Hindawi Publishing Corporation
Journal of Sensors
Volume 2015, Article ID 874782, 6 pages
http://dx.doi.org/10.1155/2015/874782

Research Article

Distributed Software-Attestation Defense against Sensor Worm Propagation

Jun-Won Ho

Department of Information Security, Seoul Women's University, Seoul 139-774, Republic of Korea

Correspondence should be addressed to Jun-Won Ho; jwho@swu.ac.kr

Received 15 November 2014; Accepted 6 March 2015

Academic Editor: Fanli Meng

Wireless sensor networks are vulnerable to sensor worm attacks in which the attacker compromises a few nodes and makes these compromised nodes initiate worm spread over the network, targeting the worm infection of the whole nodes in the network. Several defense mechanisms have been proposed to prevent worm propagation in wireless sensor networks. Although these proposed schemes use software diversity technique for worm propagation prevention under the belief that different software versions do not have common vulnerability, they have fundamental drawback in which it is difficult to realize the aforementioned belief in sensor motes. To resolve this problem, we propose on-demand software-attestation based scheme to defend against worm propagation in sensor network. The main idea of our proposed scheme is to perform software attestations against sensor nodes in on-demand manner and detect the infected nodes by worm, resulting in worm propagation block in the network. Through analysis, we show that our proposed scheme defends against worm propagation in efficient and robust manner. Through simulation, we demonstrate that our proposed scheme stops worm propagation at the reasonable overhead while preventing a majority of sensor nodes from being infected by worm.

1. Introduction

Since sensor nodes are usually deployed in unattended manner, they could be physically captured by an attacker. By compromising the captured nodes and launching various attacks with these compromised nodes, an attacker can have a malicious impact on the sensor networks. However, it takes substantial amount of time to physically capture and compromise a large number of nodes and thus it is not suitable for fast-spread node compromise in sensor networks. To mitigate this limitation, an attacker could launch worm attacks in which self-propagating worm originates from a few number of compromised nodes and is widely spread over the network in order to convert the benign nodes to the malicious nodes. Because worm could be very quickly propagated over the network, it is very important to detect and stop worm propagation as soon as possible in order to minimize the number of nodes infected by worm.

To meet this goal, several schemes have been developed for worm defense in the sensor network [1–3]. The key idea of these schemes prevents worm propagation by enforcing two adjacent nodes to have distinct software versions, leading to worm dissemination failure between two adjacent nodes. However, if different software versions can have common vulnerability, these schemes will not work because the shared vulnerability can be exploited for worm to spread between two adjacent nodes. Therefore, more active defense mechanisms without this drawback should be required rather than passive techniques such as software diversity.

To achieve this requirement, we propose on-demand software-attestation based scheme to actively defend worm propagation in sensor network. Software attestation is a malicious node detection technique in which the flash image of sensor node is checked with the normal image and the subverted flash image is detected [4–8]. We adapt software attestation to actively defend the network against worm attacks. More specifically, every node randomly chooses a list of attestees against which it is in charge of attesting. Whenever it receives packets from the neighboring nodes that belong to the attestee list, it performs software attestations against these neighbors and detects the nodes infected by worm, leading to hindrance of worm spread over the network.

We analytically show that our proposed scheme hampers sensor worm propagation robustly and efficiently. In addition, we evaluate the performance of our proposed scheme through simulation. The simulation results illustrate that at most 3655.7 attestations per time slot are required for worm propagation barrier on average and at most 54.5 attestations per time slot are required for benign traffic on average, where the network consists of 1000 nodes. Furthermore, the faction of infected nodes does not exceed 7.86% and 23.26% when the size of the attestee list stored in a node is 200 and 100, respectively. These results indicate that our proposed scheme restrains a majority of sensor nodes from being infected at the reasonable attestation overhead.

