International Journal of Security and Its Applications
Vol.8, No.2 (2014), pp.25-40
http://dx.doi.org/10.14257/ijsia.2014.8.2.03

Lightweight Attestation Scheme for Wireless Sensor Network

Shinsaku Kiyomoto and Yutaka Miyake
KDDI R & D Laboratories Inc.
kiyomoto@kddilabs.jp

Abstract

Wireless sensor networks (WSNs) have been deployed for several applications that use M2M communications. Malicious code propagation is considered a serious threat on WSNs. In this paper, we considere a simple, distributed remote attestation method for a WSN that does not require secret information, precise timing measurement, or a tamper-resistant device. Each node holds a value for the attestation of other nodes, and sensor nodes check the validity of each other's data. A sensor node terminates itself to prevent propagation of malicious code, when the sensor node finds a unexpected changes in stored program and data. We analyzed the probability of success in protecting the WSN against malicious code under random network models. The analyses confirmed the feasibility of the attestation scheme and we obtained a fundamental approach to finding an appropriate frequency for attestation.

1 Introduction

Sensor networks have been deployed for a wide variety of applications [2] such as environment sensing and security monitoring. A sensor network is an ad-hoc network consisting of many sensor nodes. Each sensor node is battery powered and equipped with integrated sensors, data processing capability, and short-range radio communications, but the computational resources of the sensor nodes are limited. Communicated data should be protected against eavesdropping and alteration by an adversary. Thus, secure communication channels established by cryptographic primitives are a common requirement for sensor networks. Another important security issue for wireless sensor networks is robustness against node-invasion attacks using malicious code. Sensor networks face the serious threat that an adversary can inject malicious code into the sensor nodes and then control those nodes. The malicious code propagates to other nodes using the communication channels. Several studies of malicious code propagation models in sensor networks have been published [5, 20, 35, 21, 22]. However, there has been insufficient analysis of the effectiveness of attestation as protection against malicious code propagation on wireless sensor networks. Remote attestation is an effective way to discover a compromised node. Generally, remote attestation methods implement a check mechanism on each node, and the mechanism sends proof of the integrity of the node to a verifier. A cryptographic primitive using a secret key or tamper-resistant module is needed to protect the check mechanism. For example,

ISSN: 1738-9976 IJSIA

a TPM chip is used to store the mechanism securely. However, the implementation in sensor nodes may be difficult due to the limited circuit size. A software-based approach using secret information may be possible to apply to sensor nodes, but we must first solve the problem of securely storing secret information on a sensor node. Another approach to remote attestation is to measure the timing of computation instead of using a secret key. This approach requires precise timing measurements for the attestation process.

In this paper, we present a model for analyzing the ability of attestation to find a sensor node that includes malicious code. The main focus of this paper is the analysis of a lightweight attestation scheme for resource-constrained devices such as sensor nodes. The present scheme is a distributed attestation scheme that does not use either a secret key or precise timing measurement. We consider a model of node infection under a random network topology and analyze the probability of successful protection using the attestation scheme.

The rest of the paper is organized as follows: Section 2 introduces related work, and Section 3 explains the model to be analyzed, including the adversary model and the distributed attestation scheme. Analysis results are presented in Section 4 and we conclude this paper in Section 5.

2 Related Work

There are several challenges involved in wireless sensor network security [3, 16, 33], and the main focus of existing research is on scalable trust management based on lightweight key management and distribution schemes appropriate for large-scale sensor networks [13, 11, 15]. Perrig *et al.* presented a suite of security blocks optimized for sensor networks [26]. Kong *et al.* proposed a localized public-key infrastructure mechanism based on a secret sharing scheme [23]. Ito *et al.* proposed a strongly resilient polynomial-based random key pre-distribution scheme (RPoK) [17] for wireless sensor networks such that a private sub-key is not directly stored on each sensor node. Ruj *et al.* addressed pairwise and triple-key establishment problems in wireless sensor networks [28], and they presented a novel concept of triple-key distribution in which a common secret key is established among three nodes.

Another issue is to ensure the robustness of sensor networks and to measure it. The random deployment of nodes results in uneven connectivity with critical nodes, making the network non-robust to node failure. Venuturumilli and Minai proposed a distribution algorithm [36] to design a robust, energy-efficient network that optimized the transmission range for each sensor node. Some studies considered robustness against random node failure [9, 8, 25], as in a general heterogeneous network that is highly optimized, and in which an adversary breaks some sensor nodes to obstruct the transmission of data.

Existing techniques require a secret key for computing checksum codes or precise timing measurements during the attestation protocol. Some techniques are based on tamper-resistant hardware [27]. Sailer *et al.* described an attestation technique [29] that relies on the TPM chip standardized by the Trusted Computing Group. BIND [34] is an attestation service using TPM, that ties the proof of what code has executed to the data the code produced. A timing-based attestation proposed by Kovah *et al.* [24] also used TPM hardware. Seshadri *et al.* presented a primitive mechanism [32] to achieve verifiable code execution on untrusted legacy hosts. Their scheme did not require any hardware support, but their target platform was mainly PCs. Jakobsson and Johansson presented a lightweight software-based attestation scheme [18] that used a secret key provided from a proxy. SWATT is a technique that performs attestation on embedded devices using a software verification function [30] that was also applied to remote attestation between a base station and sensor nodes [31]. Another software-based remote code attestation scheme proposed by AbuHmed *et al.* used a pseudo-random traversal [1] for computation of a checksum on a sensor node. Choi *et al.* proposed an attestation protocol [7] for sensor nodes. USAS [19] is a detection algorithm for compromised nodes on a wireless sensor network. DataGuard [39] is a dynamic data attestation scheme based on data boundary integrity for avoiding overflow attacks.

