IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON SECURITY IN WIRELESS COMMUNICATIONS AND NETWORKING

Received November 17, 2016, accepted December 30, 2016, date of publication January 5, 2017, date of current version March 2, 2017.

Digital Object Identifier 10.1109/ACCESS.2017.2648853

Distributed Detection of Sensor Worms Using Sequential Analysis and Remote Software Attestations

JUN-WON HO[1] AND MATTHEW WRIGHT[2]
[1]Department of Information Security, Seoul Women's University, 01797 Seoul, South Korea
[2]Department of Computing Security, Rochester Institute of Technology, 14623-5603 Rochester, NY USA

Corresponding author: J.-W. Ho (jwho@swu.ac.kr)

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No. 2016R1C1B1014126).

ABSTRACT Recent work has demonstrated that self-propagating worms are a real threat to sensor networks. Since worms can enable an adversary to quickly compromise an entire sensor network, they must be detected and stopped as quickly as possible. To meet this need, we propose a worm propagation detection scheme for sensor networks. The proposed scheme applies a sequential analysis to detect worm propagation by leveraging the intuition that a worm's communication pattern is different from benign traffic. In particular, a worm in a sensor network requires a long sequence of packets propagating hop-by-hop to each new infected node in turn. We thus have detectors that observe communication patterns in the network, a worm spreading hop-by-hop will quickly create chains of connections that would not be seen in normal traffic. Once detector nodes identify the worm propagation pattern, they initiate remote software attestations to detect infected nodes. Through analysis and simulation, we demonstrate that the proposed scheme effectively and efficiently detects worm propagation. In particular, it blocks worm propagation while restricting the fraction of infected nodes to at most 13.5% with an overhead of at most 0.63 remote attestations per node per time slot.

INDEX TERMS Wireless sensor networks, sequential analysis, worm detection.

I. INTRODUCTION

In wireless sensor networks, an attacker can easily compromise sensor nodes by physically capturing them and then launch a variety of attacks by exploiting the compromised nodes. However, finding and physically compromising a large number of nodes would take time and effort and may put the attacker at risk of being discovered. A much better option for the attacker is to capture and compromise just a few nodes and infect these nodes with self-propagating malware, i.e. with a worm. By simply reintroducing infected nodes back into the network, the worm could quickly spread and take over a large number of sensors, giving the attacker control over much of the network's operations.

Until recently, however, there was reason to believe that existing wireless sensor networks were immune from such attacks. Sensor motes built on the Harvard architecture keep code and data separate, making software-based exploits like buffer overflows difficult to create. Francillon and Castelluccia [3] showed that this belief was wrong by demonstrating a remote node compromise attack in which malicious code is loaded permanently into Harvard-architecture sensor motes and then converted into a self-propagating worm.

This presents a major problem for sensor network security. As we have seen on the Internet, worms can spread very quickly and cause tremendous damage [26]. In a sensor network, worm propagation allows the attacker to perform wide-spread node compromise with just a few captured nodes. Hence, worm propagation attacks should be detected and stopped as quickly as possible to minimize the potential for damage in the network.

Although the prevention and detection of worm propagation attacks are essential to stop wide-spread node compromise in sensor networks, little research has been done to address this problem. The main work in preventing worm propagation is by Yang *et al.* [23], who propose a scheme based on a software diversity technique. Most sensor motes operate their main software program from flash memory, and we call this program the *flash program*. The main idea of their approach is to divide the network into a set of grid cells and assign a different version of the flash program to

each cell in such a way that two adjacent cells do not share the same version of the flash program. Although a worm can infect one cell by exploiting a vulnerability in the flash program assigned to that cell, it may fail to infect the adjacent cells due to differences in the software. This approach works under the assumption that each version of flash program has vulnerabilities that are distinct from all the others. This scheme is expanded to consider the case in which multiple versions of flash programs are installed in a sensor [24]. It may be possible, however, to find different vulnerabilities in two or more versions of the flash program and get a worm to spread into some or all of the network. Further, although there are ways to diversify software automatically [12], there are no guarantees that the vulnerabilities in one version of the software are different from those in another version. It is unclear that providing such a guarantee is any less difficult than automating the finding and eliminating of security vulnerabilities from software in general. Thus, different versions of the flash program may share a vulnerability such that only variations on the same exploit are required.

If worm propagation cannot be easily prevented, it may be detected. A particularly robust way to detect compromised nodes is with remote software attestation [1], [11], [15], [17], [22]. The main idea of software attestation is to prove the integrity of a node's programs on the basis of software or hardware. Since any node can attest against any other nodes and detect compromised nodes without using dedicated hardware such as TPMs [20], remote software attestation is a promising technology for discerning any infected nodes in a resource-constrained sensor network. Ho [8] proposed an *attestation-only* scheme in sensor network. This is our prior work whose main idea is to have each node randomly choose a set of nodes in each time slot and perform remote attestation against these nodes whenever it receives packets from them, thus detecting nodes infected by the worm. Although this attestation-only approach detects worm propagation with little attestation overhead, it could fail to detect infectious nodes that are not included in the randomly selected set, leading to a decline in detection capability.

To mitigate this limitation of the attestation-only approach, we propose a worm propagation detection scheme using the Sequential Probability Ratio Test (SPRT). We leverage the following intuition: a worm usually propagates in a hop-by-hop manner, creating a chain of worm connections. Hence, we are likely to observe this chain grow link by link when a worm is being propagated. On the other hand, chains of benign connections are unlikely to be seen because the dominant communication pattern is many-to-one, between multiple source nodes and a single data aggregator, rather than peer-to-peer communication patterns [9]. This is mainly because data aggregation is efficient in terms of sensory data delivery to the base station. We are thus unlikely to observe communication that looks like packet repropagation in the benign case. Using this intuition, we apply the SPRT with a null hypothesis that a worm is not being propagated and an alternate hypothesis that a worm is being propagated. In the SPRT, the lower (resp. upper) limit represents benign (resp. worm) activity, and these limits are dynamically configured in accordance with type of samples. If the number of samples of benign (resp. worm) activity hits or crosses the lower (resp. upper) limit, the null (resp. alternate) hypothesis is accepted. We combine the SPRT with the attestation-only approach in such a way that the SPRT detects the infectious nodes missed by the attestation-only approach while still detecting any infectious nodes that would be found through the attestation-only approach. Every benign node blocks the communications from the infectious nodes detected by the scheme.

The main benefit of the proposed scheme is that it detects worm propagation quickly. The scheme works against any type of worm, because it detects the fundamental characteristic of packet repropagation that every type of worm exhibits. Thus, even if the attacker creates and propagates zero-day and polymorphic worms that would be difficult for signature-based schemes to detect, our proposed scheme still detects them. Moreover, our proposed scheme will not falsely identify benign nodes as infected nodes. This is because our proposed scheme is rooted on remote software attestation that achieves virtually zero false positive.

We validate our proposed scheme through analysis and simulation. Specifically, we quantitatively show that our scheme greatly limits the attacker's gains from worm propagation. We also show analytically that our scheme requires few samples to make a decision, leading to fast detection and blocking of worm propagation. In simulation, we demonstrate that our scheme quickly detects and stops worm propagation with less than three samples on average in all cases while keeping the fraction of infected nodes to at most 13.5% in all cases. Furthermore, our scheme requires each node to perform at most 0.63 and 0.31 attestations per node per time slot in the case of no worm and the case of a worm, respectively. These simulation results signify that our scheme quickly detects worm and minimizes the damage incurred by worm at the cost of very few attestations. Our proposed approach leverages both software attestation and the SPRT, and we show that it achieves better worm detection capability than our prior attestation-only approach [8] with very little additional attestation overhead. In particular, our proposed scheme reduces the number of infected nodes between 16.9% and 43.6% compared with our prior work.

The rest of paper is organized as follows. §II presents related work. §III then describes the network assumptions and attacker models for our proposed scheme. In §IV, we describe our worm propagation detection scheme using the SPRT with a randomized sampling strategy and analyzes its security and performance. §V presents the simulation results for the proposed scheme, and §VI concludes the paper.

II. BACKGROUND

In this section, we first briefly describe why prior work in Internet worm detection and prevention cannot be applied to sensor networks. We then discuss work showing that worms

are a threat to networks of sensors with various system architectures, and finally we point out the limitations of related work in worm propagation prevention and detection for sensor networks.

A. INTERNET WORMS

The existing solutions [10], [13], [14], [16] designed for Internet worm detection and prevention cannot be directly applied for sensor networks. This is mainly because they rely on secure infrastructure that is not available in sensor networks. In particular, signature detection approaches [10], [14] require substantial computation overhead for signature generation and maintenance, which is not suitable for resource-constrained sensor networks.

Even though we are the first to apply the SPRT to detect worm in *wireless sensor networks*, Schechter *et al.* [16] adapted the SPRT to detect scanning worms in the Internet. This approach leverages the intuition that scanning worms are very likely to cause connection failures when they try to infect susceptible nodes. As this approach can only be used to detect worms that scan the network for targets, it is not appropriate for sensor networks, in which target nodes can simply be neighbor nodes. On the other hand, our proposed scheme can be applied to detect any types of worm because it leverages the fundamental characteristic of packet repropagation that every type of worm exhibits. Therefore, our scheme is a more general SPRT-centric approach than that of Schechter et al. in terms of sensor worm detection.

B. WORMS IN SENSOR NETWORKS

Several researchers have recently showed the feasibility of malicious code injection attacks and worm propagation attacks against sensor devices [3]–[5]. Specifically, Gu *et al.* [5] experimented with transient malicious code execution on Mica2 motes and explored the possibility of malicious code propagation. Goodspeed [4] proposed a way of running malicious codes on the MSP430-based TelosB motes. Francillon and Castelluccia [3] conducted experiments with permanent malicious code execution on a MicaZ mote that is one of Atmel AVR-based sensor devices. Moreover, they also demonstrated that the malicious code can be easily extended to a self-propagating worm. Thus, it is clear that an attacker could build a worm that can propagate even on sensor motes built upon the Harvard architecture.

