Quantifying Node Security in Wireless Sensor Networks under Worm Attacks

Alex Ramos[1], **Breno Aquino**[1], **Raimir Holanda Filho**[1], **Joel J. P. C. Rodrigues**[1,2,3]

[1]Graduate Program in Applied Computer Science (PPGIA)
University of Fortaleza (UNIFOR) – CE – Brazil

[2]National Institute of Telecommunications (Inatel) – MG – Brazil

[3]Instituto de Telecomunicações, Universidade da Beira Interior, Portugal

{alex.lacerda, brenoaquino}@edu.unifor.br,

raimir@unifor.br, joeljr@ieee.org

Abstract. *The peculiar characteristics of wireless sensor networks (WSNs) make them vulnerable to physical attacks. Once a sensor node is physically captured by an adversary, it can be modified not only to perform malicious activities to disrupt network operation but also to propagate malicious worms to infect other nodes. In the face of such a threatening scenario, the system administrator needs to be aware of which nodes may have been compromised, so that appropriate countermeasures can be taken in a timely fashion. This paper presents the Sensor Security Status (S3), a security metric model for estimating in an online manner the probability that a sensor node has been infected, based on both the interaction among nodes and the alerts from the intrusion detection system (IDS). Simulation results show that S3 can accurately estimate node security level with low performance overhead and power consumption.*

1. Introduction

The situational awareness provided by security metrics is essential to help system administrators take informed decisions regarding the security status of a network and its components [Zonouz et al. 2015]. In wireless sensor networks (WSNs), a metric that is able to quantify the security level of sensor nodes can be used, for example, to identify which nodes should be trusted (to provide reliable data) or which nodes need careful attention from attack response mechanisms. Ultimately, the security level of nodes can be used to determine the security status of the entire WSN and the sensor data it provides for users [Ramos and Filho 2015].

Although several security metrics have been proposed for traditional networks [Pamula et al. 2006, Wang et al. 2007], the unique characteristics of sensor nodes make it impossible to directly apply those metrics to WSNs. This requires security metrics especially designed to quantify security based on the specific attack types and vulnerabilities of sensor networks [Ramos and Filho 2015]. In particular, the resource constraints, and deployment in open areas make sensor nodes vulnerable to physical attacks. In this type of attack, an adversary captures a node and retrieves secret information from its memory in order to gain access to the network. In addition, the adversary can compromise the captured node to make it perform several malicious activities in the WSN (e.g., routing attacks like sinkhole, misdirection, etc) [Walters et al. 2007].

To easily gain full control of the network, the adversary may also attempt to have the captured node propagate malicious worms to compromise more nodes via wireless communication. Once a node is infected by the worm, it will be able to both behave maliciously (*e.g.,* carry out routing attacks) and re-propagate the worm to infect other nodes [Francillon and Castelluccia 2008, Haghighi et al. 2016]. This makes worms one of the most devastating types of attack.

To detect misbehaving nodes, an intrusion detection system (IDS) can be deployed in the WSN [Raza et al. 2013]. Although IDSs are usually not capable of identifying all intrusions that occur (because the dynamic operation of WSNs may make it difficult to distinguish normal behavior from malicious behavior), the malicious nodes that the IDS is actually able to identify indicate that the network is under attack and, consequently, that other nodes may also have been compromised.

Therefore, to effectively portray the current security status of nodes and provide administrators with useful information when attacks are occurring, a security metric for WSNs should consider two main factors: (1) benign nodes can be compromised by malicious nodes (*i.e.,* worm attacks); and (2) to avoid that some compromised nodes go undetected, the intrusions that the IDS is able to identify can be used to predict other intrusions. Intuitively, the greater the number of malicious nodes the IDS detects, the higher the chance that other nodes have also been compromised.

The few existing security metrics for WSNs ([Anand et al. 2005, Ramos and Filho 2015]) fail to handle these aspects, because they either ignore IDS alerts or disregard the fact that a node can be compromised by other nodes. To address those limitations, this paper presents a security metric model called *sensor security status (S3)*. The S3 model uses IDS alarms received in real-time to estimate how much the security level of each sensor node of a WSN has been affected by intrusions. This assessment is performed using an *attack propagation graph (APG)*. Considering that adversaries can take advantage of the network communication pattern to propagate worm attacks [Ho 2015], the APG captures how nodes can compromise others through their communication behavior.

The attack propagation graph is automatically constructed during an initial configuration phase when sensor nodes behave normally. Then, the APG is transformed into a Bayesian network (BN) so that inferences about the security status of nodes can be made. More precisely, when a new alert is raised by the IDS, a belief propagation algorithm, the Gibbs sampler [Casella and George 1992], is applied to compute the probability that each WSN node has been affected by the intrusion and has become compromised.

The remainder of this paper is organized as follows. Section 2 presents system models and assumptions. Section 3 describes the proposed S3 model. Section 4 provides an evaluation of S3. The past related work is reviewed in Section 5. Finally, Section 6 concludes this paper.

2. System Model and Assumptions

This section presents network, attack, and security models, as well as other assumptions considered.