The rest of the paper is organized as follows. In Section 2, we present the related work. In Section 3, we describe the network assumptions and adversary models for our scheme. In Section 4, we propose a worm propagation defense scheme based on software attestation in sensor networks and then present the security and performance analyses of our proposed scheme. In Section 5, we present the simulation results of our proposed scheme. In Section 6, we conclude the paper.

2. Related Work

Recent research work demonstrates that malicious code injection and worm propagation attacks can be launched against sensor devices [9–11]. Goodspeed [10] leveraged format string and buffer overflow vulnerabilities to execute malicious codes on the MSP430-based TelosB motes. Gu and Noorani [11] demonstrated that malicious codes can be temporarily executed on Mica2 motes and spread to neighboring motes. Francillon and Castelluccia [9] showed that malicious codes can be permanently and remotely injected into the program memory of Atmel AVR-based sensor motes such as the MicaZ mote. To the best of our knowledge, this work is the first to shatter a common opinion that permanent malicious code injection is impossible on sensor devices designed in the Harvard architecture.

To defend against the sensor worm attacks, several schemes are proposed to prevent worm propagation in sensor networks [1–3, 12]. Yang et al. [3] first applied a software diversity technique to prevent worm propagation in sensor networks. The key idea of this work is that the flash image of the nodes in a cell A is different to the one of the nodes in the A's adjacent cells, where the network is broken into a set of grid cells. The proposed scheme will prevent a worm from propagating through two adjacent cells as long as different versions of flash image do not have any common vulnerabilities. However, it is hard to automatically guarantee that different versions of flash image have distinct vulnerabilities. If we can find them out, we can fix them and thus we do not need this scheme. On the other hand, when all versions of flash image have the common vulnerability that has not yet been detected, worm can propagate through two adjacent cells by exploiting this vulnerability, leading to the entire network infection. Gui et al. [1] explored how the software diversity technique defers worm infection through active sensor nodes when they are selected by random node scheduling. Liu et al. [2] proposed a role-based graph coloring scheme for software diversity in sensor networks. Since both [1, 2] leverage the software diversity technique for worm propagation prevention, they have the same limitations as in [3]. Sun et al. [12] proposed an immunization-based worm defense in which the network operator employs a set of immune nodes to prevent or slow down worm propagation. However, [12] mainly focuses on how to select the immune nodes while not presenting any details of how immune nodes can stop worm propagation.

Software attestation is a useful technique to discern the subverted flash image of malicious node with virtually zero false positives. Several researchers applied software attestation for malicious node detection in sensor networks [4–8]. More specifically, the base station determines whether the flash image has been maliciously altered by attesting randomly chosen portions of flash image or the entire one [5, 6]. In both [4, 7], the attestation process is performed in localized and distributed manner. In [8], the dynamic attestation process is performed against the executing flash image.

However, these schemes do not specify how often the attesters need to perform the attestation against the attestees. Our proposed scheme performs software attestation in on-demand manner in order to block sensor worm propagation in the network.

3. Preliminaries

In this section, we first describe the network assumptions and then the adversary model for our proposed scheme.

3.1. Network Assumptions. We assume a *static* sensor network that is widely adopted for sensor node deployment. In static sensor network, each sensor node never alters its position after being deployed in the network. Furthermore, we assume that a node can directly communicate with one-hop neighboring nodes within its communication vicinity. Through multihop communication, a node is thus able to communicate with nodes out of one-hop communication range.

3.2. Adversary Models. We assume that the attacker can compromise a set of nodes, inject worm codes into these compromised nodes, and make them worm originators, leading to widespread worm propagation. Moreover, we assume that the flash images of the infected nodes are subverted such that these infected nodes perform various malicious activities.