C. WORM PREVENTION IN SENSOR NETWORKS

There are few works studying the prevention of worm propagation in sensor networks. Specifically, Yang *et al.* [23] proposed a software diversity technique to prevent worm propagation in sensor networks. In this work, the network is divided into a set of grid cells and a group of sensors are deployed in each cell. A flash program is then installed into each node in such a way that all nodes in the same cell have the same version of the program, but nodes in adjacent cells have different programs. Thus, even if an attacker creates a worm by exploiting the vulnerability of a version of flash program assigned to a cell and infects the entire cell with that worm, the worm will fail to infect the adjacent cells if the programs in those cells have different vulnerabilities. This technique is extended to deal with the worm prevention in case that multiple flash versions are installed in a sensor [24]. The main strength of this work is that it prevents a worm from propagating between two adjacent cells, and hence the rest of the network, with little overhead. However, it is hard to automatically make the different versions of a flash program possess different vulnerabilities. We do not need this scheme if we succeed in discovering different vulnerabilities, since we could then fix them. On the other hand, a worm could infect all sensor nodes by exploiting any unidentified common vulnerability shared by all versions of flash program. Furthermore, if different vulnerabilities can be found in two or more versions of the code, the worm code could be programmed with all the exploits. This worm can first infect one cell and then switch vulnerabilities to infect one of the neighboring cells, thereby spreading through much of the network.

Gui *et al.* [6] investigated the impact of using a software diversity technique on worm infection prevention through active sensor nodes when the active state periods of nodes are determined by random node scheduling. Liu *et al.* [25] also proposed a software diversity technique combined with using a role-based graph coloring in sensor networks. Since these schemes are based on the software diversity technique, they have the same restrictions as in [23] and [24]. Sun *et al.* [19] deterred or decelerated worm propagation by employing a set of immune nodes. However, this approach does not describe the specific details of how worm propagation is blocked by immune nodes while largely focusing on immune node selection mechanism.

Shen *et al.* [18] formulate differential-equations based game to model the dynamics between attack and defense in terms of malware propagation in sensor network. It then derive the optimal dynamic strategies for attack and defense. Although this work presents a game theoretical framework to restrain malware propagation in sensor networks, however, it does not provide actual prevention technique to stop worm propagation in sensor networks.

D. WORM DETECTION IN SENSOR NETWORKS

To the best of our knowledge, our previous work [8] is the first contribution for worm detection in sensor networks. The key idea of [8] is to have each node perform the remote attestation against a set of nodes that are randomly chosen in each time slot. The attestation of a node in the set only occurs when the attesting node receives packets in that time slot from the node. Although this approach efficiently detects sensor worms with little overhead, it could fail to detect infectious nodes that do not belong to the randomly chosen set. To overcome this limitation, we leverage the SPRT to improve the detection capability of our previous work [8] with additional little overhead. Note that the basic idea of our proposed scheme originates from the chapter 7 of the first

author's Ph.D. thesis [7]. However, we tremendously revise and improve the chapter 7 in Ph.D. thesis in terms of scheme, security analysis, and evaluation. The biggest distinct point is how the SPRT is incorporated into the remote software attestation technique. In chapter 7 of Ph.D. thesis, sensor worm detection mostly depends on the SPRT while getting little aid of remote software attestation, leading to failure in very slow worm detection. However, our proposed scheme utilizes the balanced cooperation of the SPRT and attestation, leading to the efficient and effective worm detection.

III. PRELIMINARIES

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

A. NETWORK ASSUMPTIONS

We assume a *static* sensor network, in which the locations of sensor nodes are not changed after being placed over the network field.

B. ATTACKER MODELS

We assume that the attacker sets up a set of compromised nodes to be worm originators and has them spread the worm over the network, thereby seeking to gain control of as much of the network as possible without being detected. Since an epidemic model provides a useful way to estimate the expected rate of infection and numbers of infected nodes at various points of time, it is widely used for worm propagation modeling in the Internet, and thus we adopt it to model worm propagation in sensor networks. Specifically, we make use of the discrete time version of the simple epidemic model [2], in which the overall rate of new infections is given by:

$$\frac{dI_t}{dt} = \rho I_t[N - I_t] \tag{1}$$

and the infection quota is determined in units of time slots according to the following equation:

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

where N is the total number of sensor nodes in the network, and ρ is the pairwise infection rate [2]. I_0 indicates the number of worm originators. I_t is the cumulative infection quota from the 0th time slot to the tth time slot. Accordingly, the infection quota in the tth time slot is calculated as $I_t - I_{t-1}$.

In this model, we assume that a sensor has only two states: *susceptible* and *infectious*. All sensor nodes are initially in the susceptible state except worm originators that are in the infectious state. Once a susceptible node is infected by worm, its state is changed to infectious.

We also assume an intelligent worm attack in which infectious nodes do not try to infect benign nodes that have already performed attestation against the worm and found an infection. This assumption is reasonable in the sense that worm infection can be progressed quickly without wasting time in multiple infection attempts on these benign nodes. Note that this assumption is not used in our prior work [8], where the worm instead tries to infect any benign nodes.

We must not only model the rate of propagation, but also how the worm propagates through the network. We assume that the attacker employs a hop-by-hop worm propagation strategy in which each infectious node propagates the worm to its neighboring susceptible nodes. Sensor networks generally use localized protocols for clustering, data aggregation, and other activities [9], and they do not have much scope for arbitrary multi-hop peer-to-peer communication. If the worm is not propagated in a neighbor-to-neighbor fashion, and t does not exhibit the common pattern of normal local traffic, it would be easy to detect the worm propagation. To reduce the chance of being detected, the attacker will thus rely on a hop-by-hop propagation strategy that makes worm propagation look like the normal local traffic.

When this hop-by-hop propagation strategy is used together with the epidemic model, however, it could fail to meet the worm infection quota in a given time slot, because the infectious nodes could have few susceptible neighbors at that time. In this case, we assume that the infectious nodes maintain the infection rate by choosing susceptible nodes at random from the network and propagating the worm to them. Since sensor networks typically have dense node deployments to ensure networking and sensing coverage, each node will likely have sufficient neighbors to make the hop-by-hop propagation strategy dominate over random propagation, keeping the chance of detection low.

IV. WORM DETECTION USING THE RANDOMIZED-SPRT

In this section, we describe and analyze our approach to detecting worms in sensor networks. First, we discuss the use of attestation in detecting infected nodes in sensor networks, including why an approach based on attestation alone is insufficient for worm detection. We then present how we can detect worms effectively by combining a statistical decision mechanism together with attestation, leading to the details of our proposed scheme. Finally, we analyze the proposed scheme's security and performance.

A. ATTESTATION

To detect infected nodes, we could rely on remote software attestation techniques [1], [11], [15], [17], [22]. The key idea of these techniques is to validate the integrity of another node's programs on the basis of software or hardware. Since any node can attest against any other nodes without the aid of specialized hardware such as TPMs [20], remote software attestation is a promising technology for identifying any infected nodes in a sensor network. Moreover, this technique fulfills virtually zero false positives, indicating that benign nodes will not be detected as the infectious nodes.

In our prior work [8], we proposed an attestation-only approach that utilizes remote software attestation to detect worm propagation in sensor networks. Specifically, each node randomly selects a set of attestees per time slot. Whenever each node receives data packets from the nodes

belonging to a randomly chosen attestee set, it performs attestations against these nodes and detects any nodes infected by the worm. Since attestation is performed randomly, this approach detects worm propagation with little attestation overhead. However, it could miss the worm propagation incurred by nodes that are not included in the randomly chosen attestee set, contributing to decay in the detection capability.

B. INTEGRATING THE SPRT WITH ATTESTATION

To mitigate this limitation of our prior work, we first propose a worm detection scheme based on the Sequential Probability Ratio Test (SPRT), which is a statistical decision process that makes fast and accurate decision with a small number of samples. We then integrate the SPRT-based scheme with attestation such that attestation-only is first applied and then the SPRT is adapted to detect the worm propagation missed by attestation-only. The main merit of this integration is in achieving efficient and effective worm propagation detection while keeping the attestation overhead low. In other words, each node independently detects worm propagation through attestation while a group of nodes in a region collaborate to catch worm propagation through the SPRT, creating a robust worm detection capability with low attestation overhead.

The specific details of our proposed scheme are presented in the following section.

C. PROTOCOL DESCRIPTION

We first introduce the basic intuition and concept of the SPRT-based worm detection scheme. We then describe three phases in the proposed scheme that integrates the SPRT-based scheme with attestation-only approach.

For worm propagation detection in sensor networks, we leverage the simple intuition that the propagation pattern of worms is different from the patterns of benign packets. In particular, let us consider a worm scenario in which an infectious node A sends a worm to a susceptible node B and infects B with it. After being infected by A, an infectious B will repropagate the worm to a susceptible node C to infect it. Hence, if we sample the traffic being sent between the nodes in this region, we will observe a packet sent from A to B followed by a packet sent from B to C. We call this communication pattern a *chain*.

The main difference between chain and multi-hop communication is as follows: Each hop or link in a chain represents a message sent from the source to the destination, whereas multi-hop communication occurs between one source and one destination with multiple intermediate nodes as the hops. As sh in Figure 1, there are two pairs of source and destination in a $A - B - C$ chain. On the other hand, there is only one pair of source and destination in a $A - C$ multi-hop communication.

In a local region, multiple data source nodes usually send their sensor readings to a single data aggregator while rotating the role of data aggregator between them. Due to its efficiency, this many-to-one approach is the dominant

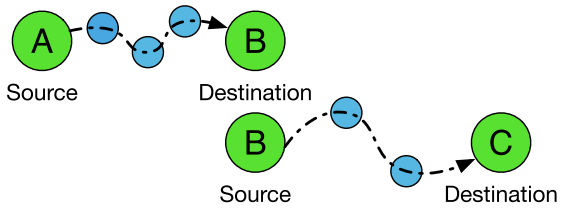

FIGURE 1. *A* – *B* multi-hop communication is created by packet sent from from *A* to *B* via intermediate nodes, whereas the *A* – *B* – *C* chain is created by a packet sent from *A* to *B* followed by packet sent from *B* to *C*.

communication pattern in a local region of a sensor network [9]. In the entire network, multiple aggregators usually form a hierarchical structure in such a way that the aggregated data is transmitted from lower-level aggregators up to higher-level ones in a tree structure, eventually reaching the base station. Note that higher-level aggregation also forms a many-to-one communication pattern.

We can see quantitatively the efficiency gain of many-to-one data aggregation. Let us assume that there are g sensor nodes in a region and let us denote the average hop distance between two randomly chosen nodes by Ω. In the case that individual nodes directly send data to the receiver, the average number of hops for data reporting would be $O(g \times \Omega)$. In the case of data aggregation, the average number of hops would be $O(g + \Omega)$, which is much lower communication overhead.