2.1. Network Model

The base station (BS) is assumed to be a central command node with no resource constraint problem and it cannot be compromised by attacks. it is also assumed that the WSN

is comprised of static nodes that periodically send sensor readings to the base station by means of a multi-path routing protocol such as the standardized Routing Protocol for Low Power and Lossy Networks (RPL) [Winter et al. 2012]. In RPL, a destination-oriented directed acyclic graph (DODAG) is created to enable message forwarding from sensor nodes to the DODAG root (*i.e.,* the base station). Each node knows its RPL-parents but has no information regarding its children. Every node periodically chooses a *preferred* PRL-parent to forward its messages, as shown in the left hand side of Fig. 1. The preferred RPL-parent of each node is chosen from the parent set and is periodically updated according to some predefined routing metrics (*e.g.,* remaining energy, link quality).

In the physical and link layers, sensor nodes are assumed to implement the IEEE 802.15.4 protocol which is the *de facto* standard for low power and lossy wireless networks such as WSNs.

2.2. Attack Model

WSNs can be target of several types of attack [Walters et al. 2007]. S3 assumes that attacks can be initiated from node capture. A node that is physically captured by an adversary can perform malicious activities to disrupt network operation. Those activities include attacks such as selective forwarding, sinkhole, and data alteration [Raza et al. 2013]. In addition, a compromised node can transfer data packets with malicious code to compromise its neighbors (worm attack) [Francillon and Castelluccia 2008, Haghighi et al. 2016]. This worm propagation process can repeat itself and lead to the compromise of the whole network if countermeasures are not taken [Ho 2015]. Therefore, an adversary can compromise an entire WSN with a single node capture. It is important to note that researchers have developed practical worm attacks on both Harvard architecture (*e.g.,* Mica motes) [Francillon and Castelluccia 2008] and Von Neumann architecture (*e.g.,* TelosB) sensor devices [Giannetsos et al. 2009].

To attain their objective of maximizing the amount of compromised nodes, worms can apply different propagation strategies. Although broadcasting a worm to all neighbors may seem to be the best strategy, it can result in severe congestion of network traffic and hence decrease, or even stop, the propagation rate [Khayam and Radha 2005]. Therefore, we assume an intelligent worm, which seeks to maximize the number of infected nodes by using the normal communication pattern of the network to propagate itself, as discussed in [Ho 2015]. In a RPL-based WSN, this could be accomplished by making compromised nodes transmit the worm only to the current preferred RPL-parent. This strategy is also useful to avoid the worm propagation from being easily identified by any worm detection mechanism that could eventually be present in the network [Ho 2015].

2.3. Security Model

The S3 model assumes the existence of an IDS on the WSN. IDSs such as SVELTE [Raza et al. 2013], for example, can detect compromised nodes that perform malicious activities (*e.g.,* sinkhole, data alteration, etc.). A worm detection mechanism, such as the scheme proposed in [Ho 2015], may be present in the network as well. S3 also assumes the existence of response mechanisms that may attempt to recover malicious nodes detected by the IDS. Node recovery might be achieved in various different ways. A simple recovery approach would be to reload the node's program [De et al. 2009].

3. Sensor Security Status Metric

The goal of S3 is to automatically evaluate the security level of each sensor node in an online manner so as to provide awareness of how secure the network is. The intuition leveraged by S3 is that when a malicious node is identified by the IDS, it is possible that this node has been infected by a worm that have already managed to infect other nodes. Therefore, by using IDS alerts as evidence that an attacker is present in the WSN, S3 attempts to predict future attacks or attacks that may have already occurred but have not been detected by the IDS (yet).

To do so, S3 uses an attack propagation graph to assess how a worm could take advantage of the communication pattern of nodes to spread itself through the network. The APG, which is automatically built during an initial configuration phase, captures the communication dependencies among neighboring sensor nodes. After this phase, the APG is then turned into a Bayesian network that is combined with real-time IDS alerts by means of a node compromise dissemination analysis procedure to estimate the probabilities that sensor nodes have been affected by an attacker.

More specifically, the security measure provided by S3 for each node indicates the probability that the node has been compromised by a malicious worm given that one or more compromised nodes have been detected by the IDS. Accordingly, the security measure provided by S3 is a real value lying between 0 and 1 (inclusive). A larger S3 value indicates less security. Every time the current system state changes, *i.e.,* when a new alert is raised by the IDS or a malicious node is recovered, then the node security measures provided by S3 are updated. In the following, the formalism and mathematical models used to compute S3 are presented.

3.1. Attack Propagation Graph

The APG is a directed acyclic graph which captures all possible paths that sensor nodes can use to forward messages to the base station. Each vertex in the APG represents a sensor node and a direct communication dependency between two nodes exists if data packets (messages) flow from one node to the other, in the direction of the base station. This relationship is represented in the APG by an arrow (directed edge) between the two nodes. For example, if node n_j receives sensor data from node n_i, it is said that node n_j is dependent on n_i, which is represented as $n_i \rightarrow n_j$. Each arrow is labeled with a probability value $Pr(n_i \rightarrow n_j)$ that indicates the fraction of times node n_i forwards messages to the base station through n_j.

The APG of a sample RPL-based WSN is shown in the right hand side of Fig. 1. The probabilities on arrows represent $Pr(n_i \rightarrow n_j)$ values. For example, node E forwards sensor data to the base station through nodes A and B with probabilities 0.85 and 0.1, respectively. In other words, A was the preferred RPL-parent of E 85% of the time, while 10% of the time B was the preferred RPL-parent.