Our attestation scheme is described below.

- The present scheme is a distributed attestation scheme. Each node holds a value for the attestation of other nodes, and nodes use the values to check the validity of each other's data.
- The scheme uses a simple attestation function that reduces the burden of each attestation by just sending a value, thus allowing frequent attestations. No precise timing measurements or secret keys are required.
- The attestation is executed on the basis of a check for any changes in the quantity of data. There is no empty in a local storage area after the initialization process, and all the data in the entire local storage area is the target of the attestation. A compromised node may try to send valid data instead of altered data to a node that checks the validity of the data, but the victim node cannot hold both the valid data and the altered data without alteration of data in the local storage area, due to the lack of the space on the local storage area.
- When a sensor node finds any unexpected changes in the data, it terminates to prevent the propagation of malicious code.

The objective of this paper is to analyze the properties of the above simple attestation scheme in a certain network model.

3 Model

In this section, we present a network model, an attestation scheme, and an adversary model for our analysis.

3.1 Sensor Network

Generally, a sensor network consists of cheap sensor nodes that are able to communicate only with neighboring sensor nodes [10, 14, 40]. We assume a sensor network that consists of sensor nodes as described below.

Sensor node. The sensor nodes only have limited computing resources and short-range communication capabilities. The sensor nodes communicate with each other and with master nodes, if a secret key is shared. The sensor node has a CPU, a temporary cache for the CPU (primary memory), and local storage for programs and data. As in previous studies [38, 37, 4], we assume that malicious code is inserted into local storage and the target of the attestation scheme is data in local storage. All sensor nodes have a local storage area and we assume that the local storage area is divided into two parts: a program area $\mathcal{M}_\mathcal{P}$

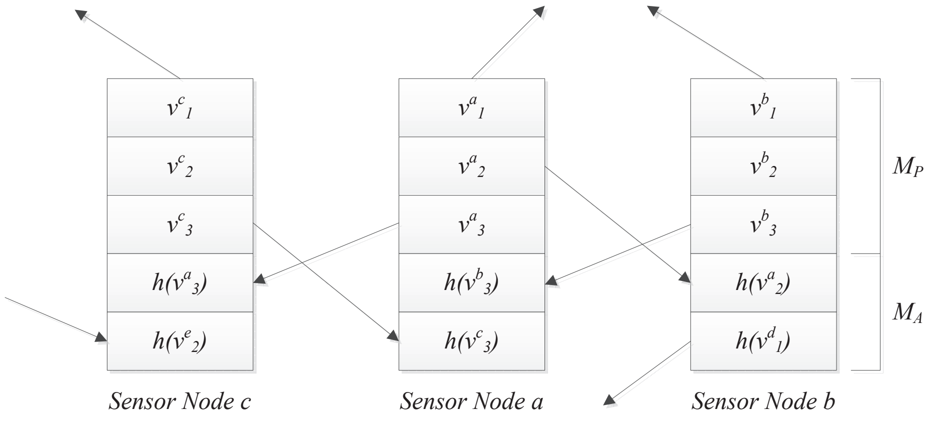

Figure 1. Distributed Attestation Scheme

for storing programs and related data and another memory area $\mathcal{M_A}$ for attestation. The program area $\mathcal{M_P}$ and the memory area $\mathcal{M_A}$ consist of l registers p_i $(0 \leq i \leq l-1)$ and m registers s_j $(0 \leq j \leq m-1)$, respectively. The total local storage area $\mathcal{M}$ denotes m_k $(0 \leq k \leq l+m-1)$. If the total size of the programs for the sensor node is smaller than the size of the $\mathcal{M_P}$, the remaining space of $\mathcal{M_P}$ is filled with random data. A sensor node can execute a checksum function $h(x)$, and can communicate with some other nodes in its vicinity. Sensor nodes construct an ad-hoc network with nearby sensor nodes, and data is transferred via the network. When a sensor node terminates, no communication of data can be received or transferred via the sensor node.

3.2 Attestation Scheme

An attestation scheme for the sensor node has to be able to discover malicious code (malware) to ensure the integrity of data in the program area. A sensor node that finds malicious code using the attestation scheme terminates itself to prevent further distribution of malicious code on the sensor network. The attestation scheme consists of the following two steps;

Initialization: A sensor node a obtains a value v_i^b in a register p_i^b from a nearby sensor node b, and the sensor node calculates the checksum value $h(v_i^b)$ where the function $h(x)$ is a certain checksum function, and stores the checksum value in $\mathcal{M_A}$. For simplicity, the checksum function is a bijective function that is defined as $h(x) : \{0,1\}^n \rightarrow \{0,1\}^n$, and it satisfies the *second-preimage resistance*. Sensor node a repeats the calculation and stores the checksum value of the register on another sensor node until the memory area $\mathcal{M_A}$ is full. All sensor nodes execute this operation in the initialization stage. Note that a register is chosen with the aim of avoiding duplication if possible, but some registers may not be chosen due to limited memory size $\mathcal{M_A}$. To avoid such situation, we should set $\mathcal{M_P} = \mathcal{M_A}$. To avoid duplication, the sensor node sends a re-selection request to another sensor node, when the data is already chosen by another sensor node. When a sensor node receives such a request, the sensor node randomly selects a different register again. We assume that malicious code cannot be injected during the initialization process. After the initialization process, all registers of all sensor nodes have no remaining space for additional code injection.