In sensor networks, the normal network operations such as data aggregation, clustering, and time synchronization are generally performed through the local communications rather than arbitrary peer-to-peer ones [13], and accordingly the normal network traffic is usually propagated in a neighbor-to-neighbor manner. In order to minimize the chance of being detected, the attacker would like to have worm propagation pattern be similar to the ordinary pattern of the normal network traffic in sensor networks. From this perspective, it is reasonable to assume that the attacker adopts a hop-by-hop worm propagation strategy in which worm is propagated in a neighbor-to-neighbor manner.

Although hop-by-hop propagation strategy indicates how worm is spread over the sensor network, it does not specify how many nodes can be infected by worm within a certain

amount of time. To quantify the worm infection quota, we use the discrete time version of simple epidemic model [14], which is widely employed to model worm propagation in wired and wireless networks. In simple epidemic model, a sensor node is assumed to have two states: *susceptible* and *infectious*. All sensor nodes besides worm originators, which are in infectious state, are initialized to the susceptible state. If the susceptible nodes are infected by worm, their state will be switched to infectious. Furthermore, the infection quota in units of time slots is given by

$$I_t = (1 + \epsilon n) I_{t-1} - \epsilon I_{t-1}^2, \tag{1}$$

where n is the total number of sensor nodes in the network and ϵ is the pairwise infection rate [14]. I_0 represents the number of worm originators. I_t is the cumulative infection quota from the 0th time slot to the tth time slot. Therefore, the infection quota in the tth time slot is computed as $I_t - I_{t-1}$.

In the simple epidemic model together with hop-by-hop propagation strategy, the network topology could make the infectious nodes have fewer susceptible neighbors than the infection quota. As a result, the worm infection quota might not be fulfilled in a time slot. To soothe this problem, we slightly modify the discrete time version of simple epidemic model such that the infection quota deficit in the current time slot is carried over to the next time slot, meeting the total infection quota over the entire time period. In the modified version of simple epidemic model, the infection quota in the $t + 1$st time slot is calculated as $I_{t+1} - I_t + b_t$ $(t \geq 1)$, where b_t is the infection quota deficit in the tth time slot and the infection quota in the first time slot is $I_1 - I_0$.

4. Sensor Worm Propagation Defense Using On-Demand Software Attestation

In this section, we first describe the details of our sensor worm propagation defense scheme and then analyze the security and performance of it.

4.1. Protocol Description. Let us denote n as the total number of nodes in the network. Each node has distinct ID and accordingly there are n distinct IDs in the network. Moreover, we assume that the entire time domain is divided into a series of time slots. After being deployed, every node acts as *attester* during its lifetime. As attester, each node selects the list of *attestees* against which it performs attestations. More specifically, each attester generates the attesstee list by selecting m distinct IDs uniformly at random from the entire ID space (i.e., n distinct IDs). Note that the attestee selection process is repeated every time slot and thus a node will highly likely have distinct attestee list for each time slot.

Suppose that a node u sends a packet to a node v where u and v are in the vicinity of each other. Upon receiving the packet sent by u, v first checks whether u belongs to the attestee list. If so, it works as primary-attester and performs software attestations against u. If the attester v checks whether the flash image of u has been subverted and decides the attestee u as infectious node, it sends *Attester Role Assignment* message to all neighboring nodes. Upon receiving this message, each neighbor w of v works as secondary-attester such that it performs the attestations against all neighboring nodes each time it receives packets from them. Note that the attester v will work as secondary-attester as well as primary-attester in order to expedite the infectious node detection.

In the sense that the infectious nodes are highly likely close to each other due to hop-by-hop worm propagation strategy, the attester role assignment contributes to fast detection of infectious nodes.

After detecting infectious neighbor nodes, every node stops communicating with them and thus the infectious nodes are isolated from the network.

4.2. Security Analysis. In this section, we derive the probability that an infectious node is detected by primary-attesters.