In a benign many-to-one communication scenario, short chains may occur because the normal sensor data usually traverses a few aggregators from a source to the base station. As a result, the likelihood of the occurrence of a chain in a benign scenario will be less than that of a chain when a worm is being propagated.

Using this intuition, we apply the Sequential Probability Ratio Test (SPRT) [21] to the worm propagation detection problem. Specifically, we define an alternate (resp. null) hypothesis that the worm is (resp. not) being propagated. The SPRT starts in the middle between the alternate hypothesis and null hypothesis and accepts the alternate (resp. null) hypothesis if the number of times that packet repropagation occurs hits or crosses the upper (resp. lower) threshold.

Our proposed scheme consists of three phases: a *Setup Phase*, an *Attestation-only Phase*, and a *SPRT-Attestation Phase*. In the setup phase, the network operator and each sensor node set up the necessary things for running the other two phases. In the attestation-only phase, each node performs remote attestations against randomly chosen attestees and detects worm propagation incurred by infectious attestees. If there are nodes that were not examined in the attestation-only phase and there are a certain number of compromised nodes in a region, the SPRT-attestation phase is initiated. In this phase, the SPRT is performed against the nodes unchecked in attestation-only phase. If the SPRT accepts the alternate hypothesis, infectious nodes are then detected through further remote attestations.

1) SETUP PHASE

a: PRE-DEPLOYMENT

In the pre-deployment stage, the network operator assigns every sensor node a unique ID and secret key material for pairwise key establishment. A hop-by-hop authentication mechanism is used to check whether a packet is sent by nodes with legitimate key material. More specifically, when a node transmits a packet to its neighbors, it authenticates the packet using a message authentication code (MAC), which is generated with a secret key shared between them. Therefore, upon receiving a worm packet, the susceptible node first checks its MAC and accepts it only if it is authentic. This means that worm packets should originate from the infectious nodes having legitimate secret key material.

b: DETECTOR ELECTION

After deployment, every sensor node discovers its neighbors and then elects itself as a worm propagation detector with probability p_d, which is configured in the pre-deployment stage. A worm propagation detector also acts as an attester. Before processing the received packets, an attester performs remote attestation against the nodes that send packets to the [illegible]ter. If an attester decides through remote attestation that the packet sender has been compromised, it will not communicate with it further. Therefore, the worm propagation detector is able to attest and detect any infectious nodes before processing worm packets and thus is virtually immune to worm infection. Each node repeats this detector selection process periodically. Hence, each node will likely function as a detector in rotation. This random rotation process will prevent the detectors from being exposed to attacker and also helps nodes save the energy consumption incurred by performing the detector function.

2) ATTESTATION-ONLY PHASE

Before the start of each time slot, each node i acts as an attester and chooses a set of attestees A, $|A| = m$, uniformly at random from the entire node space. If attester i receives a packet from node $j \in A$, it performs remote attestation against j. If j is determined to be infectious, i blocks all communication from j. Then, node i sends an attester notification message to its neighboring nodes. After receiving this message, every neighboring node of i performs remote attestations against any nodes that send packets to itself before processing these packets. If it decides that the packet sender is infected by worm, it blocks all communications from the infected node. Attesters will also block all communications from any infectious nodes denying attestation process while sending packets to attesters. If benign nodes could not participate in attestation process due to unreliable network connection, they would also be unable to communicate with attesters, thereby being discerned from malicious refusal to attestation process.

Attester notification messages are authenticated using MAC with a secret key shared between i and i's neighboring node. This authentication will be used to prevent DoS attacks incurred by malicious nodes. Specifically, if a malicious node sends a large number of fake attester notification messages to the nodes in a benign region, these nodes receiving the messages will perform many attestations against each other but will not find any malicious nodes, incurring lots of attestation overhead. However, if each node in benign region could process attester notification messages up to a preset threshold, discern the fake message senders based on their MACs, and block all further communications from them, the attacker will take little benefit from this type of DoS attack.

If node i receives a packet from node k that does not belong to the attestee list A, it first checks whether the fraction of compromised nodes among its neighbors is greater than or equal to a preset threshold τ. If so, node i initiates the SPRT-Attestation Phase. The main rationale behind this check is to prevent the SPRT from being run in a benign situation where all nodes are benign in the network. Thus, the overhead incurred by the SPRT will be zero in a benign situation.

3) SPRT-ATTESTATION PHASE

a: COMMUNICATION PATTERN BROADCAST

Once in the SPRT-Attestation phase, each time node u receives a packet from node v, u first performs packet preprocessing as follows. Node u checks whether v and u are the source and destination of the packet, respectively. If so, node u immediately broadcasts the source and destination IDs of the received packet to its neighbors. We call a pair of the source and destination IDs a *communication pattern*, or simply a *pattern*. Upon receiving the pattern, u's neighbor accepts it if u acts as a worm propagation detector. Otherwise, it discards the pattern. The pseudocode for a Packet Preprocessing Unit (PPU) is described in Algorithm 1.

Algorithm 1 Packet Preprocessing Unit (PPU)

INPUT: incoming packet *pkt*

if *pkt*.destination == u and *pkt*.source == u's neighbor **then**
  broadcast <*pkt*.*sourceID*, *pkt*.*destinationID*> to neighbors
end if

As previously described, many-to-one communications are the dominant pattern observed in both the local region and globally. As a result, the communication patterns are not frequently broadcasted, and the pattern broadcast overhead will be reasonable. In Section IV-E, we analytically show that the communication pattern overhead is not substantial.

Since a worm usually consists of multiple packets [3], it takes a certain amount of time to infect a node. As a result, node u is able to broadcast a communication pattern of worm packets before being infected. Moreover, the base station does not generate and broadcast the communication pattern, because it is the final destination of the data sent by sensor nodes and thus never repropagates the packets.

Furthermore, communication patterns are transmitted in plaintext. Thus, compromised nodes can generate and send fake patterns with the IDs of their neighbors that contribute to occurrence of merged patterns, resulting in false positives. However, once worm propagation is detected, the compromised nodes will be attested by their neighbors, detected and isolated from the network. Therefore, this type of attack will lead to the detection of the compromised nodes and is not useful for the attacker.

Every detector w divides the time domain into a series of time slots and performs worm propagation detection upon receiving the patterns from its neighbors. Note that each detector w independently maintains its time domain and thus there is no need to operate time synchronization process in the network. More specifically, w records the IDs of all of its neighbors. Each time w receives a pattern (s_i, d_i) such that $s_i \neq d_i$ in a given time slot, w stores s_i in its memory in such a way that s_i is linked to d_i. It then creates a *merged pattern* (s_j, s_i, d_i) if it already has a pattern (s_j, d_j) such that s_i $_j$ and $s_j \neq d_i$. If w creates multiple merged patterns per (s_i, d_i) such as $(s_{j1}, s_i, d_i), \ldots, (s_{jk}, s_i, d_i)$, it accepts only one merged pattern (s_{j1}, s_i, d_i) and discards other ones. This is mainly because one merged pattern is enough to fully capture a packet repropagation performed by s_i. Hence, at most one merged pattern is created per incoming pattern. We use the merged pattern as evidence for the occurrence of packet repropagation. As a result, as we observe more merged patterns, there will be a higher likelihood of worm propagation. Detector w refreshes the pattern information by erasing them at the end of each time slot. The main rationale behind this refresh is to reduce the likelihood that a benign pattern creates a merged pattern and accordingly minimizes the likelihood that benign packet transmission is misclassified as worm propagation. For efficient pattern processing, detector w does not deal with the same communication pattern more than once in a time slot, even if it receives multiple same communication patterns.

b: WORM DETECTION BASED ON THE RANDOMIZED-SPRT

We first describe how the Sequential Probability Ratio Test (SPRT) is used to detect a worm attack and then point out the limitation of the SPRT against an intelligent worm attack. Finally, we present a Randomized-SPRT scheme in order to enhance the security resilience of the SPRT-based detection.

The SPRT is a statistical decision process in which it takes a sample and it moves toward the upper (resp. lower) limit in accordance with the type of sample, leading to the acceptance of the alternate (resp. null) hypothesis [21]. We define H_1 (resp. H_0) as the alternate (resp. null) hypothesis that a worm is (resp. is not) being propagated.

More specifically, let $C_k (k \geq 1)$ denote the kth pattern received and stored by detector w. We define a Bernoulli random variable A_k as follows: If C_k creates a merged pattern, we set $A_k = 1$. Otherwise, we set $A_k = 0$. We define the success probability γ of the Bernoulli distribution as $\gamma = \Pr(A_k = 1) = 1 - \Pr(A_k = 0)$. Since H_0 and H_1 are mutually disjoint in the entire sample space, we have:

$$\Pr(A_k) = \Pr(A_k|H_0) \times \Pr(H_0) + \Pr(A_k|H_1) \times \Pr(H_1) \quad (3)$$

By only considering $\Pr(A_k = 1)$, we have

$$\begin{aligned} \Pr(A_k = 1|H_0) &= \frac{\Pr(A_k = 1) - \Pr(A_k = 1|H_1) \times \Pr(H_1)}{\Pr(H_0)} \\ \Pr(A_k = 1|H_1) &= \frac{\Pr(A_k = 1) - \Pr(A_k = 1|H_0) \times \Pr(H_0)}{\Pr(H_1)} \end{aligned} \quad (4)$$

$\Pr(A_k = 1)$ indicates the probability that merged pattern occurs. From Equation 4, we have the intuitive interpretation that $\Pr(A_k = 1|H_0)$ increases as $\Pr(A_k = 1|H_1) \times \Pr(H_1)$ decreases. Also, the lower $\Pr(A_k = 1|H_0) \times \Pr(H_0)$ is, the higher $\Pr(A_k = 1|H_1)$ is.

Given predefined thresholds γ_0 and γ_1 such that $\gamma_0 < \gamma_1$, if the detector finds that $\gamma \leq \gamma_0$, it is likely that a worm is not being propagated. On the other hand, if $\gamma \geq \gamma_1$, it is likely that a worm is being propagated. The problem of deciding whether a worm is likely being propagated or not can be formulated as a hypothesis testing problem with null and alternate hypotheses of $\gamma \leq \gamma_0$ and $\gamma \geq \gamma_1$, respectively.