Attestation: A sensor node a randomly chooses a register p^a_x in the program area $\mathcal{M}_\mathcal{P}$ and the memory area $\mathcal{M}_\mathcal{A}$. If the register belongs to $\mathcal{M}_\mathcal{P}$, sensor node a obtains the corresponding checksum value $h_x = h(v^b_x)$ of the register p^b_y from another sensor node b that stores the checksum value h_x, computes the checksum value $h_{x'} = h(v^b_x)$ from the register p^a_x, and compares the value $h_{x'}$ with the checksum value h_x from another sensor node b. Otherwise, the register p^a_x holds the checksum value $h_z = h(v^b_z)$ of a register p^b_z on another sensor node b. The sensor node a obtains the register value v^b_z of the register p^b_z that corresponds to the checksum value h_z, calculates the checksum value $h_{z'} = h(v^b_z)$ from the register value v^b_z, and then compares the value $h_{z'}$ with the checksum value h_z stored in register p^a_x. Malicious code stored in the memory area $\mathcal{M}_\mathcal{A}$ can be found by checking not only $\mathcal{M}_\mathcal{P}$ but also $\mathcal{M}_\mathcal{A}$, randomly.

If node a finds a conflict between the values from node b, node a executes an additional step named a termination process as follows:

Termination: To avoid the propagation of malicious code, sensor node a requests termination from node b and then terminates itself, when node a finds node b suspicious by reason of a conflict between the checksum and register values. Node b also terminates when receiving a termination request from node a. Note that sensor node a cannot distinguish two cases: checked data on sensor node b is infected or its own data (in sensor node a) is infected. Thus, both sensor nodes should be terminated.

The terminations should end propagation of the malicious code. When a node is terminated, no communication passes from the compromised node or via the node, which isolates the malicious code. In another case, the potentially malicious node may not actually be infected by malicious code. In this case, it is possible that the node checking the potentially malicious node has malicious code, leading to the conflict between the values. In either case, the malicious code cannot propagate to other nodes if both nodes are terminated.

Extension to Multi-Register Check. We can extend this scheme to a multi-register checking scheme, where we use $h(x) : \{0,1\}^m \to \{0,1\}^n$ $(m \geq n)$. A sensor node a checks some registers such as v^b_1, v^b_2, and v^b_3 in a step, where the node a computes $h_x = h(v^b_1||v^b_2||v^b_3)$ and hold the checksum value h_x, even though the node a obtain the all register values via a communication channel with the node b in the attestation step. Note that the node can only check some registers from the same node.

3.3 Adversary Model

We assume that the adversary is not able to either extend or replace the hardware of sensor nodes to increase memory size. The adversary model is defined as follows;

Adversary. *The objective of an adversary is to control sensor nodes using malicious code. An adversary will try to install malicious code on sensor nodes,replacing any register(s) in $\mathcal{M}_\mathcal{P}$ or $\mathcal{M}_\mathcal{A}$ with the malicious code. For simplicity, we specify that the size of the malicious code is the same as the size of a register.*

Now, we consider attacks described in existing papers [31, 6, 18]. There are nine different attacks that might be made on wireless sensor networks, and we will focus on an analysis of three types of attack, the data substitution attack, the memory copy attack, and the checksum value modification attack. as the other seven attacks are not required to be considered or just beyond the scope of this paper as explained below.

- **Guessing Attack.** An adversary may guess the value that is requested by a checking node. However, it can be assumed that the probability of success of such a guess is negligibly small.
- **Data Substitution Attack.** An adversary may attempt to change some location in the memory region. The adversary modifies selected portions of the program area and maintains a copy of the unmodified portions elsewhere in the program area or the memory area. The attestation request is forced to refer to a location where the corresponding unmodified portion exists. We should consider this attack in our evaluation.
- **Memory Copy Attack.** An adversary maintains a copy of the unmodified values of a program area or a memory area while injecting malicious code. Either the correct code is copied to another location in memory and the malicious code is injected at the location of correct code, or the attacker simply injects the malicious code in some other location in the area. We should consider this attack in our evaluation.
- **Computation Attack.** This attack includes on-the-fly execution of a correct checksum value of the register and replacing an input of the attestation algorithm. To compute a correct checksum value (or just input a correct checksum value to the attestation algorithm), the attacker has to hold a correct register value or a correct checksum value. Thus, this attack is identical to the memory copy attack.
- **Parallelization/Re-Ordering Attack.** This type of attack is for timing-based attestation schemes. The adversary speeds up the self-checksum computation by executing operations in parallel or by re-ordering the operations. This speedup allows the adversary to compute the correct checksum faster than expected. Our attestation scheme is not based on a timing measurement. Thus, this type of attack need not be considered.
- **Replay Attack.** An infected node may send a correct value that was obtained in a previous attestation; however, to send the value, the node must store the value in the local storage area. This attack is not realistic due to the limited size of the local storage area. Thus, an attacker will try to use a data substitution attack or a memory copy attack.
- **Impersonation Attack.** An adversary could interrupt the execution of the attestation process. There is no way to carry out an impersonation attack in our scheme. A sensor node is checked by the nearest nodes, and it is assumed that no node is able to impersonate another and in this way come between an attester and a subject.
- **Compression Attack.** A compression attack [6] is an extension of the memory copy attack. An adversary compresses data in the local storage area in order to obtain enough memory space to store malicious code. Techniques [38, 37] for avoiding compression attacks have been presented. Thus, we will not consider a protection mechanism against a compression attack here.