Recall that there are n distinct IDs in the network and a node maintains the list of m attestees against which it performs attestations. When a primary-attester receives a worm packet from an infectious node, the probability that it detects an infectious node is calculated as m/n. Therefore, when s primary-attesters receive a worm packet from an infectious node, the probability that they detect an infectious node is given by

$$\begin{aligned} P_s &= 1 - \prod_{i=1}^{s} \left(1 - \frac{m}{n}\right) \\ &= 1 - \left(1 - \frac{m}{n}\right)^s. \end{aligned} \tag{2}$$

By applying the fact that $(1 + x) \leq e^x$ to (2), we have

$$\begin{aligned} P_s &= 1 - \left(1 - \frac{m}{n}\right)^s \\ &\geq 1 - e^{-ms/n}. \end{aligned} \tag{3}$$

From (3), the lower bound on P_s is calculated as $1 - e^{-ms/n}$.

Figure 1 shows how the lower bound of infectious node detection probability is affected by the number of primary-attesters and the ratio of attestee list size to total number of nodes. When $m/n \geq 0.25$, it is guaranteed that an infectious node is detected with probability of more than 0.95. Furthermore, when the number of primary-attesters is at least 30, infectious detection probability is at least 0.95 even in case of $m/n = 0.1$. This demonstrates that our proposed scheme achieves high infectious node detection capability. As s increases, the lower bound on P_s tends to rise. This indicates that a growth in the number of primary-attesters contributes to an increase in the likelihood that infectious node is detected. Moreover, as s rises, we observe the narrower gap among the lower bounds on P_s in three cases of m/n. This implies that the growth rate in lower bound on P_s rises as m/n falls off.

4.3. Performance Analysis. In this section, we compute the attestation, communication, and storage overhead of our proposed scheme. For this purpose, we take account of *benign* and *worm* case. In benign case, we consider $K \geq 1$ distinct sender-receiver pairs such that each sender sends $\lambda \geq 1$ benign packets to each receiver. In worm case, we consider

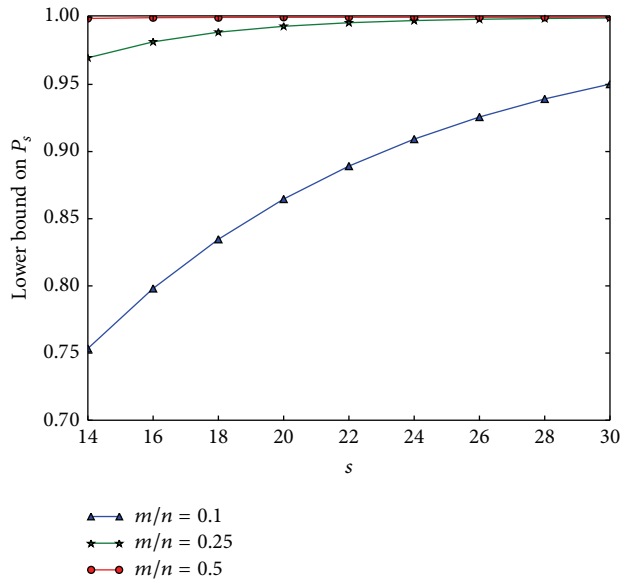

FIGURE 1: The effects of s and m/n on the lower bound of infectious node detection probability.

R distinct infectious-susceptible pairs such that an infectious node propagates λ worm packets to a susceptible node. We also assume that each node has d neighbors on average.

We define the attestation overhead as an average number of software attestations that are performed in the network. When there is no traffic in the network, the attestation overhead will be zero because each node initiates attestation process against a neighbor node only if it receives packets from that neighbor. In benign case, the attestation overhead is calculated as $R \times \lambda \times (m/n)$ since a receiver is a primary-attester and it attests against a sender with probability m/n whenever it recieves a benign packet. In worm case, the attestation overhead incurred by primary-attesters is $R \times \lambda \times (m/n)$ because R susceptible nodes are primary-attesters. Since all neighbors of primary-attesters act as secondary-attester and primary-attesters also take secondary-attester role, there are at most $R \times (d + 1)$ secondary-attesters in the network. Let us consider a worst-case scenario in which all secondary-attesters are susceptible and each secondary-attester receives $\lambda' \geq 1$ worm packets. The attestation overhead incurred by secondary-attesters is at most $R \times (d + 1) \times \lambda' \times (m/n)$ in the worse case. As a consequence, the total attestation overhead in the worst case is calculated as $R \times (m/n) \times ((d + 1)\lambda' + \lambda)$.

