IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 30, 2020, accepted October 15, 2020, date of publication October 27, 2020,
date of current version November 10, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3034096

SGF-MD: Behavior Rule Specification-Based Distributed Misbehavior Detection of Embedded IoT Devices in a Closed-Loop Smart Greenhouse Farming System

PHILIP VIRGIL ASTILLO, JIYOON KIM, (Student Member, IEEE), VISHAL SHARMA, (Member, IEEE), AND ILSUN YOU, (Senior Member, IEEE)
Department of Information Security Engineering, Soonchunhyang University, Asan 31538, South Korea

Corresponding author: Ilsun You (ilsunu@gmail.com)

This work was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education (NRF-2020R1I1A2073603) as well as the Soonchunhyang University Research Fund.

ABSTRACT Smart farming is rapidly revolutionizing the agricultural sector where embedded Internet of Things (IoT) devices are integrated into the field to maintain or improve the quality of products as well as increase food production. Despite the tremendous benefits, various cybersecurity threats of IoT can also be inherited by the sector. In this paper, we propose a lightweight specification-based distributed detection to identify the misbehavior of heterogeneous embedded IoT nodes efficiently and effectively in a closed-loop smart greenhouse farming system. To expand the monitoring space of a node, we exploited the Kalman-filter algorithm and simple statistical operations to obtain estimates of data. Accordingly, this enables a monitoring node to assess a target node that has distinct physical characteristics and access to natural phenomena. Along with this, we derive the behavior-rules that are specific to the target system and carefully translate these rules into a state machine diagram. Besides, we formally verify the functional correctness of the monitoring processes as well as ensure that the behavior specifications are completely covered by using the model checker tool UPPAAL. Through extensive experimental simulation using Proteus, we verify its applicability to resource-constrained embedded devices, e.g., Arduino-Uno, as well as show high accuracy in detecting misbehaving nodes while having low false alarms.

INDEX TERMS Smart greenhouse farming (SGF), Internet-of-Things, cyber-agroterrorism, misbehavior detection, specification-based approach.

I. INTRODUCTION

The continuous increase of the world's population has proportionally increased the demand for sustainable food production. Humans have found ways to increase yield as well as improve the quality of agricultural products through research and integration of various technologies. One of the significant approaches is Smart Farming or Precision Agriculture [1].

Smart Farming is an emerging application of Information and Communication Technologies (ICT) together with the Internet-of-Things (IoT) to monitor crop, environmental, and soil conditions as well as control fertigation and irrigation systems [1]–[4]. It is composed of interconnected embedded devices with various sensors and actuators such as temperature, humidity, soil moisture sensors, motors, variable-rate sprayer, etc. and reports time-series data to a remote application that supports in the optimization of agricultural processes [5], [6].

Despite the tremendous benefits of integrating ICT and IoT in the agricultural sector, it is also facing various cybersecurity threats. This has exposed the sector to be targeted by agri-competitors and hacktivists with malicious intent [1]. Unfortunately, agri-companies or farmers paid very much little attention to protect the sector against cyberattacks [7]. Additionally, they still do not fully realize the implications even if the attacks are successfully carried out.

Among the different cyberattacks, the denial of service (DoS) and ''cyber-agroterrorism'' attacks are the empirically determined threats that are critical to Smart Farming. In the sector, DoS attacks degrade the availability of real-time data.

The associate editor coordinating the review of this manuscript and approving it for publication was Chunhua Su.

The availability of real-time conditions of the field is essential as crops are sensitive to the environment. For example, irrigation must be applied following the recommended soil moisture condition of the planted crops. The delayed availability of soil moisture data can lead to over-irrigation or under-irrigation which could degrade the quality of crops as well as decrease yield. Meanwhile, ''cyber-agroterrorism'' attacks target the confidentiality and integrity of data being reported by nodes. As such, attackers can effectively conduct national agri-tech espionage and successfully malign agri-producers, which results in the demotion of its reputation in the food supply chain [1].

Clearly, it is of paramount importance to protect smart farming systems from hostile attackers as it is rapidly revolutionizing the agricultural sector in addition to being faced with cyber threats. For such an aim, an intrusion detection system (IDS) can play a significant role in countering possible security threats and attacks. To be applied to Smart Farming, IDS must be as lightweight as possible considering that most IoT devices have limited energy, memory, and low computational capability [8]. Moreover, it is required to address zero-day attacks that can result from the difficulty for a timely smart update. Note that zero-day attacks can be regarded as unknown-attacks from the perspective of IDS.