Notice that the APG only represents the fraction of messages successfully delivered to next-hop neighbors. This means that, due to packet loss, the $Pr(n \rightarrow .)$ values of a given node n may not sum up to 1. For example, while 0.95 of E's messages have been successfully delivered to nodes A and B, 0.05 have been lost (*i.e.,* $1 - (0.85 + 0.1)$). This allows the APG to capture the lossy behavior of WSN's communication links.

Generation of the APG: To obtain the information required to build the APG, an information collection agent (ICA) is installed in each sensor node. During the initial

configuration phase, this agent maintains a variable that counts the number of messages the node has successfully delivered to each of its parents. To do so, the agent periodically sends dummy messages to the current preferred RPL-parent of the underlying node. Since RPL periodically chooses a new preferred parent from the parent set of each node, when the configuration phase terminates, the counters stored by the agent will correspond to the number of times each parent was chosen as the preferred parent.

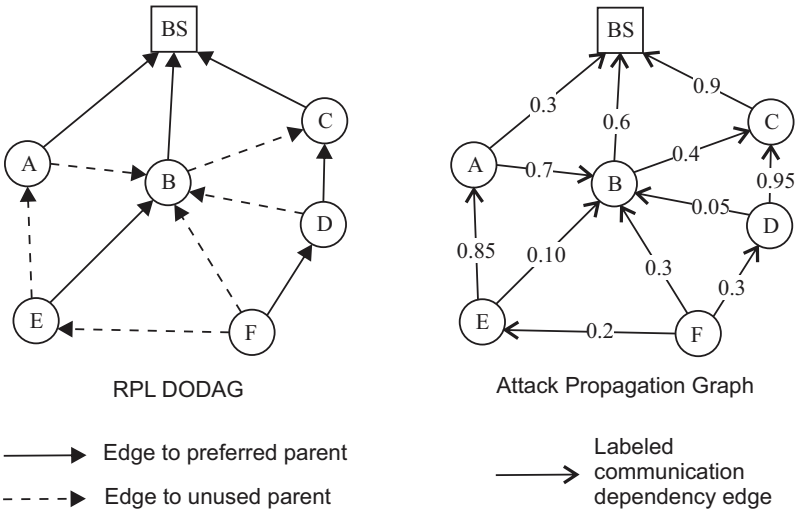

Figure 1. RPL DODAG and corresponding APG of sample WSN.

The number of messages lost is also stored. To accomplish this, ICA interacts with the link layer protocol. Since the IEEE 802.15.4 is an acknowledgment-based protocol, ICA knows that the node's messages have been successfully delivered to the preferred parent if an ACK frame is received by the link layer. On the other hand, if an ACK is not received by the sender after a predefined threshold time (and a specific number of retransmissions), the link layer considers that the message has been lost. In this case, the lost messages counter of the sender is incremented by ICA.

In the end of the configuration phase, each sensor node sends its counters to the base station along with their associated node IDs. Specifically, a *counter message* sent to the base station by a given node A contains the ID of A, the IDs of A's RPL-parents and their respective (message delivery) counters collected by ICA. The counter message also contains the number of lost messages.

As soon as the counter messages of nodes arrive in the base station, they are parsed in order to generate a *frequency DAG*, which is similar to the APG but with message counters in the edge labels, rather than probabilities. When all counter messages are received, the resulting frequency DAG is then converted into an APG. This is accomplished by converting the frequency labels into probabilities, which is done by dividing each frequency label by the total number of messages sent by the source node (i.e., the total successfully delivered messages + the number of lost messages).

It should be highlighted that ICA only runs on sensor nodes during the initial configuration phase. Moreover, all other steps performed by S3 are performed in the base station, namely, frequency DAG generation, APG generation, BN generation, and BN inference. Furthermore, the number of dummy messages sent by ICA can be decreased

since ICA can take advantage of the sensor readings that are periodically sent by the nodes to the base station (as discussed in Section 2). Depending on the frequency those sensor readings are sent, ICA can maintain its counters updated even if no dummy messages are sent. Therefore, the overhead added by ICA to the WSN is as low as possible.

3.2. Bayesian Network

To model how worms can propagate by taking advantage of the communication pattern of sensor nodes, the APG is translated into a Bayesian network (BN) that captures the probabilities that each node can be compromised by other nodes. More precisely, each APG vertex is modeled as a Bernoulli random variable representing the security state of a node, *i.e.,* 1 (True) if the node is compromised, or 0 (False) otherwise. Since it is assumed that worms propagate according to the network communication behavior, each arrow will represent a cause-consequence relationship between two nodes, meaning that one node can be compromised by the other. Each arrow probability $Pr(n_i \rightarrow n_j)$ will then correspond to the probability that node n_j gets compromised by a worm sent by n_i (in the case that n_i has been directly or indirectly compromised by an attacker). For example, if n_i is a compromised node and $Pr(n_i \rightarrow n_j) = 0.85$, then n_j has 0.85 chance of being compromised by n_i since this number represents the probability that n_j receives a message (containing worms) from n_i (*i.e.,* , the probability that n_j is the preferred RPL-parent of n_i).