We define the communication overhead as an average number of *Attester Role Assignment* messages sent in the network. In benign case, communication overhead is zero because there are no infectious nodes and thus *Attester Role Assignment* messages are never sent by attesters. In worm case, after primary-attesters detect infectious nodes by attesting against them, they send *Attester Role Assignment* messages to their neighbors. Thus, the communication overhead is computed as $R \times (m/n) \times d$.

Finally, we define storage overhead as the number of IDs that need to be stored per node. Since each node stores m IDs that are randomly selected, this overhead is calculated as m in both benign and worm cases.

5. Simulation Study

In this section, we first explain the simulation environments and then describe the simulation results.

5.1. Simulation Environment. We developed a simple simulation program to evaluate our proposed scheme. In our simulation, we place 1000 sensor nodes in a square area field of 1000 m × 1000 m and configure the communication radius of a sensor node to 50 m. Under this network setting, we are able to evaluate how our proposed scheme detects worm propagation in large-scale network. Moreover, we employ a group deployment strategy in which a group of sensor nodes is placed toward the group deployment point and the actual placement follows the two-dimensional Gaussian distribution:

$$f(x, y) = \frac{1}{2\pi\sigma^2} e^{-((x-x_g)^2+(y-y_g)^2)/2\sigma^2}, \quad (4)$$

where (x_g, y_g) is the group deployment point and σ is the standard deviation. In our group deployment, the number of groups is set to 20, the number of nodes in a group is configured to 50, and σ is set to 50. We set $m = 100, 200$, where m is the size of the attestee list that a node needs to store.

We evaluate our scheme in two cases: *benign* and *worm*. In the benign case, we consider only benign traffic in the network. In the worm case, we consider only worm traffic in the network. The main rationale behind this case separation is to explore the worm spread defense capability and the attestation overhead without being affected by the benign packets and the worm packets, respectively.

In the benign case, we model the number of benign packets to be generated as a homogeneous Poisson process with rate parameter γ. As a consequence, the interarrival times of benign packets follow the exponential distribution. More specifically, the interarrival time between two consecutive benign packets is calculated as $-(\ln(U))/\gamma$, where U is uniform random variate such that $0 \leq U < 1$. We configure γ from 1.0 to 10.0 via increments of 1.0. Under these settings, the number of benign packets to be generated is from 3000 to 30000 via increase of 3000 on average. For each time slot, we randomly select as many pairs of source and destination nodes as the number of benign packets to be generated by Poisson process. Additionally, destination node is randomly chosen from the neighbors of source node. This is reasonable because local communication patterns are more prevalent than arbitrary peer-to-peer ones in sensor networks [13].

In the worm case, as described in Section 3, we adopt a hop-by-hop worm propagation strategy together with the discrete-time version of the simple epidemic model [14], in which the infection quota deficit in the current time slot is carried over to the next time slot, meeting the total infection quota over the entire time period. Recall that ϵ is the pairwise infection rate; I_0 indicates the number of worm originators. We consider a single worm originator and accordingly I_0 is set to one. We set ϵ from 0.0001 to 0.001 via increments of 0.0001. Furthermore, we assume that a single worm packet is used to infect a susceptible node. This assumption is regarded as the best-case scenario for the attacker because the fast