Based on this problem formulation, we describe how detector w runs the SPRT to make a decision about worm propagation from the n observed samples, where A_k is treated as a sample. The log-probability ratio on n samples R_n is given as:

$$R_n = \ln \frac{\Pr(A_1, \ldots, A_n|H_1)}{\Pr(A_1, \ldots, A_n|H_0)}$$

We assume that A_k is independent and identically distributed. The main rational behind this i.i.d. assumption on A_k is as follows: In the case of benign traffic, A_k will likely be independent because the merged patterns could be generated by benign packet transmission. In the case of a worm, the question is whether there is independence of A_k's given H_1. In other words, although the probability that $A_k = 1$ is higher for all k when H_1 is true, we only require independence between samples given that fixed probability. Still, there may be correlations due to the worm activity. Correlation between consecutive samples, however, makes detection more likely. For example, a series of 1's will quickly drive the detector to accept alternate hypothesis H_1. It becomes an interesting question as to whether a slow worm can evade detection by careful spacing of activity (creating $A_k = 0$ several times in a row). He can know that the SPRT has either accepted H_0 or moved below the halfway point with high probability before he propagates the worm to create the next linked pattern. The challenge for the attacker is that multiple detectors are watching multiple nodes, so perfect coordination with no detection at any point will make the worm extremely slow.

Then R_n can be rewritten as:

$$R_n = \ln \frac{\prod_{k=1}^{n} \Pr(A_k|H_1)}{\prod_{k=1}^{n} \Pr(A_k|H_0)} = \sum_{k=1}^{n} \ln \frac{\Pr(A_k|H_1)}{\Pr(A_k|H_0)} \quad (5)$$

Let φ_n denote the number of times that $A_k = 1$ in the n samples. Then we have

$$R_n = \varphi_n \ln \frac{\gamma_1}{\gamma_0} + (n - \varphi_n) \ln \frac{1-\gamma_1}{1-\gamma_0} \tag{6}$$

where $\gamma_0 = \Pr(A_k = 1|H_0)$, $\gamma_1 = \Pr(A_k = 1|H_1)$.

On the basis of the log-probability ratio R_n, the SPRT for H_0 against H_1 is given as follows:

- $\varphi_n \leq l_0(n)$: accept H_0 and terminate the test.
- $\varphi_n \geq l_1(n)$: accept H_1 and terminate the test
- $l_0(n) < \varphi_n < l_1(n)$: continue the test with another observation.

Where α' is a user-configured false positive rate and β' is a user-configured false negative rate.

$$l_0(n) = \frac{\ln \frac{\beta'}{1-\alpha'} + n \ln \frac{1-\gamma_0}{1-\gamma_1}}{\ln \frac{\gamma_1}{\gamma_0} - \ln \frac{1-\gamma_1}{1-\gamma_0}}, \quad l_1(n) = \frac{\ln \frac{1-\beta'}{\alpha'} + n \ln \frac{1-\gamma_0}{1-\gamma_1}}{\ln \frac{\gamma_1}{\gamma_0} - \ln \frac{1-\gamma_1}{1-\gamma_0}}$$

Although the SPRT-based approach quickly and accurately detects worm attacks, it will fail to detect intelligent worm attacks in which a worm could dynamically change its infection rate. This is because, if the SPRT-based approach uses a fixed ϵ to a low value for quick worm detection, an intelligent worm will adaptively change its infection rate to a low value, leading to a failure to detect the worm. Putting it a different way, the SPRT-based approach is vulnerable to a slow worm attack in which the worm slowly infects benign sensor nodes and thereby reduces the likelihood of the occurrence of merged communication patterns, making worm look like benign traffic under a low ϵ value.

To mitigate this limitation of the SPRT-based approach, we propose a randomized sampling approach in which the SPRT randomly takes the samples of type H_0 with less weight than the ones of type H_1, where ϵ acts as a weighting factor. More specifically, we consider a round required for the decision process to reach a decision. For each round, ϵ is selected uniformly at random from the range of $[1, \epsilon_{max})$ such that $\epsilon_{max} > 1$. We call this approach of using SPRT with randomized sampling *Randomized-SPRT*. Because Randomized-SPRT randomly selects samples of type H_0 less frequently than the ones of H_1, the decision for acceptance of H_1 will be expedited. Accordingly, the Randomized-SPRT acts as an effective and efficient countermeasure against intelligent worm with dynamic infection rates and slow worms.

For each round, we construct the SPRT with randomized sampling by substituting $(\gamma_0)^\epsilon$ for γ_0 in the SPRT. In the case that $\epsilon > 1$, $\gamma_0 > (\gamma_0)^\epsilon$ holds, and thus the samples with type H_0 are chosen with less weight than the ones with type H_1, leading to a speed-up in the acceptance of H_1.

In randomized sampling, the log-probability on n samples L_n is changed to:

$$L_n = \varphi_n \ln \frac{\gamma_1}{(\gamma_0)^\epsilon} + (n - \varphi_n) \ln \frac{1-\gamma_1}{1-(\gamma_0)^\epsilon}$$

Accordingly, the SPRT for H_0 against H_1 is changed to:

- $\varphi_n \leq l_0(n)$: accept H_0 and terminate the test.
- $\varphi_n \geq l_1(n)$: accept H_1 and terminate the test
- $l_0(n) < \varphi_n < l_1(n)$: continue the test process with another observation.

Where:

$$l_0(n) = \frac{\ln \frac{\beta'}{1-\alpha'} + n \ln \frac{1-(\gamma_0)^\epsilon}{1-\gamma_1}}{\ln \frac{\gamma_1}{(\gamma_0)^\epsilon} - \ln \frac{1-\gamma_1}{1-(\gamma_0)^\epsilon}},$$

$$l_1(n) = \frac{\ln \frac{1-\beta'}{\alpha'} + n \ln \frac{1-(\gamma_0)^\epsilon}{1-\gamma_1}}{\ln \frac{\gamma_1}{(\gamma_0)^\epsilon} - \ln \frac{1-\gamma_1}{1-(\gamma_0)^\epsilon}}$$

c: STOPPING THE WORM

If the randomized-SPRT terminates in the acceptance of H_1 (resp. H_0), detector w decides that the worm is likely (resp. is not) being propagated within its vicinity. When H_0 is accepted, w restarts the randomized-SPRT. When H_1 is accepted, w performs the remote attestations against its neighboring nodes to detect the nodes infected by the worm. Then, as with Attestation-Only, w stops receiving communications from the infectious nodes and sends attester notifications messages to its neighbors. The neighbors similarly attest the nodes that send packets to themselves before processing the packets and then block their communications from infectious nodes. As in Attestation-Only phase, attester notification message is authenticated using MAC with a secret key shared between w and w's neighbor to prevent DoS attacks incurred by malicious nodes. Any infectious nodes maliciously declining the attestation process are isolated from the network as in Attestation-Only phase.

D. SECURITY ANALYSIS

In this section, we describe the resilience of the Randomized-SPRT scheme against worm infections. Recall that our proposed scheme is based on remote software attestation fulfilling virtually zero false positives. Thus, we do not present any analysis on false positives.

In the following Lemma, we investigate how many susceptible nodes are infected when a detector detects worm propagation in its vicinity.

Lemma 1: Let ζ denote the fraction of merged communication patterns incurred by worm propagation out of n communication patterns that detector w receives in its vicinity and processes. Given n communication patterns, w detects worm propagation in its vicinity if at least $\zeta^* \times n$ nodes in its vicinity are infected such that

$$\zeta^* = \frac{\ln \frac{1-\beta'}{\alpha'} + n \ln \frac{1-(\gamma_0)^\epsilon}{1-\gamma_1}}{(\ln \frac{\gamma_1}{(\gamma_0)^\epsilon} - \ln \frac{1-\gamma_1}{1-(\gamma_0)^\epsilon})n}$$

Proof: Since φ_n is the number of times that the merged communication pattern occurs in n communication patterns, $\varphi_n = \zeta \times n$ holds. According to the Randomized-SPRT, w detects worm propagation in its vicinity if $\varphi_n \geq l_1(n)$, which is rewritten as $\zeta \geq \zeta^*$ such that $\zeta^* = \frac{\ln \frac{1-\beta'}{\alpha'} + n \ln \frac{1-(\gamma_0)^\epsilon}{1-\gamma_1}}{(\ln \frac{\gamma_1}{(\gamma_0)^\epsilon} - \ln \frac{1-\gamma_1}{1-(\gamma_0)^\epsilon})n}$. Note that one merged communication pattern indicates that one

susceptible node is infected. For simplicity, we assume that the infected nodes do not try to reinfect other infected nodes. Hence, $\zeta \times n$ merged communication patterns correspond to $\zeta \times n$ infected nodes. Accordingly, if the number of infected nodes is at least $\zeta^* \times n$, worm propagation in the vicinity of w is detected. □

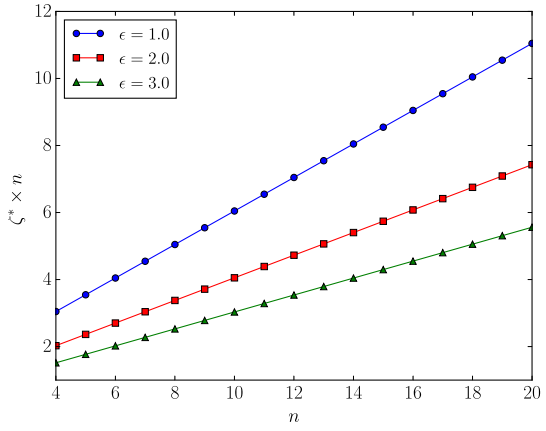

FIGURE 2. The affects of ϵ and n on $\zeta^* \times n$, which is the minimum number of infected nodes required for detector w to detect worm propagation in its vicinity.

Using Lemma 1, we now investigate how changing values of ϵ and n impact $\zeta^* \times n$. For this study, we set $\alpha' = \beta' = 0.01$, $\gamma_0 = 0.1$, $\gamma_1 = 0.9$. As shown in Figure 2, we see that $\zeta^* \times n$ increases in proportion to n. This implies that a detector can significantly restrain the number of infected nodes as long as it detects worm propagation with a small number of communication patterns. For example, a detector is able to limit the number of infected nodes to no more than eight if it detects worm propagation with at most ten samples. Given a value of n, we observe that $\zeta^* \times n$ decreases as ϵ increases. This means that Randomized-SPRT with larger values of ϵ achieves better resilience against worm infections.

Next, we investigate the detection capability of the Randomized-SPRT in terms of time measured in number of rounds. Recall that we consider a round required for the Randomized-SPRT to reach a decision and ϵ is selected per round uniformly at random from the range of $[1, \epsilon_{max})$ such that $\epsilon_{max} > 1$. Since ϵ is randomly determined every round, it is difficult for the attacker to predict ϵ in each round. Hence, the attacker is not likely to attempt to adjust the w infection rate. The following lemma further explains the resilience of the Randomized-SPRT.