Since each node in the BN is directly compromised by its parent nodes (source of incoming arrows[1]), a conditional probability table (CPT) is created (with the aid of arrow probability values) and associated with each node. The CPT in a given node n stores the probability that this node gets compromised (or not) given different combination of states of its BN-parent nodes $Pa[n]$. In other words, the CPT corresponds to the conditional probability distribution $Pr(n|Pa[n])$. Formally, let $Pr(n) = 1 - Pr(\overline{n})$. For $p_n^i \in Pa[n]$, let a_i be the communication arrow $p_n^i \rightarrow n$. Considering that a node cannot be compromised by a worm if none of its BN-parents is compromised, then $Pr(n|Pa[n])$ is defined as follows:

$$Pr(n|Pa[n]) = \begin{cases} 0, & \forall p_n^i \in Pa[n], p_n^i = 0, \\ Pr\left(\bigcup_{p_n^i=1} a_i\right), & otherwise. \end{cases} \quad (1)$$

Considering that a node can become compromised by any of the BN-parents that is already compromised, then the probability $Pr\big(\bigcup_{p_n^i=1} a_i\big)$ is derived as follows:

$$Pr\left(\bigcup_{p_n^i=1} a_i\right) = 1 - \prod_{p_n^i=1} [1 - Pr(a_i)] \quad (2)$$

Fig. 2 illustrates how a CPT is generated for a sample BN. For example, node B cannot become compromised if none of its BN-parents is compromised, *i.e.,* $Pr(B|\overline{A}, \overline{C}) = 0$. If only the node A is compromised, B gets compromised only

[1]Notice that a RPL-parent node is the opposite of a BN-parent node. For example, for an edge $n_i \rightarrow n_j$, node n_j is the RPL parent of n_i, while n_i is the BN-parent of n_j.

when messages are received from A, *i.e.,* $Pr(B|A,\overline{C}) = 0.6$. If both BN-parents are compromised, B will become compromised when messages are received by either of its BN-parents, *i.e.,* $Pr(B|A,C) = 1-(1-0.6)\times(1-0.9) = 0.96$.

Note that since node A has no parents, its prior probability is set to a value very close to zero (*i.e.,* 0.0001). Alternatively, to account for other uncertainties (*e.g.,* node capture), the administrator could choose a prior probability value that represents his/her subjective belief on the likelihood that node A can be directly compromised by an adversary (rather than a parent node). This uncertainty could be extended to other nodes by changing the zero probability value in Eq. 1.

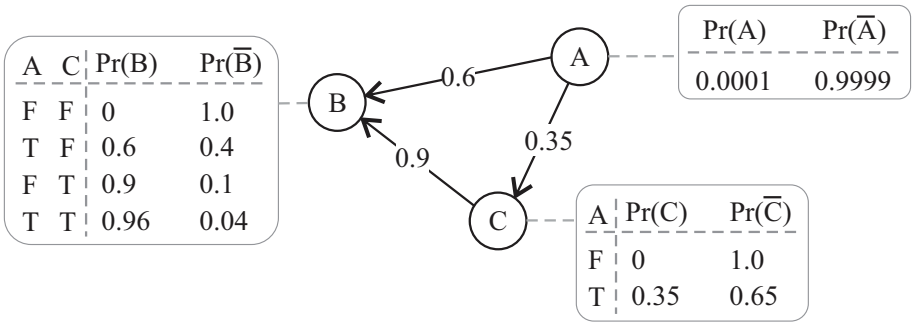

Figure 2. Bayesian network illustration.

3.3. Node Security Level Computation

When the initial configuration phase is concluded, the information collection agents are deactivated and the Bayesian network is generated from the APG in order to estimate the security level of sensor nodes in an online manner. Every time the current system condition changes (*i.e.,* when intrusions are observed by the IDS or compromised nodes are recovered), the state of each vertex in the BN is set accordingly. Then, Bayesian inference techniques of forward and backward propagation are used to update the probability that each sensor node is directly or indirectly affected by the compromised nodes.

Formally, let $N = \{n_1, ..., n_q\}$ be the set of vertices in the BN and $E = \{n'_1, ..., n'_r\} \subset N$ be the set of compromised nodes detected by the IDS (observed attack evidences). Notice that the state of each node in E is true, *i.e.,* $\forall n'_i \in E, n'_i = 1$. Let $n_j \in N - E$ be a node whose posterior probability has to be obtained (*i.e.,* a query node). We are interested in computing the posterior probability of n_j given E, *i.e.,* the conditional probability $Pr(n_j|E)$, which is given as follows:

$$Pr(n_j|E) = \frac{Pr(n_j, E)}{Pr(E)} = \frac{Pr(n_j, n'_1, ..., n'_r)}{Pr(n'_1, ..., n'_r)} \tag{3}$$

Let $H = \{n''_1, ..., n''_k\} \subset N$ be the set of nodes in the BN which are different from query nodes and evidence nodes (*i.e.,* hidden nodes), thus $N = \{n_j\} \cup E \cup H$. The numerator and denominator in Eq. 3 can be expressed using the joint probability of all BN nodes as follows:

$$Pr(n_j|E) = \frac{\sum_{n''_1,...,n''_k \in \{0,1\}} Pr(n_j, n'_1, ..., n'_r, n''_1, ..., n''_k)}{\sum_{n_j,n''_1,...,n''_k \in \{0,1\}} Pr(n'_1, ..., n'_r, n_j, n''_1, ..., n''_k)} \quad (4)$$

