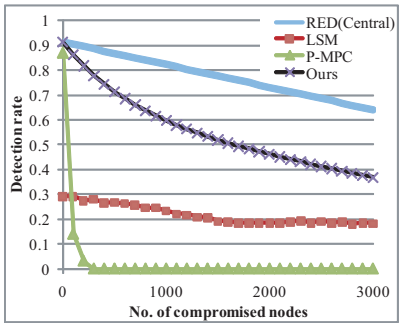


Figure 7: Detection rate depending on no. of compromised nodes

small compared with other protocols.

Finally, we conducted a simulation of the detection rate when the number of compromised nodes varied. If a witness node was a compromised node, the replicated node attack was not detected. In our protocol, the detection also failed if the attacker knew the location of the witness nodes. We set the number of nodes to 10,000. RED, LSM, and our protocols had high resiliency to many compromised nodes. However, P-MPC was easily affected by compromised nodes because all witness nodes candidates are leaked to an attacker if he compromises one node.

6. CONCLUSION

We proposed a detection protocol for node replicated attacks. We realized full distributed detection by assigning the process initialization role to each node. Each node is preloaded with a unique seed and uses it to initialize the detection process. We divide nodes into groups to increase resiliency to many compromised nodes, and we verified the resiliency of our protocol in mathematical analyses and in simulations. We also proposed a reporter node determination protocol. This protocol can be used for our detection protocol and other detection protocols dealing with node replication attacks. By using this protocol, the detection rate of node replication attacks was significantly higher than that of existing node determination protocols when there were several compromised nodes.

7. REFERENCES

[1] Dan Boneh, Craig Gentry, and Michael Hamburg. Space-efficient identity based encryptionwithout pairings. In *IEEE FOCS*, pages 647–657, 2007.

[2] Jehoshua Bruck, Jie Gao, and Anxiao (Andrew) Jiang. Localization and routing in sensor networks by local angle information. In *ACM MOBIHOC*, pages 181–192, 2005.

[3] Haowen Chan, Adrian Perrig, and Dawn Song. Random key predistribution schemes for sensor networks. In *IEEE Symposium on Security and Privacy*, pages 197–213, 2003.

[4] Gilbert Chen, Joel W. Branch, and Boleslaw K. Szymanski. Local leader election, signal strength aware flooding, and routeless routing. In *IEEE IPDPS*, page 244.1, 2005.

[5] M. Conti, R. Di Pietro, L.V. Mancini, and A. Mei. A randomized, efficient, and distributed protocol for the detection of node replication attacks in wireless sensor networks. In *ACM MobiHoc*, pages 80–89, 2007.

[6] Dezun Dong, Yunhao Liu, and Xiangke Liao. Self-monitoring for sensor networks. In *ACM MobiHoc*, pages 431–440, 2008.

[7] Wenliang Du, Jing Deng, Yunghsiang S. Han, Shigang Chen, and Pramod K. Varshney. A key management scheme for wireless sensor networks using deployment knowledge. In *IEEE INFOCOM*, 2004.

[8] Brad Karp and H. T. Kung. GPSR: Greedy perimeter stateless routing for wireless networks. In *ACM MOBICOM*, pages 243–254, 2000.

[9] Jooyoung Lee and Douglas R. Stinson. On the construction of practical key predistribution schemes for distributed sensor networks using combinatorial designs. *ACM Trans. Inf. Syst. Secur.*, 1(2):1094–9224, 2008.

[10] Donggang Liu, Peng Ning, and Wenliang Du. Group-based key predistribution for wireless sensor networks. *ACM Trans. Sen. Netw.*, 4(2):1550–4859, 2008.

[11] Liqian Luo, Chengdu Huang, Tarek F. Abdelzaher, and Jack Stankovic. Envirostore: A cooperative storage system for disconnected operation in sensor networks. In *IEEE INFOCOM*, pages 1802–1810, 2007.

Digital Object Identifier: 10.4108/ICST.WICON2008.4796
http://dx.doi.org/10.4108/ICST.WICON2008.4796

[12] Sergio Marti, T. J. Giuli, Kevin Lai, and Mary Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *ACM MOBICOM*, pages 255–265, 2000.

[13] James Newsome, Elaine Shi, Dawn Song, and Adrian Perrig. The sybil attack in sensor networks: analysis & defenses. In *ACM IPSN*, pages 259–268, 2004.

[14] Peng Ning, An Liu, and Wenliang Du. Mitigating dos attacks against broadcast authentication in wireless sensor networks. *ACM Trans. Sen. Netw.*, 4(1):1–35, 2008.

[15] Bryan Parno, Adrian Perrig, and Virgil Gligor. Distributed detection of node replication attacks in sensor networks. In *IEEE S&P*, pages 49–63, 2005.

[16] Adrian Perrig, Robert Szewczyk, Victor Wen, David E. Culler, and J. D. Tygar. SPINS: Security protocols for sensor networks. In *ACM MOBICOM*, pages 189–199, 2001.

[17] Kui Ren, Kai Zeng, and Wenjing Lou. Secure and fault-tolerant event boundary detection in wireless sensor networks. *IEEE Transactions on Wireless Communications*, 7:354–363, 2008.

[18] Adi Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO 84 on Advances in cryptology*, pages 47–53, 1985.

[19] Radu Stoleru, Pascal Vicaire, Tian He, and John A. Stankovic. Stardust: A flexible architecture for passive localization in wireless sensor networks. In *ACM SENSYS*, pages 57–70, 2006.

[20] Hailun Tan, Sanjay Jha, Diet Ostry, John Zic, and Vijay Sivaraman. Secure multi-hop network programming with multiple one-way key chains. In *ACM WiSec*, pages 183–193, 2008.

[21] Anthony D. Wood and John A. Stankovic. Denial of service in sensor networks. *IEEE Computer*, 35(10):54–62, 2002.

[22] Hao Yang, James Shu, Xiaoqiao Meng, and Songwu Lu. SCAN: Self-organized network layer security in mobile ad hoc networks. *IEEE Journal on Selected Areas in Communications*, 24(2):261–273, 2006.

[23] Bo Zhu, Venkata Gopala Krishna Addada, Sanjeev Setia, Sushil Jajodia, and Sankardas Roy. Efficient distributed detection of node replication attacks in sensor networks. In *IEEE ACSAC*, pages 257–267, 2007.

Digital Object Identifier: 10.4108/ICST.WICON2008.4796
http://dx.doi.org/10.4108/ICST.WICON2008.4796