Lemma 2: Suppose that the values of ϵ in the Randomized-SPRT are $\epsilon_1, \epsilon_2, \ldots, \epsilon_{w-1}, \epsilon_w$ in w rounds. Moreover, assume that the Randomized-SPRT with ϵ_i $(1 \le i \le w)$ is terminated in processing n communication patterns. Then the lower bound on probability P_w that worm propagation is detected before the $(w+1)$th round is $1 - e^{-\sum_{i=1}^{w} P_s^i}$, where

$$P_s^i = 1 - \frac{\ln \frac{1-\beta'}{\alpha'} + n \ln \frac{1-(\gamma_0)^{\epsilon_i}}{1-\gamma_1}}{n(\ln \frac{\gamma_1}{(\gamma_0)^{\epsilon_i}} - \ln \frac{1-\gamma_1}{1-(\gamma_0)^{\epsilon_i}})}$$

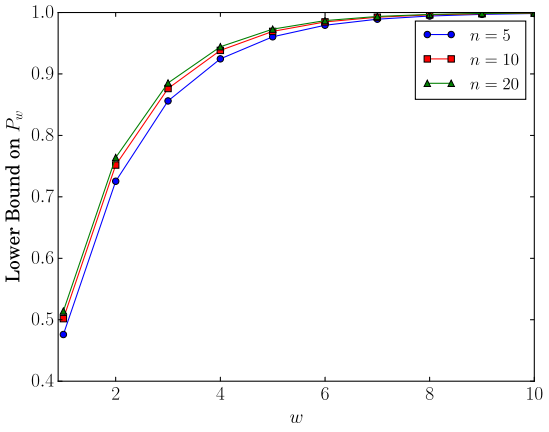

FIGURE 3. Lower bound on P_w vs. w when $\epsilon_{max} = 3.0$.

Proof: For each round i, the Randomized-SPRT with ϵ_i terminates in acceptance of H_1 if $\varphi_n \ge l_1(n)$ holds, leading to worm propagation detection. Thus, the worm propagation detection probability in each round i is calculated as

$$P_s^i = \frac{n - l_1(n)}{n} = 1 - \frac{\ln \frac{1-\beta'}{\alpha'} + n \ln \frac{1-(\gamma_0)^{\epsilon_i}}{1-\gamma_1}}{n(\ln \frac{\gamma_1}{(\gamma_0)^{\epsilon_i}} - \ln \frac{1-\gamma_1}{1-(\gamma_0)^{\epsilon_i}})}$$

If worm propagation detection succeeds in at least one round before the $(w+1)$th round, worm propagation will be detected before the $(w+1)$th round. Therefore, P_w is computed as $1 - \prod_{i=1}^{w}(1 - P_s^i)$. By using the property $(1+x) \le e^x$, we have

$$P_w = 1 - \prod_{i=1}^{w}(1 - P_s^i) \ge 1 - e^{-\sum_{i=1}^{w} P_s^i}$$

□

Using Lemma 2, we explore how the lower bound on P_w is affected by n and w. For this study, we consider a scenario in which the system has the same value of ϵ in all w rounds such that $\epsilon_{max} = \epsilon_1 = \epsilon_2 = \cdots = \epsilon_{n-1} = \epsilon_n$, leading to $P_s^1 = P_s^2 = \cdots = P_s^{w-1} = P_s^w$. This scenario indicates that the worm is propagated with the same γ in all w rounds. The reason why we use this scenario is because it helps to evaluate the detection capability of the system with ϵ_{max}. As shown in Figure 3, when $\epsilon_{max} = 3.0$, we see that an increase in both the number of rounds and the number of communication patterns contributes to a rise in the lower bound on P_w. With any number of communication patterns, however, only a few rounds are required to raise P_w above 0.8. In less than 10 rounds, detection of the worm is virtually assured.

E. PERFORMANCE ANALYSIS

In this section, we first compute how many samples are required on average for the Randomized-SPRT to reach a decision. We then present the communication and storage overheads of our proposed scheme.

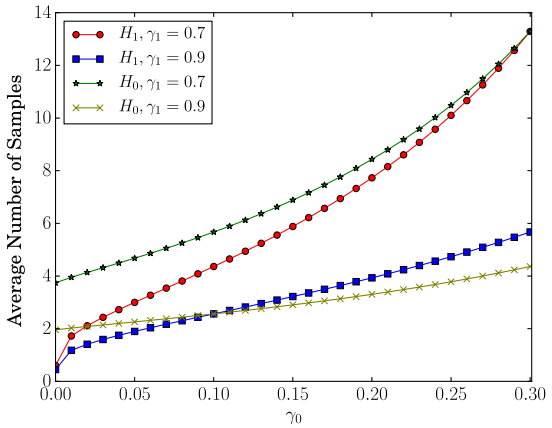

FIGURE 4. The effect of γ_0 and γ_1 on the expected number of samples conditioned on H_0 (no worm) and H_1 (worm).

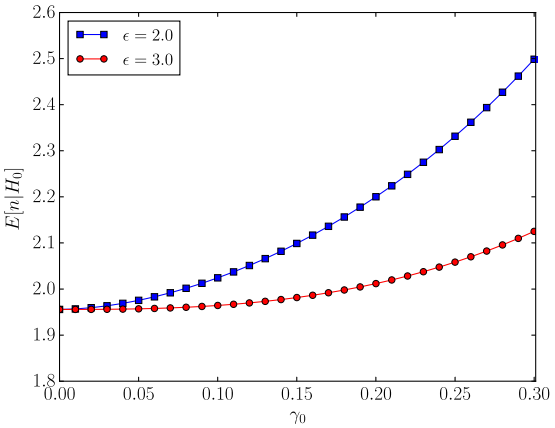

FIGURE 5. The effect of γ_0 on the expected number of samples conditioned on H_0 (no worm) when $\gamma_1 = 0.9$, $\alpha' = \beta' = 0.01$.

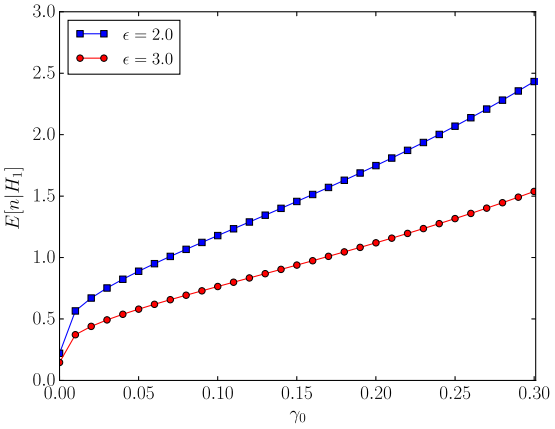

FIGURE 6. The effect of γ_0 on the expected number of samples conditioned on H_1 (worm) when $\gamma_1 = 0.9$, $\alpha' = \beta' = 0.01$.

1) AVERAGE NUMBER OF SAMPLES FOR A DECISION

We denote by n the number of samples that are required for the SPRT to terminate. n is considered to be a random variable because it is determined in accordance with the types of samples. Recall that H_0 is the null hypothesis (the benign case) and H_1 is the alternate hypothesis (a worm is being propagated). According to [21], we calculate $E[n]$ conditioned on hypotheses H_0 and H_1 as follows:

$$E[n|H_0] = \frac{(1-\alpha')\ln\frac{\beta'}{1-\alpha'} + \alpha'\ln\frac{1-\beta'}{\alpha'}}{\gamma_0\ln\frac{\gamma_1}{\gamma_0} + (1-\gamma_0)\ln\frac{1-\gamma_1}{1-\gamma_0}}$$
$$E[n|H_1] = \frac{\beta'\ln\frac{\beta'}{1-\alpha'} + (1-\beta')\ln\frac{1-\beta'}{\alpha'}}{\gamma_1\ln\frac{\gamma_1}{\gamma_0} + (1-\gamma_1)\ln\frac{1-\gamma_1}{1-\gamma_0}} \quad (7)$$

Figure 4 shows how γ_0 and γ_1 affect $E[n|H_0]$ and $E[n|H_1]$ when $\alpha' = \beta' = 0.01$. We see that the SPRT reaches a decision with a small number of samples when γ_0 and γ_1 are set to a small value and a large value, respectively. We also observe that $E[n|H_0]$ is larger than $E[n|H_1]$ when γ_0 is a small value. However, as γ_0 increases, the gap between $E[n|H_0]$ and $E[n|H_1]$ decreases and eventually $E[n|H_1]$ exceeds $E[n|H_0]$. This means that a small value of γ_0 contributes to the faster termination of the SPRT in acceptance of the alternate hypothesis than the null one.

Recall that ϵ is selected per time round uniformly at random from the range of $[1, \epsilon_{max})$ such that $\epsilon_{max} > 1$. By replacing γ_0 with $(\gamma_0)^\epsilon$ in $E[n|H_0]$ and $E[n|H_1]$ of the SPRT, we calculate $E[n|H_0]$ and $E[n|H_1]$ of the Randomized-SPRT as follows:

$$E[n|H_0] = \frac{(1-\alpha')\ln\frac{\beta'}{1-\alpha'} + \alpha'\ln\frac{1-\beta'}{\alpha'}}{(\gamma_0)^\epsilon\ln\frac{\gamma_1}{(\gamma_0)^\epsilon} + (1-(\gamma_0)^\epsilon)\ln\frac{1-\gamma_1}{1-(\gamma_0)^\epsilon}}$$
$$E[n|H_1] = \frac{\beta'\ln\frac{\beta'}{1-\alpha'} + (1-\beta')\ln\frac{1-\beta'}{\alpha'}}{\gamma_1\ln\frac{\gamma_1}{(\gamma_0)^\epsilon} + (1-\gamma_1)\ln\frac{1-\gamma_1}{1-(\gamma_0)^\epsilon}}$$

As shown in Figures 5 and 6, we examine how ϵ affects the number of observations needed to make a decision, $E[n|H_0]$ and $E[n|H_1]$. We vary γ_0 from 0.0 to 0.3 and fix $\gamma_1 = 0.9$, $\alpha' = \beta' = 0.01$. We first notice that the Randomized-SPRT requires a small number of samples on average to make a decision. Given a value of γ_0, we observe that an increase in ϵ contributes to a decrease in both $E[n|H_0]$ and $E[n|H_1]$. We also see that the growth rates of $E[n|H_0]$ and $E[n$ in the Randomized-SPRT are slower than the ones in the SPRT. This is mainly because $\gamma_0^\epsilon < \gamma_0$ holds, and thus the Randomized-SPRT requires fewer samples on average than the SPRT.