In a BN, the joint probability of all vertices is given by the chain rule as:

$$Pr(n_1, ..., n_q) = \prod_{j=1}^{q} Pr(n_j|Pa[n_j]) \quad (5)$$

By combining Eqs. 4 and 5, the posterior probability $Pr(n_j|E)$ can be solved for any node n_j. For example, in Fig. 2, suppose that nodes A and B are identified by the IDS as malicious. The posterior probability of C being compromised is calculated as follows:

$$\begin{aligned}
Pr(C|A,B) &= Pr(C,A,B)/Pr(A,B) \\
&= 0.46 \text{ where,} \\
Pr(C,A,B) &= 0.35 \cdot 0.0001 \cdot 0.96 \\
&= 0.0000336, \\
Pr(A,B) &= \sum_{C \in \{0,1\}} Pr(A,B,C) \\
&= (0.0001 \cdot 0.6 \cdot 0.65)_0 + (0.0001 \cdot 0.96 \cdot 0.35)_1 \\
&= 0.0000726
\end{aligned}$$

Since exact inference calculation procedures like the one presented above can become computationally infeasible for large BNs, s3 makes use of an approximate Monte Carlo inference algorithm, namely, the Gibbs sampler. In summary, the Gibbs sampler generates a sequence of samples from a joint probability distribution of a set of random variables $X = \{X_1, ..., X_n\}$. By using a large number of samples, it is possible to approximate the right joint distribution. Specifically, to compute a joint distribution $Pr(X = X_1, ..., X_n|e_1, ..., e_m)$, where e_i is an evidence, the Gibbs sampler initializes X to an arbitrary value in its state space and then samples an adjacent state, with the conditional probability $Pr(X|e)$ conducting the sampling procedure. Repeating the sampling procedure at sufficiently long intervals makes the joint distribution converge.

4. Performance Evaluation

In this section, a simulation-based evaluation of S3 is presented, in terms of its performance and accuracy. The experiments performed allowed to determine: (a) the minimum number of dummy messages that ICA needs to send during the configuration phase; (b) the energy overhead generated by ICA in the WSN; (c) the amount of time required by S3 to both build its fundamental data structures (i.e., APG and BN) and estimate the security metric value of sensor nodes; and finally, (d) the mean error of the estimated metric values.

4.1. Implementation and Experimental Setup

The information collection agent (ICA) is implemented in the Contiki OS [Dunkels et al. 2004], an open source and widely used operating system for WSNs and the Internet of things. Contiki uses extensively tested implementations of both IEEE 802.15.4 and RPL (contikiRPL). The RPL implementation is based on IPv6. Hence, uIP, an IP stack implementation in Contiki, is used to enable IP communication in the WSN.

To implement the approximate inference in the Bayesian network, the DlibC++ [King 2002] open source library is used. This library is widely adopted in both industry and academia.

The experiments were carried out in Cooja [Osterlind et al. 2006], the Contiki network simulator, which has demonstrated to generate realistic results [Raza et al. 2013]. Sensor nodes in Cooja run deployable code and are emulated at the hardware level. In the simulated WSNs, Tmote Sky [Polastre et al. 2005] nodes were used. The base station used was a real laptop that communicated with Cooja by means of a serial socket interface. The laptop was running Ubuntu 16.04 and had a 2.20 GHz Intel Core i5-5200U CPU and 4.0 GB of RAM. Each simulation scenario was run 10 times, and the average and the standard deviation of the results were computed to show their precision.

4.2. Minimum Number of ICA's Dummy Messages

Before estimating the security level of sensor nodes, S3 needs to capture the network communication pattern in the configuration phase and translate it into an APG. To do so, the number of messages sent by ICA should be large enough to build a model that correctly represents the actual network behavior. At the same time, to avoid adding too much overhead in the sensor nodes, the amount of ICA messages sent should be as small as possible.

In this section, the number of dummy messages required for the convergence of the APG parameters is evaluated for a typical WSN consisting of 30 nodes. In each sensor node, the information collection agent sends a total of $3,000$ dummy messages to the current preferred RPL-parent in order to update the counter variables. Therefore, a total of $90,000$ one-hop dummy messages are sent in the network.

In the described scenario, the first parameter evaluated is the number of edges in the APG. Notice that each time a RPL-parent is firstly chosen as preferred by a given node, a new edge should be added in the APG. Fig. 3 shows the APG size *vs.* the number of dummy messages transmitted. The number of edges quickly approaches 175, before $20,000$ messages are sent. Then, it starts stabilizing at approximately $30,000$ messages, i.e., when $1,000$ messages have been transmitted per node.

The second parameter analyzed is the convergence of the probability values (labels) of the APG edges. Specifically, the normalized edge probability updates (i.e., the absolute difference between the current and the updated values) are computed. Fig. 4 shows the average behavior of those values as each node sends its $3,000$ dummy messages. As can be seen, the normalized updates quickly converge to zero at approximately $1,000$ messages. This means that on average the edge probability values begin to converge when each node transmits $1,000$ messages.