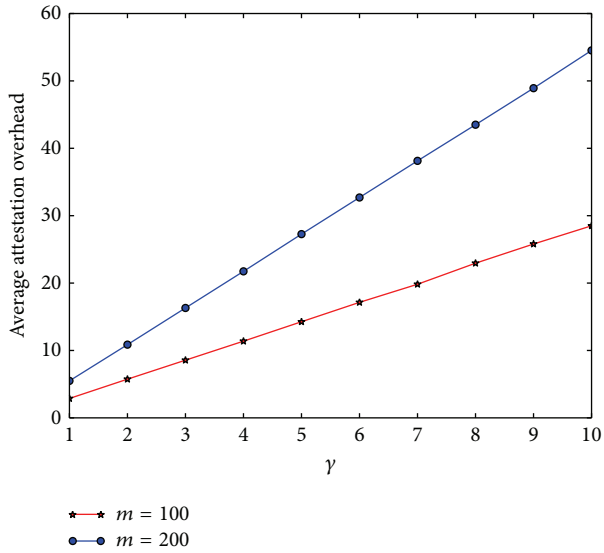

FIGURE 2: The effect of Poisson rate parameter γ on the attestation overhead in the benign case.

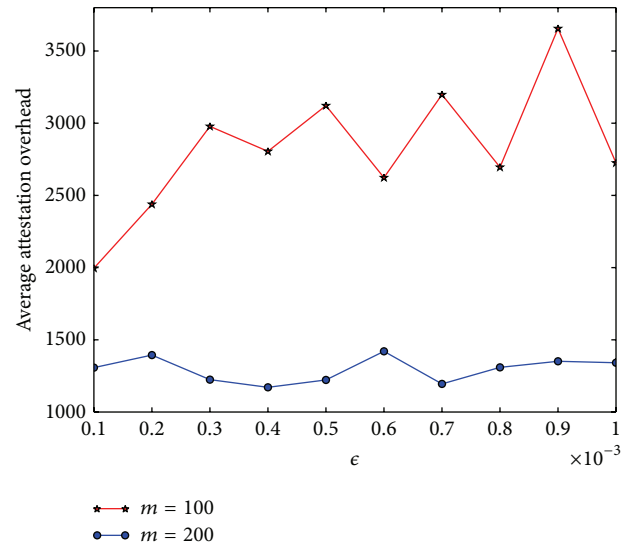

FIGURE 3: The effect of worm infection rate ϵ on the attestation overhead in the worm case.

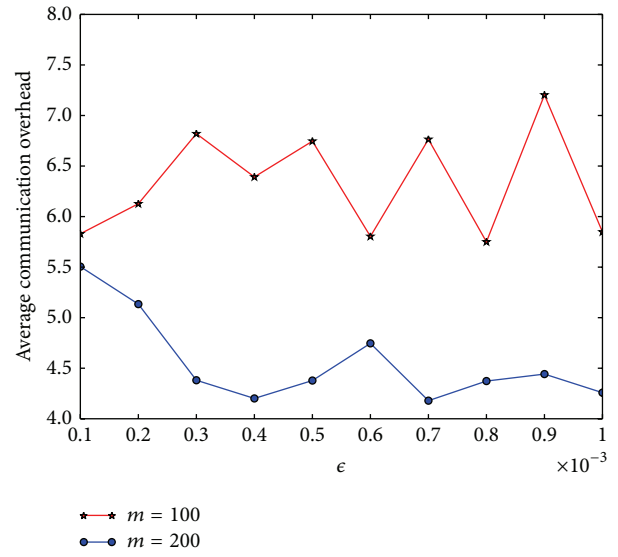

FIGURE 4: The effect of worm infection rate ϵ on the communication overhead in the worm case.

infection is beneficial for the attacker. Hence, we evaluate our scheme under the worst-case scenario in terms of the number of worm packets.

5.2. Simulation Results. We use the following metrics to evaluate the performance of our proposed scheme.

(i) *Fraction of Infectious Nodes* is the fraction of infectious nodes when all worm propagations are blocked in the network.