2) COMMUNICATION OVERHEAD

We define communication overhead as the average number of communication patterns that need to be sent in the network per time slot. In the benign case, in which there are no compromised nodes in the network, the attestation-only phase is only performed without going into the SPRT-attestation phase, and hence the communication overhead is zero.

In the worm case, the SPRT-attestation phase is performed by benign nodes among whose neighbors the fraction of compromised nodes is at least τ. Assume that there are N sensor nodes in the network. We also assume that p_τ is the fraction of nodes performing the SPRT-attestation phase in

the network. Furthermore, we assume that every node performing the SPRT-attestation phase receives on an average b packets per time slot. Then the communication overhead will be calculated as $N \times p_\tau \times b$. This means that every sensor node needs to send $p_\tau \times b$ communication patterns per time slot. This overhead will be temporarily incurred until worm propagation is completely stopped by our proposed scheme and the compromised nodes are revoked from the network. Once all compromised nodes are revoked, the network will go back to benign case with zero overhead. Hence, the communication overhead is reasonable.

3) STORAGE OVERHEAD

We define storage overhead as the average number of communication patterns that need to be stored by a sensor node. Recall that each detector erases the saved information at the end of each time slot. Moreover, the number of communication patterns received by a detector is equal to the number stored by them.

In the benign case, the storage overhead is zero because the SPRT-attestation phase is not executed. In the worm case, the storage overhead is computed as follows. There are $N \times p_d$ detectors on average in the network, where p_d is the detector selection probability. Assume that a communication pattern is received by c detectors on an average. By using the analysis derived from the communication overhead computation, the total number of communication patterns to be stored per time slot in the network is calculated as $\frac{c \times N \times p_\tau \times b}{N \times p_d} = \frac{c \times p_\tau \times b}{p_d}$. As a result, storage overhead will be $\frac{c \times p_\tau \times b}{N \times p_d} \approx O(\frac{c \times b}{N})$. Similar to the communication overhead of the worm case, this overhead is reasonable because it will be temporarily maintained until worm propagation is completely detected by our proposed scheme and the network goes back to the benign case.

V. SIMULATION STUDY

In this section, we will first describe our simulation environment and then present the simulation results.

For simplicity, we call our proposed scheme Randomized-SPRT. We call our prior work [8] Attestation-Only.

A. SIMULATION ENVIRONMENT

We developed a simple simulation program to evaluate the Randomized-SPRT and compare it with Attestation-Only. In our simulation, we place 500 sensor nodes in a square region of 200 m $\times$ 200 m and set the communication radius of each sensor node to 50 m. We adopt a group deployment strategy in which a group of sensor nodes is placed in the field according to the two-dimensional Gaussian distribution:

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

where (x_g, y_g) is the group deployment point and σ is the standard deviation. We set the number of groups to 25, with 20 nodes in each group. We also set $\sigma = 50$, which is equivalent to the communication radius of a sensor node. Although time synchronization is not needed in the network as mentioned in Section IV-C.3, for a simple time modeling, we divide the entire time domain into 90 time slots such that a time slot duration is 10 simulation seconds and perform the simulation in units of time slots.

For Randomized-SPRT, we configure the detector selection probability to be $p_d = 0.1$. We also set both the user-configured false positive threshold α and the false negative threshold β to 0.01, and we set the lower threshold γ_0 and the upper threshold γ_1 to 0.1 and 0.9, respectively. The rationale behind these configurations is discussed in Section IV-E. Additionally, we configure $\epsilon_{max} = 5.0$, and thus ϵ is selected in each round uniformly at random from the range of [1.0, 5.0).

Moreover, we set $\tau = 0.01$ which is the threshold used to initiate the Randomized-SPRT from the attestation-only phase. We configure $m = 50, 100$ for the size of the set A of attestees in the attestation-only phase.

We consider three cases for our evaluation: *benign*, *benign-compromise*, and *worm*. In the benign case, nodes only send benign packets. In this case, we focus on the performance of the attestation-only phase under benign traffic. In the benign-compromise case, nodes only send benign packets, but there are a few nodes individually captured and compromised in the network. In this case, we focus on the performance of both the attestation-only and SPRT-attestation phases under benign traffic. In the worm case, nodes only send worm packets, and we focus on worm propagation detection.

In the benign case, we consider the data aggregation scenario, which is the dominant communication pattern in sensor networks [9]. Specifically, each node elects itself as an aggregator with probability 0.1. Each aggregator collects benign data packets from its source member nodes. We model the transmission of benign data packets from a source member node to an aggregator as a homogeneous Poisson process with rate parameter λ. Therefore, the inter-transmission times of benign data packets follow the exponential distribution. More specifically, the inter-transmission time between two consecutive benign data packets is calculated as $-\frac{\ln(U)}{\lambda}$, where U is uniform random variate, $0 \leq U < 1$. We also consider the configurations of $\lambda = 0.05, 0.1, 0.15$. Under these configurations, each source member sends on average 45, 90, 135 benign packets, respectively, to the aggregator in the 900 simulation seconds. If the ratio of the number of collected data packets to the number of source members is greater than or equal to a preset data aggregation threshold δ, each aggregator performs the data aggregation process and forwards an aggregated version of the data to a randomly chosen parent aggregator. The data further propagates along a tree hierarchy of aggregators, where the base station is the root. We set $\delta = 1.0$. Since there are only benign nodes in the network, only the attestation-only phase is executed. We thereby examine the performance of the attestation-only phase under benign traffic.

In the benign-compromise case, benign traffic is generated and transmitted through data aggregation as in the benign case. Additionally, we place $N_c = 1, 5$ individually captured

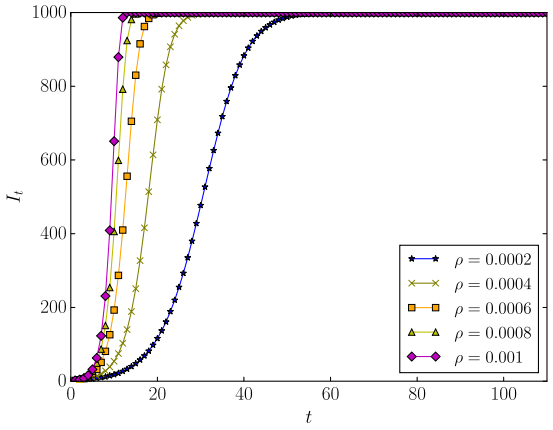

FIGURE 7. Cumulative number of infected nodes I_t vs. Time slot t when there are no defense mechanisms against worm propagation.

and compromised nodes in the network. This case is an intermediate stage between the benign case and the worm case. In this case, the attacker can capture and compromise a few number of nodes and prepare for worm propagation with these compromised nodes. Since benign nodes coexist with compromised nodes in the network, both the attestation-only and SPRT-attestation phases are performed, and hence we explore the performance of these two phases under benign traffic.

In the worm case, as described in Section III, we employ a discrete-time version of the simple epidemic model [2] together with hop-by-hop and random worm propagation strategies. Recall that ρ is the pairwise infection rate, I_0 represents the number of worm originators and I_t is the cumulative infection quota from the 0th time slot to the tth time slot. In our simulation, we consider a single worm originator and thus I_0 is set to one. As shown in Figure 7, we simulate how I_t grows over time with a variety of ρ values when the worm is propagated without detection in the discrete-time model. We see that I_t increases in line with the rise of t. We also observe that the rise of ρ results in a reduction in the number of time slots required for all sensor nodes to be infected. Based on these simulation results, we set ρ from 0.0002 to 0.001 in increments of 0.0002.

B. SIMULATION RESULTS

We use the metrics in Table 1 to evaluate the performance of Randomized-SPRT and compare it to Attestation-Only. We also present the average results for 1000 runs of the simulation in each configuration such that each run is executed for 90 time slots. For each run, we acquire each metric as the average of the results of the Randomized-SPRT that are performed.

We summarize our findings in terms of average number of samples in the Randomized-SPRT, worm defense capability, and overheads. Recall that our proposed scheme utilizes remote software attestation with virtually zero false positives and we do not present any results on false positives.

TABLE 1. Performance evaluation metrics.

Number of Samples	Ave. no. of samples required for the test procedure to reach a decision.
Fraction of Infected Nodes	Ave. fraction of infected nodes when all worm propagations are detected and blocked in the network.
Attester Notification Overhead	Ave. no. of attester notification messages sent by a node per time slot.
Attestation Overhead	Ave. no. of attestations performed by a node per time slot.
Communication Pattern Overhead	Ave. no. of communication patterns that are sent by a node per time slot.
Storage Overhead	Ave. no. of communication patterns that need to be stored in a node per time slot.
Stop Time Slot	Ave. no. of time slots when all worm propagations are detected and stopped in the network.

1) NUMBER OF SAMPLES TO REACH A DECISION WITH RANDOMIZED-SPRT

The Randomized-SPRT is applied only when some compromised nodes are detected using Attestation-Only, i.e. only for the benign-compromise and worm cases. In both the benign-compromise and worm cases for all settings we explored ($\lambda = 0.05, 0.1, 0.15$, $\delta = 1.0$), the average number of samples was less than 2.4. Most importantly, this means that the worm was detected quickly in all cases.

2) WORM DEFENSE CAPABILITY

We now present simulation results that show that Randomized-SPRT has significantly better worm detection and blocking capability than Attestation-Only, preventing a worm from gaining more than 13.5% of the nodes in the network in the worst case.

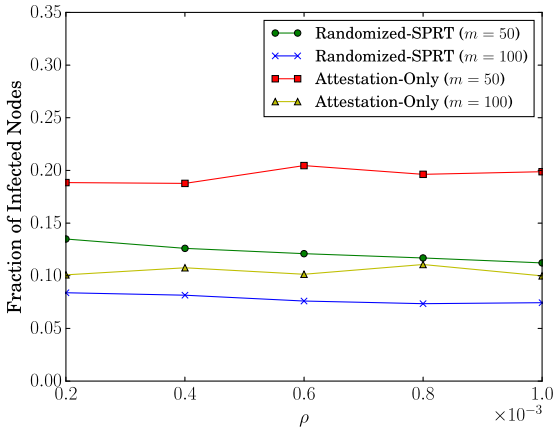

FIGURE 8. The effect pairwise infection rate ρ on the average fraction of infected nodes in the worm case.