Table 1. Parameters for Experiments

Parameter	Value
N	100
$\mathcal{M}_{\mathcal{P}}$	4
$\mathcal{M}_{\mathcal{A}}$	4
P_L	0.01, 0.02, 0.03, 0.04
r	$0.00001 \sim 1$
Termination Process	Execution/No Execution

- **Checksum Value Modification Attack.** The attacker injects a malicious code to p_i^a in sensor node a, and then replace the checksum value $h_j = H(^a_i)$ stored in sensor node b to the checksum value of the malicious code. When both register and checksum values are altered, the attestation step is correctly executed despite the presence of malicious code. However, it would be difficult for an adversary to find the correct pair of register and checksum values. We will consider this attack in our experiments.

Both the data substitution attack and the memory copy attack swap or copy a valid value from a register to another register. The adversary refers to the swapped or copied value of the register instead of the actual value in the register and is thus able to answer with the correct value. However, the entire local storage area is covered by the attestation, and the registers are randomly checked by other nodes. If any value is copied to a register, another value in the register is lost. By swapping, the value in a register is moved to another register, and the register is occupied by malicious code. We assume that the selection of the register to be checked in the attestation process is uniformly random, making it is difficult for an adversary to guess which register will be checked in the next attestation. The attestation can find the malicious code and detect the difference in the values caused by copying, and by checking the entire local storage area. In our experiment, we simply note that these three attacks all involve the alteration of data in a register.

4 Analysis

In this section, we discuss an analysis of the attestation scheme in a random network topology. We evaluate the propagation of malicious code and estimate the appropriate rate of attestation for protecting a sensor network against the malicious code.

4.1 Experiment

To construct a network topology for experiments, we used a *random graph* [12] that has N nodes connected to n edges that are chosen randomly from $N(N-1)/2$, where N is the number of nodes. The random network is generated based on the probability P_L that a sensor node selects a register of another sensor node to compute a checksum value in the initialization process. That is, the selection of a register is equivalent to the edge between two nodes; thus, a random network is generated to select the register of other sensor nodes with probability P_L until the registers of $\mathcal{M}_{\mathcal{A}}$ are full. The probability P_L controls the density of the random network. That is, larger P_L generates more contrasting

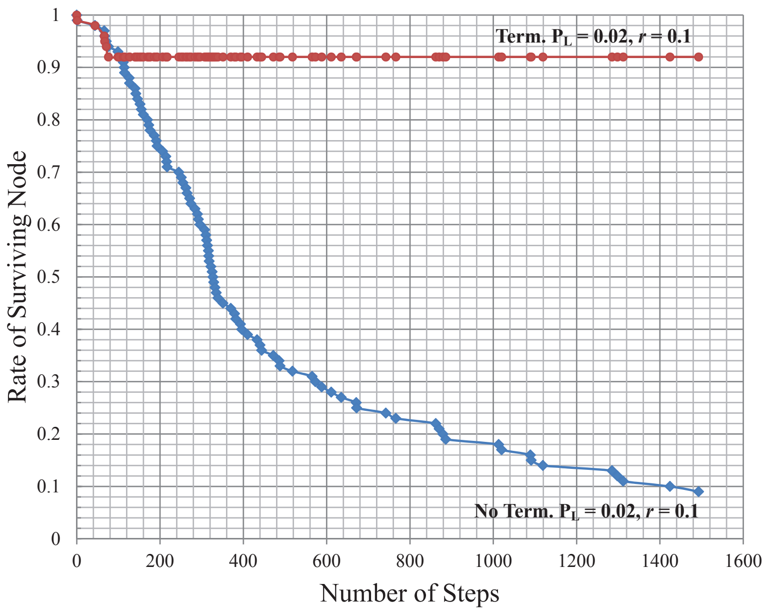

Figure 2. Example of Experimental Result

dense networks like clustered networks. The infection process of malicious code will be treated as a probabilistic behavior and simulated as follows, similarly to existing work:

1. The register of a sensor node a that has malicious code is selected randomly.
2. A register of sensor nodes connected to sensor node a, is randomly selected.
3. The selected register is infected by malicious code. If both a register value and its checksum value are compromised, no further attestation is executed between them.

We do not assume a "smart" adversary that understands the network topology and efficiently injects a malicious code into sensor nodes. Such a strong adversary model will be considered in a future study.

Using the above simulation, experiments were executed under the following conditions:

- When all sensor nodes are infected by a malicious program, the experiment is terminated.
- When all sensor nodes are terminated, the experiment is terminated.
- When no sensor node that includes malicious code exists, the experiment is terminated.
- At each step, either attestation process execution or execution of malicious code infection is randomly selected; malicious code infection is selected when $-log(p_x)/rN_m < -log(p_y)$. The parameter r is the fraction of infection/attestation, and infection or attestation is selected for each simulation step according to r. The parameter N_m denotes the number of registers having malicious code at that point. The random numbers p_x, and p_y, $(0 < p_x, p_y \leq 1)$ are used to generate a probabilistic event based on exponential random numbers.