It should be highlighted that the convergence of the APG parameters is heavily influenced by the way RPL selects the preferred parents, which in turn depends on the

objective functions and routing metrics configured by the network administrator. However, the experiments provided in this section have shown that when the default behavior of contikiRPL is used, $1,000$ dummy messages sent per node seems sufficient to generate an APG with suitable coverage.

In the simulated scenario, the time interval between two consecutive dummy messages was 35 seconds. Note that this value was used in the simulations but it is not the requirement for ICA. However, reasonably large interval values (like 35s) allow ICA to take advantage of sensor reading messages that nodes periodically send to the base station (usually around one per minute). Hence, ICA can reduce the number of dummy messages sent, which decreases the energy overhead it generates. On the other hand, large intervals have the disadvantage of increasing the duration of the training phase. Therefore, the time interval between dummy messages (i.e., the interval ICA updates its counters) should be chosen taking into account that a large value extends the configuration phase but saves energy, while a small value expends more energy but shortens the configuration phase. For example, if each node sends one dummy message per second, then the time required for the configuration phase would be approximately 17 minutes only.

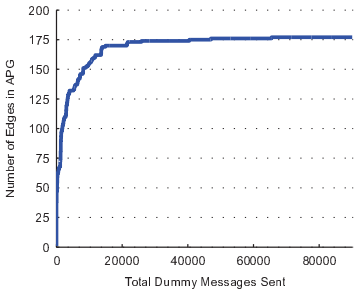

Figure 3. APG's number of edges convergence.

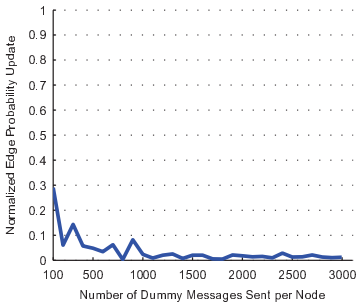

Figure 4. APG's edge probabilities convergence.

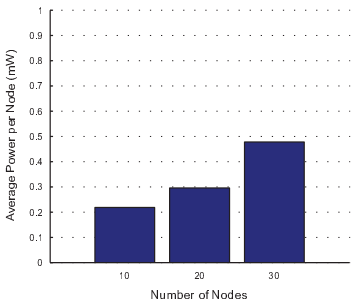

Figure 5. Node's average power consumption.

4.3. Energy Overhead in the WSN

Considering that WSN nodes are battery powered, this section evaluates the amount of power each information collection agent consumes to send $1,000$ dummy messages to its RPL-parents, maintain the counter variables, as well as send the counter messages to the base station in the end of the configuration phase.

In order to do so, the Powertrace [Dunkels et al. 2011] application provided by Contiki is used to measure the power consumption of different operation modes of sensors in terms of the number of clock ticks. The four typical operation modes are: low power mode, or LPM (MCU idle, radio off); CPU mode (MCU on, radio off); listen mode (MCU on, radio receiving); and transmit mode (MCU on, radio transmitting). The power consumption of a node is calculated as follows:

$$Power(mW) = (transmit \times 19.5\,mA + listen \times 21.8\,mA + CPU \times 1.8\,mA + LPM \times 0.0545\,mA) \times 3\,V\ /\ (32768 \times Time(s)) \quad (6)$$

Where $32,768$ is the number of clock ticks per second of Tmote Sky nodes, $Time(s)$ is the duration of the simulation (in seconds), and the current and voltage values (in mA and in V, respectively) have been obtained from the datasheet of Tmote Sky.

Fig. 5 shows the average power consumption per node (in mW) for three different sized networks, containing 10, 20, and 30 nodes, respectively. As demonstrated, the power consumed by ICA is $0.22\ mW$ in the 10-node WSN, while in the 30-node WSN ICA consumes $0.48\ mW$. Since the power values shown in Fig. 5 are only consumed during the configuration phase, not throughout the entire lifetime of the WSN, it can be concluded that the energy overhead added by ICA is fairly low.

Notice that those results reflect the worst case scenario, in which all $1,000$ dummy messages need to be sent. However, as discussed in the previous section, the time interval ICA updates its counters can be chosen so as to reduce or even eliminate the necessity of transmitting dummy messages.

4.4. Time required to estimate security metric values

As soon as the configuration phase is complete and the frequency DAG is generated, S3 performs three more steps (in the BS): (1) to convert the frequency DAG into an APG; (2) to translate the APG to a Bayesian network; and (3) to run Gibbs sampler on the BN to estimate the security level of all sensor nodes. Note that the first two steps only need to be executed once, while the third step is executed every time new IDS alerts are raised.

In this experiment, four different sized WSNs have been evaluated. The table in Fig. 6 shows the time required by steps (1) and (2) for each of those networks. As shown in the table, those times are negligible (less than $50\ ms$), for all simulated network sizes.

Because Gibbs sampler is a Monte Carlo-based statistical algorithm, it terminates when the number of sampling iterations it performs produces estimated probabilities that converge according to a given error threshold. Fig. 7 shows the time requirements for the Gibbs sampler (GS) inference procedure (step (3)) in the four evaluated WSNs, considering two distinct number of iterations commonly used in the literature [Raftery and Lewis 1992], namely, $2,000$ and $10,000$. As presented in the figure, in a small WSN comprised of 20 nodes, the security metric computation for all nodes takes 0.25 and 1.2 seconds, in the two respective number of Gibbs Sampler iterations evaluated. On the other hand, in a reasonably large network of 100 nodes, the inference time increases to 2.8 and 13.1 seconds, respectively.