(ii) *Attestation Overhead* is the number of attestations that are performed per time slot in the network.

(iii) *Communication Overhead* is the number of *Attester Role Assignment* messages that are sent per time slot in the network.

For each metric, we present the average results for 100 runs of the simulation in each configuration such that each run is executed for 100 time slots, where a time slot duration is 30 simulation seconds. Our main findings are as follows.

In the benign case, as shown in Figure 2, all nodes perform at most 54.5 attestations per time slot on average. This means that a small number of attestations are performed in the network. Furthermore, attestation overhead rises as γ increases in the benign case. This is because the more benign traffic incurs the higher attestation overhead. Given a value of γ, a rise in m contributes to an increase in attestation overhead. This is because more attestations are performed as the size of attestee list grows. In particular, as γ rises, we observe the wider gap between the attestation overhead in case of $m = 100$ and the one in case of $m = 200$. This implies that the attestee list needs to be kept in small size in order to reduce the attestation overhead under the high benign traffic.

Figures 3 and 4 show how attestation and communication overhead are affected by ϵ and m in the worm case. In terms of attestation overhead, all nodes perform at most 3655.7 and 1420.4 attestations for each time slot on average when $m = 100$ and $m = 200$, respectively. This represents the fact that a node performs below four attestations per time slot on average, burdening insignificant overhead on each node. In terms of communication overhead, the number of *Attester Role Assignment* messages sent per time slot is at most 7.2 on average, incurring slight overhead in the network. Both attestation and communication overhead in case of $m = 200$ are less than the ones in case of $m = 100$. This indicates that the large size of attestee list contributes to diminishing the attestation and communication overhead in all cases of infection rates. Putting it in a different way, when more nodes are attested, worm propagation is blocked at the earlier time, leading to decay in the attestation and communication overhead. We also observe that the attestation overhead more severely fluctuates over ϵ in case of $m = 100$ than $m = 200$. On

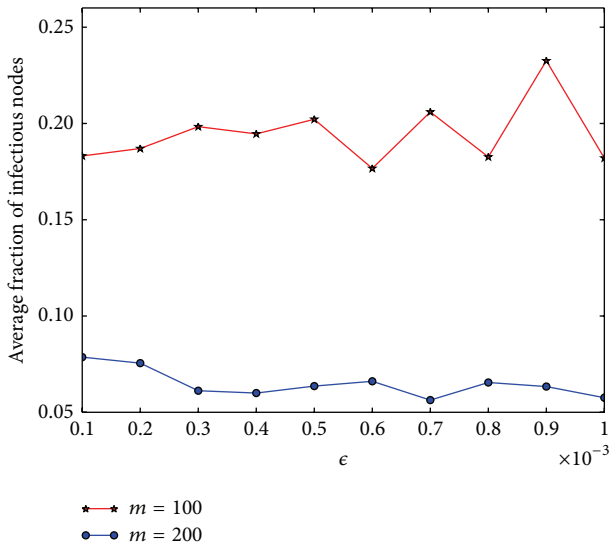

FIGURE 5: The fraction of infectious nodes versus worm infection rate ϵ in the worm case.

the other hand, in both cases, the communication overhead exhibits relatively small fluctuation over ϵ.

As shown in Figure 5, the fraction of infectious nodes is limited to at most 7.86% and 23.26% in case of $m = 200$ and $m = 100$, respectively. This means that our proposed scheme stops worm propagation while minimizing the number of infectious nodes. Moreover, the fraction of infectious nodes in case of $m = 200$ is substantially lower than the one in case of $m = 100$. This indicates that more attestations result in the lower fraction of infectious nodes.