As shown in Figure 8, for Randomized-SPRT, the average fraction of nodes infected by a fast worm with ρ =

0.001 reaches its maximum of 11.2% and 7.4% for the settings $m = 50$ and $m = 100$, respectively. This indicates that Randomized-SPRT substantially restrains the number of nodes infected by a fast worm below at most 56 out of 500 nodes in the network, thereby minimizing the damage incurred by fast-worm propagation. Additionally, the average fraction of nodes infected by a slow worm with $\rho = 0.0002$ reaches its maximum of 13.5% and 8.4% for the settings $m = 50$ and $m = 100$, respectively. Thus, Randomized-SPRT substantially suppresses the number of nodes infected by a slow worm below at most 67.5 nodes out of 500 nodes in the network. Although the number of infected nodes is larger than with the fast-worm, it is still low for detecting a stealthy worm that aims to evade our mechanism. Given a value of ρ, we notice that the average fraction of nodes infected by a worm tends to decrease as m increases. This implies that the more attestees set up in the network, the more infectious nodes are detected in the attestation-only phase without going into the SPRT-attestation phase, leading to faster blocking of the worm infection.

Note that the average fraction of infected nodes is not greatly affected by ρ. This signifies an advantage of our proposed scheme in the sense that Randomized-SPRT can maintain effective worm detection capability irrespective of the infection rate.

When comparing Randomized-SPRT to Attestation-Only, we see that Randomized-SPRT blocks worm propagation with fewer infected nodes than Attestation-Only in all configurations of m and ρ. In particular, when $m = 50$, Randomized-SPRT reduces the average number of infected nodes by between 28.4% and 43.6% compared to Attestation-Only. When $m = 100$, Randomized-SPRT cuts infections by between 16.9% and 33.5%. This demonstrates that our proposed scheme outperforms our prior work in terms of worm detection capability.

Figure 9 indicates the *stop time slot* at which all worm propagation is detected and blocked for both Randomized-SPRT and Attestation-Only. Worm propagation is stopped sooner for faster worms than slower ones. We also observe that an increase in m results in an earlier stop time slot, since infectious node detection in the attestation-only phase is sped up. Furthermore, the stop time slot in Randomized-SPRT is several time slots earlier than for Attestation-Only in all configurations of m and ρ.

3) OVERHEADS

In the following, we show simulation results demonstrating that the overheads of Randomized-SPRT are quite low and acceptable for a sensor network.

In the benign case, as shown in Figure 10, attestation overhead ranges from 0.05 to 0.29 attestations per node per time slot for all values of λ and m. This indicates that the Randomized-SPRT requires very small attestation overhead under benign traffic. Additionally, we see that increases in the number of attestees and in benign traffic lead to an increase in attestation overhead. Note that the benign case is virtually

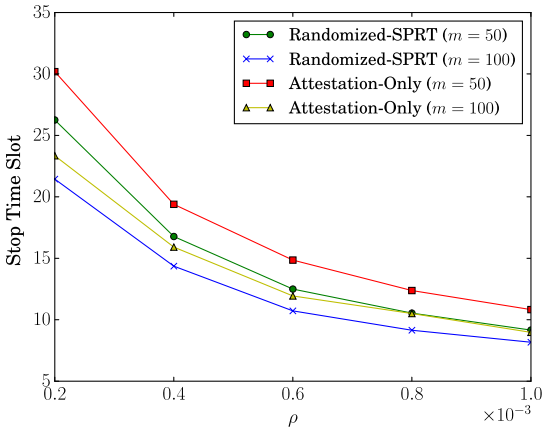

FIGURE 9. The effect of pairwise infection rate ρ and number of attestees m on the average time slot when to block worm propagation in the worm case.

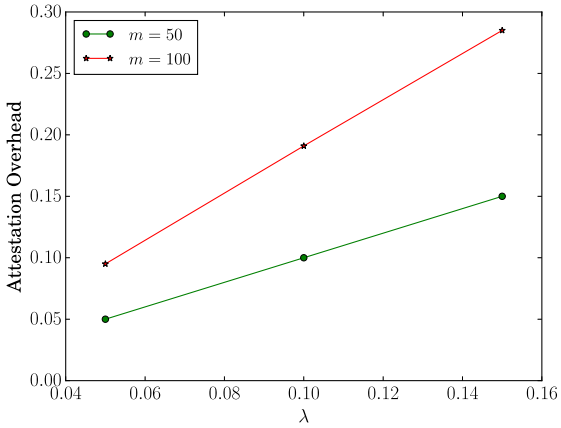

FIGURE 10. The impact of the number of attestees m and benign traffic parameter λ on the average number of attestations per node per time slot in the benign case.

the same as Attestation-Only in the sense that there are no compromised nodes and no activation of the SPRT-attestation phase.

In the benign-compromise case, attester notification overhead varies from 0.002 to 0.003 when there is one compromised node and varies from 0.016 to 0.017 when there are five compromised nodes. Despite the increase, the absolute overhead rate is still quite low. As shown in Figure 11, attestation overhead tends to grow as the number compromised nodes, the number of attestees, and benign traffic increase. However, the average number of attestation operations per node per time slot is below 0.7 and thus quite low. As shown in Figures 12, 13, both communication pattern and storage overheads tend to increase in accordance with a rise in the number of compromised nodes and benign traffic. On the other hand, a decrease in the number of attestees contributes to an increase in the communication pattern and storage overheads. We infer from this observation that the smaller number of attestees leads to less frequent execution of the attestation-only phase, resulting in more frequent execution of the SPRT-attestation phase and growth in the communication pattern and storage overheads. However, we observe that the

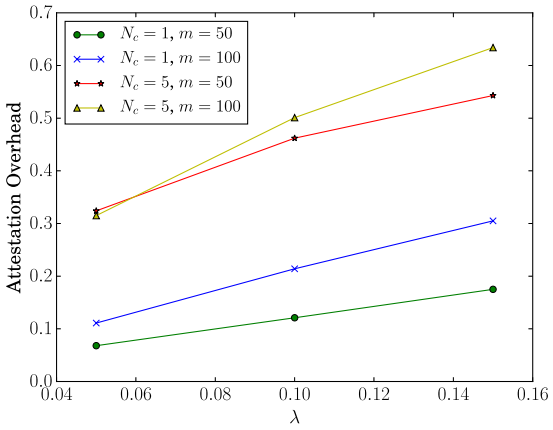

FIGURE 11. The impact of the number of attestees m and benign traffic parameter λ with number of compromised nodes N_c on the average number of attestations per node per time slot in the benign-compromise case.

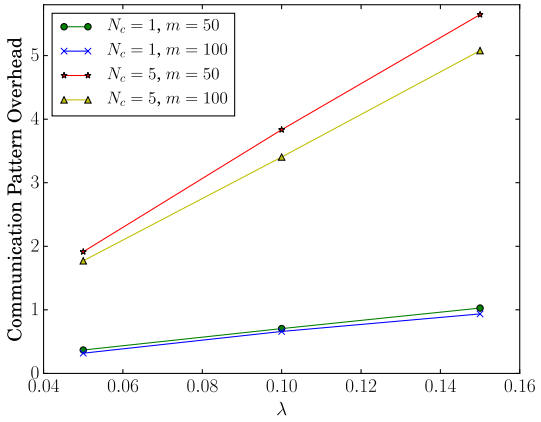

FIGURE 12. The effect of attestees m and benign traffic parameter λ with number of compromised nodes N_c on the average number of communication patterns sent by a node per time slot in the benign-compromise case.

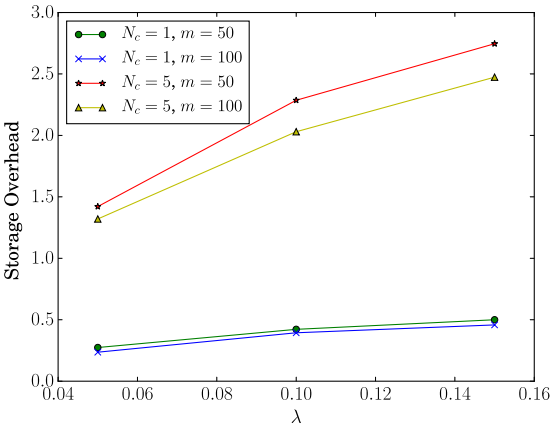

FIGURE 13. The effect of attestees m and benign traffic parameter λ with number of compromised nodes N_c on the average number of communication patterns stored in a node per time slot in the benign-compromise case.

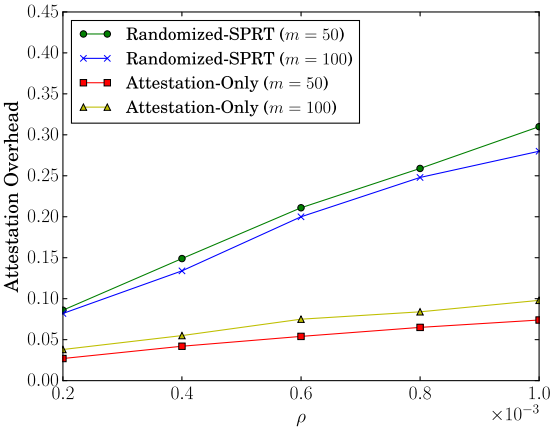

FIGURE 14. The impact of pairwise infection rate ρ and number of attestees m on the average number of attestations per node per time slot in the worm case.

communication and storage overheads are still reasonably sustained at below six patterns and three patterns, respectively.

In summary, we see that the number of compromised nodes plays a key role in incurring the overheads in benign-compromise case. This is mainly because an increase in node compromise expedites the execution of the SPRT-attestation phase, leading to expansion in overheads caused by the SPRT. However, the attacker would prefer to not need to capture and compromise a large number of nodes and instead prefer a worm or other wide-spread, automated attack.

In the worm case, Figure 14 shows how ρ and m affect the attestation overheads of Randomized-SPRT and Attestation-Only. In Randomized-SPRT, we see that the attestation overhead tends to increase with a rise in the worm infection rate ρ. Note that each node performs at most 0.31 attestations on average per time slot in all cases of (m, ρ). This indicates that Randomized-SPRT requires a very small number of attestations for worm detection and blocking. In addition, we observe that Randomized-SPRT needs slightly higher attestation overhead than Attestation-Only for all configurations of ρ and m. The additional attestation overhead incurred by Randomized-SPRT is due to the SPRT process, but it does not exceed 0.24 attestations per time slot on average, which is very small when compared to the benefit of more effective worm detection from employing the Randomized-SPRT.