We used a parameter set for the experiments as shown table 1. We evaluated the number of steps to find a compromized register and the number of steps to stop the experiments by

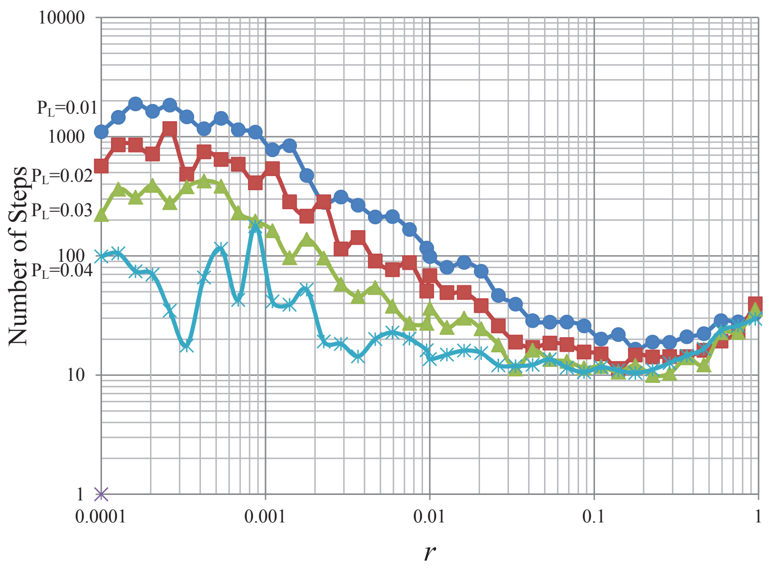

Figure 3. Number of Steps to Discover Malicious Code

four values of P_L and values of r $(0.00001 \leq r \leq 1)$. We also evaluated the fraction of surviving nodes, i.e. those that had not been invaded and terminated, when any experiments were stopped. To evaluate the effect of the termination process, the experiments include two cases: experiments with the termination process, and experiments without it.

4.2 Result

We executed a maximum of 400,000 steps in each experiment, and all results are the average values of 100 experiments. An example of experimental results is shown in figure 2. The figure includes two cases: (a) the termination case in which a sensor node executed the termination process when it discovered malicious code and (b) the no-termination case in which a sensor node executed no additional operation, even when malicious code was found. The termination case removed all malicious code from the sensor network after 76 steps, making the fraction of surviving nodes 0.92 over 76 steps. Figure 3 shows the number of steps required to discover malicious code, when P_L = 0.01, 0.02, 0.03, and 0.04. The number of steps required to discover malicious code decreased as r increased, and cases around $r = 0.03 \sim 0.3$ had low step values. Note that the propagation of malicious code was slow with small r; thus, many steps were required to discover malicious code. The number of steps to terminate an experiment is shown in Figure 4. Cases in which $r \leq 0.03$ were terminated within 6,000 steps. This means that malicious code was removed from the sensor network within 6,000 steps by the attestation and termination of sensor nodes. Those results indicate that we should select an r of less than 0.03. Figure 5 shows the fraction of surviving nodes when the experiment was terminated. Where $P_L = 0.01$, the results of the no-termination case were similar to those of the termination case because malicious code efficiently propagated to other nodes due to the sparseness of the network, and the propagation speed was slower than in other cases. The survival fraction in the no-termination cases with $P_L = 0.02, 0.03, 0.04$ was dramatically reduced as r increased. However, in cases that used the termination operation, the survival fraction remained high, reaching $r = 0.03$. Thus, the attestation scheme, including the termination process, was

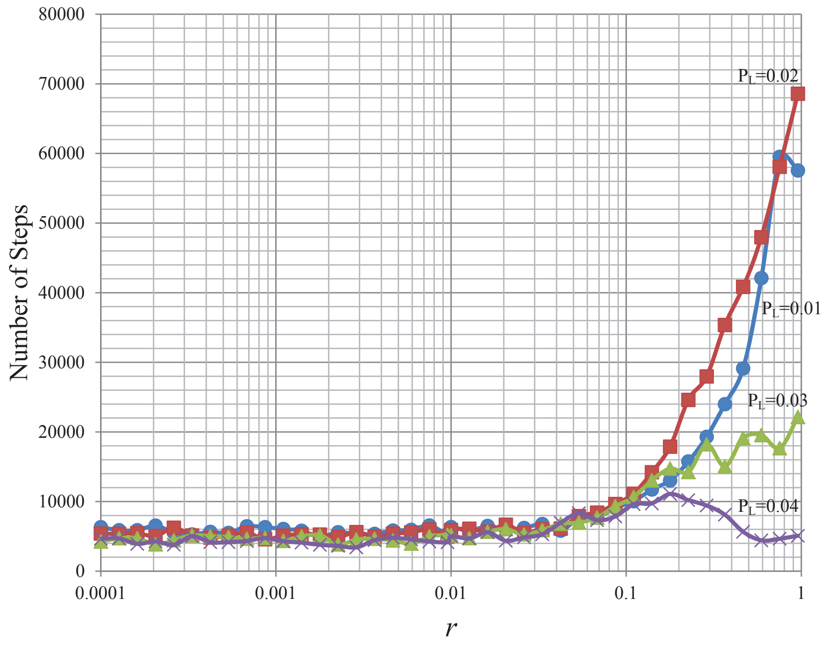

Figure 4. Number of Termination Steps

effective in protecting sensor networks against the propagation of malicious code when we selected an appropriate r, such as $r \leq 0.03$. We should select an appropriate r when configuring a sensor network, taking into account the network structure, and the local storage size of sensor nodes, as well as the expected rate of attestation for an assumed propagation model of malicious code.