Indoor farm projects such as vertical farming and greenhouse farming are currently taking over the global market [9]. Both are benefiting from smart farming solutions. Meanwhile, the security solutions introduced in this paper focus on the smart greenhouse farming (SGF) environment (See Section IIIc). However, it is worth noting that such solutions are also applicable to the vertical farming environment with minimal changes. Accordingly, this paper proposes a distributed behavior-rule specification-based misbehavior detection approach to detect misbehaving sensing nodes in SGF. Related work in [10] and [11] derived behavior-rules to monitor a target node. However, the rules were derived from the assumption that the monitoring nodes have the same sensing capabilities and access to the same phenomenon. This assumption is too strong since a multitude of embedded-IoT devices that have different tasks and sensing capabilities are integrated into SGF. Hence, if such a technique is applied holistically in the SGF environment, the monitoring capability of the nodes would be limited only to the nodes with the same physical characteristics and environmental access. For this reason, we introduce a data estimation strategy based on Kalman-Filter for us to expand the target space of a monitoring node. In other words, the monitoring node and its trusted target do not need to have the same sensors and observe the same phenomenon.

The main contributions of this paper are as follows:

(i) We derive the behavior rules of the SGF environment from given embedded system requirements.
(ii) We explore the Kalman Filter-based estimation algorithm to expand the monitoring space of monitoring IoT devices.
(iii) We express the behavior-rules as state machine diagrams based on the Unified Modeling Language (UML) to illustrate the monitoring process.
(iv) We formally verify the state diagrams to validate functional correctness and completeness of behavior rules.
(v) We develop an intrusion detection software agent that can run on the Arduino Uno microcontroller.
(vi) We evaluate our intrusion detection method using the Proteus simulation tool.

The remainder of this paper is organized as follows. Section II are surveys of related works on misbehavior detection in the IoT environment. Section III discusses the concept of Kalman-filter and other statistical operations. Section IV presents the derivation of behavior rules and the transformation of state diagrams, followed by the formal verification in Section V. Section VI describes the threat model. Sections VII and VIII present the details of our simulation and the obtained results, respectively. Finally, section IX concludes the paper and presents future works.

II. RELATED WORKS

The advent of IoT has raised security and privacy issues that need to be addressed. Intrusion Detection System (IDS) has acted as a second line of defense against inside and outside attackers not only to traditional computer networks but also to internet-connected sensory nodes [10]–[16]. It dynamically monitors the communication flow, state, and behavior of the system to find intruders and mitigate its effect on the system [17].

In general, there are three categories for intrusion detection, namely anomaly-based, signature-based, and specification-based techniques that are also being exploited in IoT.

Anomaly-based detection technique utilizes the normal operational profiles of the system to detect malicious activities or events. Machine learning techniques were considered as the most innovative method in anomaly detection [18]. Efstathopolous *et al.* [13] presented a comparative analysis of a supervised machine learning algorithm that includes One Class-SVM, Isolation Forest, Angle-Base Outlier (ABOD). Stochastic Outlier Selection (SOS), and Principal Component Analysis (PCA) for detecting an anomaly in the Smart Grid environment. Each algorithm was trained using the data obtained (temperature) from a power plant during normal operation. The authors show that utilizing the operational data and anomaly-based detection method can accurately detect cyberattacks or malicious events. Van *et al.* [19] and Liang *et al.* [14] also show high intrusion detection accuracy using the machine learning approach through deep learning algorithms.

Meanwhile, the signature-based or misuse detection technique relies on pattern matching over a database of signatures of every known system or network threats [20]. Accordingly, such an approach is weak in defending against ''zero-day attacks'' [15], [20]. To alleviate this weakness, this approach worked cooperatively with other techniques to have a more

robust IDS. Otoum *et al.* [21] proposed a hybrid technique using signature-based and anomaly-based to defend against known and unknown malicious behavior of sensory nodes, respectively. The designed architecture of the authors is composed of two co-operative subsystems such as spatial clustering and random forest methods, which resulted in a very impressive detection rate. Ozcelik *et al.* [16] has also proposed a hybrid approach where misuse detection method and trust calculation are combined to detect malicious sensor nodes in a large hierarchical clustered WSNs. In their approach, each node calculates its neighbor's five reputation values and are aggregated in a pre-selected cluster head node. The consolidated trust values are then forwarded to a based station and are matched against a misuse detection rules for the final decision if a node is misbehaving or not. Their approach has increased the network-lifetime of the nodes.