6. Conclusions

In this paper, we proposed on-demand software-attestation based scheme to stop worm propagation in sensor network. We also analytically showed that our proposed scheme blocks sensor worm propagation in efficient and robust manner. Furthermore, we evaluated the proposed scheme through simulation. The simulation results demonstrate that our proposed scheme stops worm propagation with at most 3655.7 attestations per time slot on average while at most 54.5 attestations per time slot are required on average in benign scenario, where 1000 nodes are employed in the network. Moreover, the faction of infectious nodes is sustained to at most 7.86% and 23.26% when the size of the attestee list that is maintained by a node is 200 and 100, respectively.

Conflict of Interests

The author declares that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

This work was supported by a young researchers grant from Seoul Women's University (2014).

References

[1] N. Gui, E. Zhai, J. Hu, and Z. Chen, "SWORDS: improving sensor networks immunity under worm attacks," in *Web-Age Information Management: 11th International Conference, WAIM 2010, Jiuzhaigou, China, July 15–17, 2010. Proceedings*, vol. 6184 of *Lecture Notes in Computer Science*, pp. 86–96, Springer, Berlin, Germany, 2010.

[2] Y. Liu, W. Zhang, S. Bai, and C. Wang, "Defending sensor worm attack using software diversity approach," in *Proceedings of the IEEE International Conference on Communications (ICC '11)*, pp. 1–5, IEEE, June 2011.

[3] Y. Yang, S. Zhu, and G. Cao, "Improving sensor network immunity under worm attacks: a software diversity approach," in *Proceedings of the 9th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc '08)*, pp. 149–158, May 2008.

[4] T. Abuhmed, J. Kang, D. Nyang, and K. H. Lee, "A software-based group attestation for wireless sensor networks," *Ad-Hoc & Sensor Wireless Networks*, vol. 13, no. 1-2, pp. 121–154, 2011.

[5] T. Park and K. G. Shin, "Soft tamper-proofing via program integrity verification in wireless sensor networks," *IEEE Transactions on Mobile Computing*, vol. 4, no. 3, pp. 297–309, 2005.

[6] A. Seshadri, A. Perrig, L. van Doorn, and P. Khosla, "SWATT: softWare-based attestation for embedded devices," in *Proceedings of the IEEE Symposium on Security and Privacy (S&P '04)*, pp. 272–282, May 2004.

[7] Y. Yang, X. Wang, S. Zhu, and G. Cao, "Distributed software-based attestation for node compromise detection in sensor networks," in *Proceedings of the 26th IEEE International Symposium on Reliable Distributed Systems (SRDS '07)*, pp. 219–230, IEEE, Beijing, China, October 2007.

[8] D. Zhang and D. Liu, "DataGuard: dynamic data attestation in wireless sensor networks," in *Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks (DSN '10)*, pp. 261–270, July 2010.

[9] A. Francillon and C. Castelluccia, "Code injection attacks on harvard-architecture devices," in *Proceedings of the 15th ACM conference on Computer and Communications Security (CCS '08)*, pp. 15–25, October 2008.

[10] T. Goodspeed, "Exploiting wireless sensor networks over 802.15.4," in *Proceedings of the Texas Instruments Developper Conference*, November 2008.

[11] Q. Gu and R. Noorani, "Towards self-propagate mal-packets in sensor networks," in *Proceedings of the 1st ACM Conference on Wireless Network Security (WiSec '08)*, pp. 172–182, Alexandria, Va, USA, April 2008.

[12] B. Sun, G. Yan, Y. Xiao, and T. Andrew Yang, "Self-propagating mal-packets in wireless sensor networks: dynamics and defense implications," *Ad Hoc Networks*, vol. 7, no. 8, pp. 1489–1500, 2009.

[13] C. Karlof and D. Wagner, "Secure routing in wireless sensor networks: attacks and countermeasures," *Ad Hoc Networks*, vol. 1, no. 2-3, pp. 293–315, 2003.

[14] D. J. Daley and J. Gani, *Epidemic Modeling: An Introduction*, vol. 15 of *Cambridge Studies in Mathematical Biology*, Cambridge University Press, Cambridge, UK, 1999.