4.3 Limitations of the Scheme

The attestation scheme discussed in this paper is a simple method for finding an invalid value in the memory of a sensor node. Thus, the following limitations remain:

- **Failure of Termination.** We do not assume that the termination process cannot be infiltrated by a malicious program in the experiment. All programs for attestation and termination should be stored in read-only-registers, so the adversary cannot alter the programs. However, a malicious program may intercept all operations of a valid program, so that a sensor node that includes malicious code cannot be terminated. How to ensure a valid termination process despite malicious code injection is an open issue, even though a pairing node having no malicious code can be terminated, and the compromised node may have no connection to other surviving nodes.
- **Additional Space in Memory.** We assume that an adversary stores malicious code in a local storage area. The sensor node still has a primary memory or other additional memory for temporary data like cache memory, and the adversary may try to upload malicious code or the valid data of a register into the memory, even though the size of the memory is much smaller than the areas $\mathcal{M}_{\mathcal{P}}$ and $\mathcal{M}_{\mathcal{A}}$. This is the same issue that existing schemes face, and we will evaluate the feasibility of this kind of attack by using a prototype system implemented on a real sensor node device.

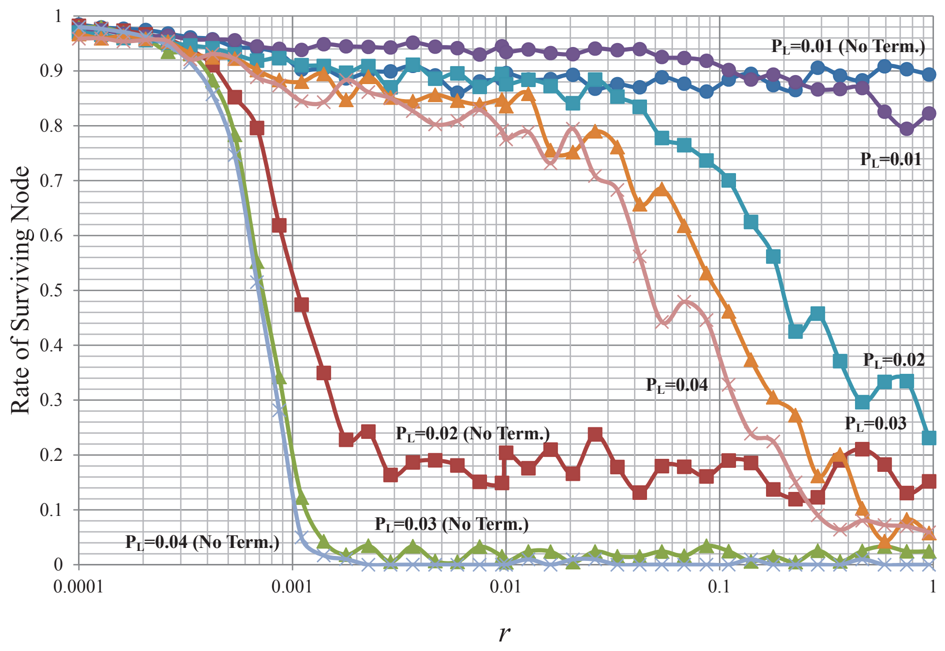

Figure 5. Rate of Surviving Nodes

- **No-response Attack.** A node invaded by malicious code may reply with no answer to another node that executes the attestation process on the node. Furthermore, an adversary may interrupt the communication of data between nodes. In these situations, the checking node cannot distinguish whether the node is infected or communication has failed. We will consider a feasible solution for this, although a simple solution is to judge it infected in such a case.
- **DoS Attacks.** Protection against denial-of-service (DoS) attacks is beyond the scope of this paper. The attestation scheme cannot protect nodes against DoS attacks and may be weak against DoS attacks. For example, an adversary may alter attestation data sent from a node, and another node may judge that the altered data is malicious code. The nodes are terminated; thus the adversary can reduce the number of surviving nodes by altering attestation data. Furthermore, confidentiality of transaction data for attestation may be required, in addition to protection against data alteration. Using secure communication based on existing techniques introduced in section 2 is a reasonable solution to prevent data alteration and eavesdropping.

4.4 Extension of Sensor Network Model

Another discussion point is the extension of the sensor network model used in this paper. For simplicity of the discussion, we only considered a network consisting of sensor nodes. Some existing wireless sensor networks consist of cheap sensor nodes that can communicate only with neighboring sensor nodes and one other type of node, a master node, that communicates with sensor nodes and belongs to an upper layer network that consists of other master nodes. The master nodes are needed to gather information from sensor nodes and transfer the information to a system that monitors all sensor nodes. The master node also has a CPU, temporary cache, and local storage. The master node has a large local storage area $\mathcal{M}_{\mathcal{A}}$, and holds many values of $h(v_x)$ of sensor nodes around the master node. Some sensor networks consist of sensor nodes and a master node, and all data gathered by the sensor nodes is sent out of the sensor network via the master node.

A master node holds many checksum values and may be able to check many registers of sensor nodes; thus, master nodes may make a sensor network more robust against malicious code. On the other hand, if malicious code is injected into a master node, many sensor nodes will be influenced and all communication via the master node will be controlled by the malicious code. Moreover, a master node is generally more expensive than a sensor node. Thus, the allocation of master nodes in a sensor network will be an important issue, when we consider how and whether to use both master nodes and sensor nodes.

The structure of the network has to be considered in real use cases. In our experiment, all sensor nodes are evaluated under the assumption that their roles are identical. However, some sensor nodes have a more important role in a real sensor network. For example, if sensor nodes around a master node are compromised, other sensor nodes cannot send information to the master node. The impact of breaking a sensor node at the core of a sensor network may be larger than that of breaking a sensor node on the edge of the network. An index denoting the importance of each node should be defined, and the damage from the malicious code propagation should be estimated according to that index. By using the index when we design a sensor network, we can consider optimization of the network structure against malicious code propagation.