In summary, the experiments have shown that for typical WSNs containing from 20 to 100 nodes, the times required to perform the steps of S3 are fairly acceptable. On the other hand, it is also important to highlight that because of the properties of Bayesian networks, the inference times may increase exponentially as the network size grows. However, a number of iterations as low as $2,000$ can be used to carry out BN inferences in a timely fashion (for larger networks), at the cost of providing slightly less accurate security metric values (as will be shown in the next section).

4.5. Accuracy of the Estimated Security Metric

Since Gibbs sampler is an approximation algorithm, this section evaluates how incorrect can be the security metric values estimated by S3 when compared to the actual security status of nodes. In particular, various scenarios were carried out where a worm propagates through the network infecting several nodes. During the experiments, the number of times each node has been compromised by the worm is counted. Then, the fraction of times each node is compromised is compared to the probability value provided by S3 for each node. The results of the comparisons are shown in terms of the absolute error of the estimated value, i.e., the absolute difference between the actual infection probability and

Number of Nodes	APG generation (ms)	BN generation (ms)
20	0.007	2
30	0.015	4
50	0.025	8
100	0.068	45

Figure 6. Time requirements of S3 steps.

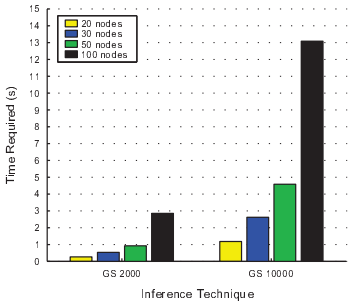

Figure 7. Time requirement of S3 inference technique.

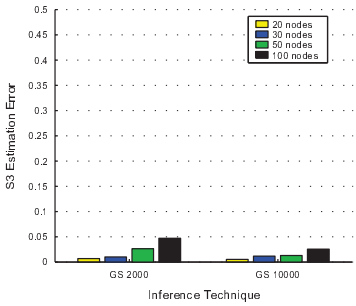

Figure 8. Error of S3 security metric estimation.

the infection probability estimated by S3. Fig. 8 presents those results for two distinct amounts of Gibbs sampler iterations and four different sized networks.

As shown in the figure, the average error in the estimated value is slightly smaller when the number of iterations is larger. Even in the worst case, the error value is fairly small, i.e., approximately 0.05 (for GS $2,000$ and 100 nodes). This error is even smaller (i.e., around 0.02) when $10,000$ Gibbs sampler iterations are performed.

In summary, the error values can be considered small for all network sizes and number of Gibbs sampler iterations. Hence, in the case of very large networks, it may be worth to decrease the number of Gibbs sampler iterations so as to improve performance (computation overhead), while still obtaining accurate S3 estimates.

5. Related Work

Most of the existing works on security quantification are focused on traditional networks. Those proposals are usually based attack graphs, which measure security based on the interdependency of system vulnerabilities (*e.g.,* the Weakest Adversary [Pamula et al. 2006] and the Attack Resistance [Wang et al. 2007] metrics). However, none of those proposals is suitable to quantify security in sensor networks due to the specific characteristics, vulnerabilities, and attack types of WSNs [Walters et al. 2007].

So far, only a few works have been proposed specifically for WSNs [Anand et al. 2005, Ramos and Filho 2015]. Anand et al. [Anand et al. 2005] propose a model that probabilistically quantify the resilience of WSN protocols against eavesdropping attacks. Their model is based on information such as sensor data distribution and topologies. However, their model is designed to evaluate security statically, rather than in an online manner.

An online security quantification scheme for WSNs has been recently proposed by Ramos et al. [Ramos and Filho 2015]. This scheme is based on three security metrics that respectively measure the resilience of the three main security mechanisms deployed in WSNs (*i.e.,* cryptography, key management, and IDS). Although this scheme addresses several attack types and considers IDS alerts, it treats attacks as independent events and, consequently, disregards worm attacks.

Finally, there exist some works [Haghighi et al. 2016, De et al. 2009] that use epidemic theory to model worm propagation in WSNs. Those proposals provide useful information that enable to understand worm attack behavior as well as to develop defensive

strategies. However, such works are not suitable for an online security evaluation since they are usually based on abstract input parameters (which are very difficult to obtain) and focus on the static analysis of the WSNs, rather than on the operational analysis.

Unlike the previous existing works, the proposed S3 model is a practical approach, which has been implemented and evaluated in a real WSN operating system. Furthermore, by using the Bayesian networks formalism, S3 is able to capture the dependency among the attacks that occur in different nodes in the WSN and evaluate node security level in an online manner. It should be noted that S3 has been mostly inspired by the security evaluation framework proposed in [Zonouz et al. 2015] for energy delivery systems.

6. Conclusion

This paper presented the Sensor Security Status (S3), a security metric model for estimating the security level of sensor nodes. Since WSNs are vulnerable to worm attacks, which can take advantage of the network communication pattern to spread throughout the network, S3 combines IDS alerts with the communication behavior of sensor nodes to estimate the probability that a node has been compromised given that other malicious nodes are present in the network. The presented simulation results show that S3 accurately represents node security level while keeping energy and performance overhead low.