As shown in Figure 15, communication pattern overhead with Randomized-SPRT increases as ρ increases. Each node, however, generates fewer than 0.09 communication patterns on average per time slot in all cases of (m, ρ). We also note that the communication pattern overhead when $m = 100$ is lower than when $m = 50$. This implies that for higher m, the attestation-only phase is more frequently performed than the SPRT-attestation phase, resulting in the occurrence of the lower communication pattern overhead. Storage overhead is equal to the communication pattern overhead in the worm case. This is because worm always targets susceptible nodes for infection and thus all communication patterns incurred by worm propagation are distinct, causing all communication patterns be stored in detectors.

Figure 16 shows how ρ and m affect the attester notification overheads for Randomized-SPRT and

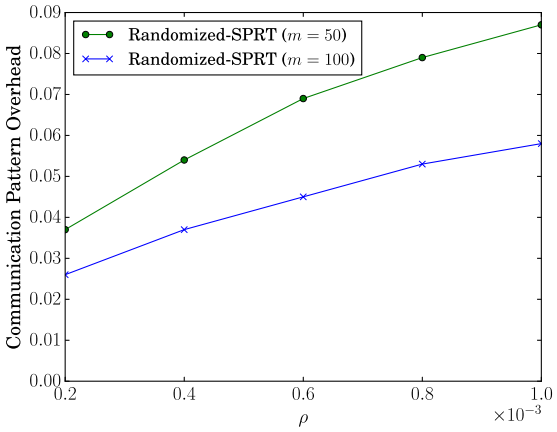

FIGURE 15. The effect of pairwise infection rate ρ and number of attestees m on the average number of communication patterns sent by a node per time slot in the worm case.

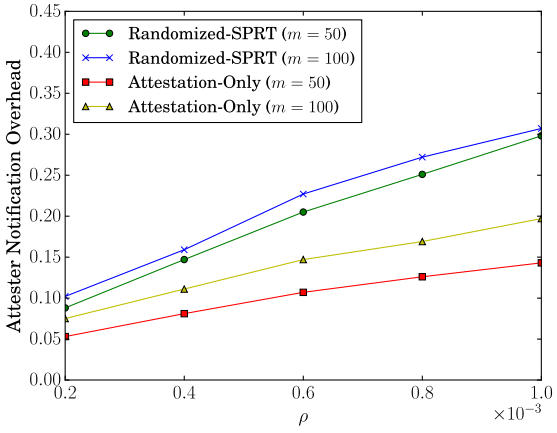

FIGURE 16. The effect of pairwise infection rate ρ and number of attestees m on the the average number of attester notification messages sent by a node per time slot in the worm case.

Attestation-Only. As ρ grows, attester notification overhead in the Randomized-SPRT tends to increase. This indicates that fast worm infection speeds up the dissemination of the attester notification messages. As expected, the attester notification overhead in Randomized-SPRT is higher than the one in Attestation-Only. This makes sense because the SPRT process incurs the cost of extra attester notification messages to achieve a superior worm detection capability. Nevertheless, in all cases of (m, ρ) for Randomized-SPRT, each detector sends to their neighbors fewer than 0.31 attester notification messages on average per time slot.

VI. CONCLUSIONS

In this paper, we proposed a Randomized-SPRT scheme for worm detection in wireless sensor networks. We have analyzed our scheme and showed that it detects worm propagation in just a few samples, thereby substantially restricting the number of infected nodes while incurring reasonable communication and storage overheads. We also evaluated the scheme through simulation, showing that our scheme achieves fast worm propagation detection with little attestation overhead. In a reasonably short time, e.g. between 8 and 27 time slots, our scheme blocks worm propagation with at most 13.5% of the network infected. These results are significantly better than a scheme based only on node attestation.

ACKNOWLEDGEMENTS

The authors would like to thank Donggang Liu for discussions on this research.

REFERENCES

[1] T. Abuhmed, N. Nyamaa, and D. Nyang, "Software-based remote code attestation in wireless sensor network," in *Proc. IEEE GLOBECOM*, Nov. 2009, pp. 1–8.
[2] D. J. Daley and J. Gani, *Epidemic Modeling: An Introduction*. Cambridge, U.K.: Cambridge Univ. Press, 1999.
[3] A. Francillon and C. Castelluccia, "Code injection attacks on harvard-architecture devices," in *Proc. ACM CCS*, Oct. 2008, pp. 15–26.
[4] T. Goodspeed, "Stack overflow exploits for MSP430 wireless sensors over 802.15.4," in *Proc. Texas Instrum. Developper Conf.*, Feb. 2008.
[5] Q. Gu and R. Noorani, "Towards self-propagate mal-packets in sensor networks," in *Proc. ACM WiSec*, Mar. 2008, pp. 172–182.
[6] N. Gui, E. Zhai, J. Hu, and Z. Chen, "SWORDS: Improving sensor networks immunity under worm attacks," in *Proc. Web-Age Inf. Manage. (WAIM)*, Jul. 2010, pp. 86–96.
[7] J.-W. Ho, "A framework for robust detection and prevention of wide-spread node compromise in wireless sensor networks," Doctoral dissertation, Dept. Comput. Sci. Eng., Univ. Texas Arlington, Arlington, TX, USA, 2010.
[8] J.-W. Ho, "Distributed software-attestation defense against sensor worm propagation," *J. Sensors*, vol. 2015, 2015, Art. no. 874782, doi: 10.1155/2015/874782.
[9] C. Karlof and D. Wagner, "Secure routing in wireless sensor networks: Attacks and countermeasures," *Ad Hoc Netw.*, vol. 1, nos. 2–3, pp. 293–315, Sep. 2003.
[10] H.-A. Kim and B. Karp, "Autograph: Toward automated, distributed worm signature detection," in *Proc. 13th USENIX Secur. Symp.*, 2004, pp. 271–286.
[11] X. Kovah, C. Kallenberg, C. Weathers, A. Herzog, M. Albin, and J. Butterworth, "New results for timing-based attestation," in *Proc. IEEE Symp. Secur. Privacy*, May 2012, pp. 239–253.
[12] R. C. Linger, "Systematic generation of stochastic diversity as an intrusion barrier in survivable systems software," in *Proc. 32nd Annu. Hawaii Int. Conf. Syst. Sci. (HICSS)*, Jan. 1999.
[13] D. Moore, C. Shannon, G. M. Voelker, and S. Savage, "Internet quarantine: Requirements for containing self-propagating code," in *Proc. IEEE INFOCOM*, Mar. 2003, pp. 1901–1910.
[14] J. Newsome, B. Karp, and D. Song, "Polygraph: Automatically generating signatures for polymorphic worms," in *Proc. IEEE Symp. Secur. Privacy*, May 2005, pp. 226–241.
[15] A.-R. Sadeghi, S. Schulz, and C. Wachsmann, "Lightweight remote attestation using physical functions," in *Proc. ACM WiSec*, Jun. 2011, pp. 109–114.
[16] S. E. Schechter, J. Jung, and A. W. Berger, "Fast detection of scanning worm infections," in *Proc. RAID*, Sep. 2004, pp. 59–81.
[17] A. Seshadri, A. Perrig, L. van Doorn, and P. Khosla, "SWATT: SoftWare-based attestation for embedded devices," in *Proc. IEEE Symp. Secur. Privacy*, May 2004, pp. 272–282.
[18] S. Shen, H. Li, R. Han, A. V. Vasilakos, Y. Wang, and Q. Cao, "Differential game-based strategies for preventing malware propagation in wireless sensor networks," *IEEE Trans. Inf. Forensics Security*, vol. 9, no. 11, pp. 1962–1973, Nov. 2014.
[19] B. Sun, G. Yan, Y. Xiao, and T. A. Yang, "Self-propagating mal-packets in wireless sensor networks: Dynamics and defense implications," *Ad Hoc Netw.*, vol. 7, no. 8, pp. 1489–1500, Nov. 2009.
[20] (2003). *Trusted Computing Group (TCG)*. [Online]. Available: https://www.trustedcomputinggroup.org/
[21] A. Wald, *Sequential Analysis*. New York, NY, USA: Dover, 2004.
[22] Y. Yang, X. Wang, S. Zhu, and G. Cao, "Distributed software-based attestation for node compromise detection in sensor networks," in *Proc. IEEE SRDS*, Oct. 2007, pp. 219–230.

[23] Y. Yang, S. Zhu, and G. Cao, "Improving sensor network immunity under worm attacks: A software diversity approach," in *Proc. ACM MobiHoc*, May 2008, pp. 149–158.
[24] Y. Yang, S. Zhu, and G. Cao, "Improving sensor network immunity under worm attacks," *Ad hoc Netw.*, vol. 47, no. 1, pp. 26–40, Sep. 2016.
[25] Y. Liu, W. Zhang, S. Bai, and C. Wang, "Defending sensor worm attack using software diversity approach," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2011, pp. 1–5.
[26] C. Zou, L. Gao, W. Gong, and D. Towsley, "Monitoring and early warning for Internet worms," in *Proc. ACM CCS*, Oct. 2003, pp. 190–199.

JUN-WON HO received the B.S. degree from the Department of Computer Science, Yonsei University, Seoul, South Korea, the M.S. degree from the Department of Electrical and Computer Engineering, University of Wisconsin-Madison, and the Ph.D. degree from the Department of Computer Science, University of Texas at Arlington, in 2010. His dissertation work focuses on the detection and prevention of wide-spread node compromise in wireless sensor networks. He is currently an Assistant Professor with Seoul Women's University. His current research interests include system and mobile security.

MATTHEW WRIGHT received the B.S. degree in computer science from Harvey Mudd College, and the M.S. and Ph.D. degrees from the Department of Computer Science, University of Massachusetts, in 2002 and 2005, respectively. His dissertation work examined attacks and defenses of systems that provide anonymity online. He is the Director of the Center for Cybersecurity with RIT and a Professor of Computing Security. His current research interests include understanding the human element of security and security and privacy in all sorts of distributed systems, including peer-to-peer, mobile, and Internet of Things. He is a recipient of the NSF CAREER Award, the Outstanding Paper Award at the 2002 Symposium on Network and Distributed System Security, and the Outstanding Student Paper Award at the 2016 European Symposium on Research in Computer Security.