Another important issue is how to update programs in registers. A secure update scheme should be considered to distinguish the software update from malicious code injection.

Our results provide a first step towards the design of robust and self-protecting WSNs. In future research, we will consider more realistic sensor networks including two types of nodes: sensor nodes and master nodes. Furthermore, we will use a real network structure for experiments, and consider the optimization of the network.

5 Conclusion

In this paper, we have presented the results of an analysis of a distributed remote attestation scheme for tiny sensor devices. We evaluated the scheme using random networks of sensor nodes and presented optimal settings for the frequency of attestation. The results suggested that the attestation scheme was effective with certain parameter settings. We also discussed some limitations of the attestation scheme, even though they are caused by the trade-off between cost of attestation and perfection level of the attestation. The scheme was applicable to wireless sensor networks, preventing the propagation of malicious code by terminating unconvincing sensor nodes.

An important issue is the power consumption of the attestation scheme, as it requires frequent attestations for each sensor node. Evaluation of the power consumption using actual sensor nodes will be needed for practical use. We will implement the attestation scheme on an actual sensor node device and evaluate the power consumption of the attestation scheme in our future research.

References

[1] T. AbuHmed, N. Nyamaa, and DaeHun Nyang. Software-based remote code attestation in wireless sensor network. In *Proc. of IEEE Global Telecommunications Conference 2009, (GLOBECOM 2009)*, pages 1–8, 2009.

[2] I.F. Akyildiz, Weilian Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *Communications Magazine, IEEE*, 40(8):102 – 114, 2002.

[3] Madhukar Anand, Eric Cronin, Micah Sherr, Matt Blaze, Zachary Ives, and Insup Lee. Sensor network security: more interesting than you think. In *Proc. of the 1st USENIX Workshop on Hot Topics in Security*, pages 25–30, 2006.

[4] Frederik Armknecht, Ahmad-Reza Sadeghi, Steffen Schulz, and Christian Wachsmann. Towards provably secure software attestation. Cryptology ePrint Archive, Report 2013/083, 2013. `http://eprint.iacr.org/2013/083`.

[5] M. Badakhshan and D. Arifler. Simulation based analysis of spreading dynamics of malware in wireless sensor networks. In *2007 International Conference on Sensor Technologies and Applications, SensorComm 2007*, pages 164–169, 2007.

[6] Claude Castelluccia, Aurelien Francillon, Daniele Perito, and Claudio Soriente. On the difficulty of software-based attestation of embedded devices. In *Proc. of the 16th ACM conference on Computer and communications security*, CCS '09, pages 400–409, 2009.

[7] Yong-Sik Choi, Young-Jun Jeon, and Sang-Hyun Park. A study on sensor nodes attestation protocol in a wireless sensor network. In *Proc. of The 12th International Conference on Advanced Communication Technology (ICACT)*, volume 1, pages 574–579, 2010.

[8] P. Crucitti, Latora V., M. Marchiori, and Rapisarda A. Error and attack tolerance of complex networks. In *Physica A*, volume 304, Issue 1-3, pages 388–394, 2004.

[9] Anthony H. Dekker and Bernard D. Colbert. Network robustness and graph topology. In *Proc. of the 27th Australasian conference on Computer Science*, volume 26 of *ACSC '04*, pages 359–368, 2004.

[10] Peter Desnoyers, Deepak Ganesan, and Prashant Shenoy. Tsar: a two tier sensor storage architecture using interval skip graphs. In *Proc. of the 3rd international conference on Embedded networked sensor systems*, SenSys '05, pages 39–50, 2005.

[11] Wenliang Du, Jing Deng, Y.S. Han, Shigang Chen, and P.K. Varshney. A key management scheme for wireless sensor networks using deployment knowledge. In *Proc. of 23rd Annual Joint Conference of the IEEE Computer and Communications Societies*, INFOCOM 2004, 2004.

[12] P. Erdӧs and A. Rènyi. On random graphs. I. *Publicationes Mathematicae*, 6:290–297, 1959.

[13] Laurent Eschenauer and Virgil D. Gligor. A key-management scheme for distributed sensor networks. In *Proc. of the 9th ACM conference on Computer and communications security*, CCS '02, pages 41–47, 2002.

[14] Omprakash Gnawali, Ki-Young Jang, Jeongyeup Paek, Marcos Vieira, Ramesh Govindan, Ben Greenstein, August Joki, Deborah Estrin, and Eddie Kohler. The tenet architecture for tiered sensor networks. In *Proc. of the 4th international conference on*

Embedded networked sensor systems, SenSys '06, pages 153–166, 2006.

[15] Lin He, Yi-Ying Zhang, Lei Shu, A.V. Vasilakos, and Myong-Soon Park. Energy-efficient location-dependent key management scheme for wireless sensor networks. In *Proc. of 2010 IEEE Global Telecommunications Conference*, GLOBECOM 2010, 2010.

[16] Md. Safiqul Islam and Syed Ashiqur Rahman. Anomaly intrusion detection system in wireless sensor networks: Security threats and existing approaches. In *IJAST*, volume 36, pages 1–8, 2011.

[17] H. Ito, A. Miyaji, and K. Omote. Rpok: A strongly resilient polynomial-based random key pre-distribution scheme for multiphase wireless sensor networks. In *Proc. of 2010 IEEE Global Telecommunications Conference*, GLOBECOM 2010, 2010.

[18] M. Jakobsson and K.-A. Johansson. Practical and secure software-based attestation. In *2011 Workshop on Lightweight Security Privacy: Devices, Protocols and Applications (LightSec)*, pages 1–9, 2011.