Since IDSs may have false positives, an interesting future work would be to extend S3 to deal with IDS inaccuracies. This could be done by integrating S3 with the IDS effectiveness metric proposed in [Ramos et al. 2017]. Furthermore, considering that the WSN topology may change over time (e.g., new nodes can be added), it would also be interesting to develop an approach to periodically rebuild the APG to reflect the changes in the network communication graph while still keeping node's energy consumption low. Another future work is to consider other worm propagation strategies.

Acknowledgments

This work has been partially supported by Coordination for the Improvement of Higher Education Personnel (CAPES), Instituto de Telecomunicações, Next Generation Networks and Applications Group (NetGNA), Covilhã Delegation, by National Funding from the FCT - Fundação para a Ciência e a Tecnologia through the UID/EEA/500008/2013 Project, and by Finep, with resources from Funttel, grant no. 01.14.0231.00, under the Radiocommunication Reference Center (Centro de Referência em Radiocomunicações CRR) project of the National Institute of Telecommunications (Instituto Nacional de TelecomunicaçõesInatel), Brazil.

References

Anand, M., Ives, Z., and Lee, I. (2005). Quantifying eavesdropping vulnerability in sensor networks. In *ACM DMSN Wksp*, pages 3–9.

Casella, G. and George, E. I. (1992). Explaining the gibbs sampler. *The American Statistician*, 46(3):167–174.

De, P., Liu, Y., and Das, S. K. (2009). Deployment-aware modeling of node compromise spread in wireless sensor networks using epidemic theory. *ACM Trans. Sen. Netw.*, 5(3):23:1–23:33.

Dunkels, A., Eriksson, J., Finne, N., and Tsiftes, N. (2011). Powertrace: Network-level Power Profiling for Low-power Wireless Networks. Technical Report 4112.

Dunkels, A., Gronvall, B., and Voigt, T. (2004). Contiki - a lightweight and flexible operating system for tiny networked sensors. In *IEEE LCN Conf.*, pages 455–462.

Francillon, A. and Castelluccia, C. (2008). Code injection attacks on harvard-architecture devices. In *ACM CCS Conf.*, pages 15–26.

Giannetsos, T., Dimitriou, T., and Prasad, N. R. (2009). Self-propagating worms in wireless sensor networks. In *ACM Co-Next Student Wksp*, pages 31–32.

Haghighi, M. S., Wen, S., Xiang, Y., Quinn, B., and Zhou, W. (2016). On the race of worms and patches: Modeling the spread of information in wireless sensor networks. *IEEE Transactions on Information Forensics and Security*, 11(12):2854–2865.

Ho, J.-w. (2015). Distributed Software-Attestation Defense against Sensor Worm Propagation. *Journal of Sensors*, 2015:1–6.

Khayam, S. A. and Radha, H. (2005). A topologically-aware worm propagation model for wireless sensor networks. In *IEEE ICDCS Wksps*, pages 210–216.

King, D. (2002). Dlibc++. Available: `http://dlib.net`. Accessed: 2016-12-08.

Osterlind, F., Dunkels, A., Eriksson, J., Finne, N., and Voigt, T. (2006). Cross-level sensor network simulation with cooja. In *IEEE LCN Conf.*, pages 641–648.

Pamula, J., Jajodia, S., Ammann, P., and Swarup, V. (2006). A weakest-adversary security metric for network configuration security analysis. In *QoP Wksp*, pages 31–38.

Polastre, J., Szewczyk, R., and Culler, D. (2005). Telos: enabling ultra-low power wireless research. In *IPSN International Symposium*, pages 364–369.

Raftery, A. E. and Lewis, S. (1992). How many iterations in the Gibbs sampler. In *In Bayesian Statistics 4*, volume 4, pages 763–773.

Ramos, A. and Filho, R. (2015). Sensor Data Security Level Estimation Scheme for Wireless Sensor Networks. *Sensors*, 15(1):2104–2136.

Ramos, A., Lazar, M., Holanda Filho, R., and Rodrigues, J. J. P. C. (2017). A Security Metric for the Evaluation of Collaborative Intrusion Detection Systems in Wireless Sensor Networks. In *IEEE International Conference on Communications (Accepted for publication)*.

Raza, S., Wallgren, L., and Voigt, T. (2013). SVELTE: Real-time intrusion detection in the Internet of Things. *Ad Hoc Networks*, 11(8):2661–2674.

Walters, J., Liang, Z., Shi, W., and Chaudhary, V. (2007). *Wireless Sensor Network Security: A Survey*, page 367. CRC Press: Boca Raton, FL, USA.

Wang, L., Singhal, A., and Jajodia, S. (2007). Measuring the overall security of network configurations using attack graphs. In *DBSec Conf.*, pages 98–112. Springer.

Winter, T., Thubert, P., Brandt, A., Hui, J. W., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, J., and Alexander, R. (2012). RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. Technical Report 6550.

Zonouz, S. A., Berthier, R., Khurana, H., Sanders, W. H., and Yardley, T. (2015). Seclius: An Information Flow-Based, Consequence-Centric Security Metric. *IEEE Transactions on Parallel and Distributed Systems*, 26(2):562–573.