[19] Xinyu Jin, P. Putthapipat, Deng Pan, N. Pissinou, and S.K. Makki. Unpredictable software-based attestation solution for node compromise detection in mobile wsn. In *Proc. of 2010 IEEE GLOBECOM Workshops (GC Wkshps)*, pages 2059–2064, 20101.

[20] V. Karyotis, M. Grammatikou, and S. Papavassiliou. On the asymptotic behavior of malware-propagative mobile ad hoc networks. In *2007 IEEE Internatonal Conference on Mobile Adhoc and Sensor Systems, MASS 2007*, pages 1–9, 2007.

[21] M. H R Khouzani and S. Sarkar. Dynamic malware attack in energy-constrained mobile wireless networks. In *2010 Information Theory and Applications Workshop (ITA)*, pages 1–11, 2010.

[22] M. H R Khouzani and S. Sarkar. Maximum damage battery depletion attack in mobile sensor networks. *IEEE Transactions on Automatic Control*, 56(10):2358–2368, 2011.

[23] Jiejun Kong, Z. Petros, Haiyun Luo, Songwu Lu, and Lixia Zhang. Providing robust and ubiquitous security support for mobile ad-hoc networks. In *Proc. of 9th International Conference on Network Protocols*, pages 251 –260, 2001.

[24] X. Kovah, C. Kallenberg, C. Weathers, A. Herzog, M. Albin, and J. Butterworth. New results for timing-based attestation. In *2012 IEEE Symposium on Security and Privacy (S & P)*, pages 239–253, 2012.

[25] G. Paul, T. Tanizawa, Havlin S., and Stanley H. E. Optimization of robustness of complex networks. In *The European Physical Journal B*, volume 38, pages 187–191, 2004.

[26] Adrian Perrig, Robert Szewczyk, J. D. Tygar, Victor Wen, and David E. Culler. Spins: security protocols for sensor networks. *Journal of Wireless Networks*, 8:521–534, 2002.

[27] Nick L. Petroni, Jr., Timothy Fraser, Jesus Molina, and William A. Arbaugh. Copilot - a coprocessor-based kernel runtime integrity monitor. In *Proceedings of the 13th conference on USENIX Security Symposium - Volume 13*, SSYM'04, pages 13–13, 2004.

[28] S. Ruj, A. Nayak, and I. Stojmenovic. Fully secure pairwise and triple key distribution in wireless sensor networks using combinatorial designs. In *Proc. of INFOCOM, 2011, IEEE*, pages 326–330, 2011.

[29] Reiner Sailer, Xiaolan Zhang, Trent Jaeger, and Leendert van Doorn. Design and

implementation of a tcg-based integrity measurement architecture. In *Proc. of the 13th conference on USENIX Security Symposium - Volume 13*, SSYM'04, pages 16–16, 2004.

[30] A. Seshadri, A. Perrig, L. van Doorn, and P. Khosla. Swatt: software-based attestation for embedded devices. In *Proc. of 2004 IEEE Symposium on Security and Privacy*, pages 272–282, 2004.

[31] Arvind Seshadri, Mark Luk, Adrian Perrig, Leendert van Doorn, and Pradeep Khosla. Scuba: Secure code update by attestation in sensor networks. In *Proceedings of the 5th ACM workshop on Wireless security*, WiSe '06, pages 85–94, 2006.

[32] Arvind Seshadri, Mark Luk, Elaine Shi, Adrian Perrig, Leendert van Doorn, and Pradeep Khosla. Pioneer: verifying code integrity and enforcing untampered code execution on legacy systems. In *Proc. of the twentieth ACM symposium on Operating systems principles*, SOSP '05, pages 1–16, 2005.

[33] Kalpana Sharma, M.K. Ghose, Deepak Kumar, Raja Peeyush Kumar Singh, and Vikas Kumar Pandey. A comparative study of various security approaches used in wireless sensor networks. In *IJAST*, volume 17, pages 31–44, 2010.

[34] E. Shi, A. Perrig, and L. van Doorn. Bind: a fine-grained attestation service for secure distributed systems. In *Proc. IEEE Symposium onof Security and Privacy 2005*, pages 154–168, 2005.

[35] Yurong Song and Guo ping Jiang. Modeling malware propagation in wireless sensor networks using cellular automata. In *2008 International Conference on Neural Networks and Signal Processing*, pages 623–627, 2008.

[36] A. Venuturumilli and A. Minai. Obtaining robust wireless sensor networks through self-organization of heterogeneous connectivity. In *Proc. of the 6th International Conference on Complex Systems*, 2006.

[37] B. Vetter and D. Westhoff. Simulation study on code attestation with compressed instruction code. In *Proc. of 2012 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, pages 296–301, 2012.

[38] Yi Yang, Xinran Wang, Sencun Zhu, and Guohong Cao. Distributed software-based attestation for node compromise detection in sensor networks. In *Proc. of the 26th IEEE International Symposium on Reliable Distributed Systems, SRDS 2007*, pages 219–230, 2007.

[39] Dazhi Zhang and Donggang Liu. Dataguard: Dynamic data attestation in wireless sensor networks. In *Proc. of IEEE/IFIP International Conference on Dependable Systems and Networks (DSN) 2010*, pages 261–270, 2010.

[40] Rui Zhang, Jing Shi, and Yanchao Zhang. Secure multidimensional range queries in sensor networks. In *Proc. of the 10th ACM international symposium on Mobile ad hoc networking and computing*, MobiHoc '09, pages 197–206, 2009.