Additionally, several researchers have also proposed the specification-based IDS solution because of the empirical weaknesses of anomaly-based and signature-based techniques. The latter approaches and hybrid IDS are not appropriate to resource-constraint devices as it needs large memory storage or high computing power [15], [22]. Meanwhile, in the specification-based method, malicious behavior is detected through the behavior deviation of system from the expert-defined rules [23]. You *et al.* [10] and Sharma *et al.* [11] applied behavior-rules specification-based approach for misbehavior detection in healthcare IoT and UAV-IoT, respectively. The authors derived the behavior-rules from a given operation profile. However, the rules were derived with the assumption that the monitoring nodes have the same sensing capabilities and access to the same phenomenon as that of the target nodes. Therefore, the target space of the monitoring nodes is limited. This motivated us to research solutions to expand the monitoring space of a node.

III. BACKGROUND

This section describes the role of the Kalman Filter along with simple statistical operations in the detection of anomalous data points. It also describes our adopted system model for the smart greenhouse farm.

A. KALMAN FILTER ALGORITHM

Kalman filter was primarily intended to estimate the true value of a phenomenon using time-series data measured in a noisy environment [24]. It has a relatively simple operation and does not need high computational power and storage, thus, making it compatible with resource-constrained IoT devices [23], [24]. Kalman filter is an iterative process where each iteration is divided into two steps: projection and correction. In the projection stage, a priori estimate state and priori estimate co-variance are obtained as in eq. 1 and 2, respectively, where A is the transition matrix, B is the control-input matrix that is multiplied to the control vector c_t, and Q is the process noise covariance matrix. In this paper, c_t is attributed as a knowledge-based rate of change of the target phenomenon such as temperature and soil moisture level. This means that c_t is equal to the phenomenon's change of value per second ζ multiplied by the time interval between two calculation instances.

$$\hat{\boldsymbol{x}}^t_{t-1} = A\hat{\boldsymbol{x}}_{t-1} + Bc_t \tag{1}$$

$$P^t_{t-1} = AP_{t-1}A^T + Q \tag{2}$$

$$c_t = \zeta(t_2 - t_1) \tag{3}$$

Subsequently, the correction stage computes the optimal Kalman Gain, a posteriori estimate state, and posteriori estimate covariance as

$$K_t = \left(P^t_{t-1}H^T\right)\left(HP^t_{t-1}H^T + R\right)^{-1} \tag{4}$$

$$\hat{\boldsymbol{x}}_t = \hat{\boldsymbol{x}}^t_{t-1} + K_t\left(z_t - H\hat{\boldsymbol{x}}^t_{t-1}\right) \tag{5}$$

$$P_t = (I - K_tH)\,P^t_{t-1} \tag{6}$$

respectively, where H is a context filtering matrix, z_t is the measured data at time t, and R is the measurement noise matrix. Time-varying values of K_t along with co-variance P_t filters out the effect of noise and quickly converges to the more accurate representation of data.

Kalman filter is vulnerable to measurement values or behaviors that are called outliers, which were not considered in the model. The influence of outliers in the model can be minimized by associating the measurement noise R as a tuning parameter. As shown in eq. 4, a higher measurement noise parameter produces a small Kalman gain. Consequently, the compensation term for the posteriori estimate will also be small. In this paper, we adopt Mahalanobis distance approximation MD and local weighting function from [27] to obtain the new tuning parameter R as described in eq. 7 and 8. This method minimizes the effect of outliers.

$$MD = \sqrt[2]{\left(z_t - \hat{\boldsymbol{x}}^t_{t-1}\right)^2\left(HP^t_{t-1}H^T + R_{t-1}\right)^{-1}} \tag{7}$$

$$R_t = \left(1 + e^{-MD}\right)^{-1} \tag{8}$$

B. OUTLIER CLASSIFICATION OF TARGET PHENOMENON

Kalman filter algorithm can be extended to classify outliers in the observed data by exploiting the estimated value for statistical inference. In this paper, a simple statistical operation such as mean and standard deviation of the distribution of root squared errors (RSE) between the observed data and corresponding posteriori estimate are used as the judging criterion in the classification of the anomalous data point. Utilizing the results from Kalman filter estimation, the RSE is described in Eq. 6.

$$D_t = \sqrt[2]{\left(z_t - \hat{\boldsymbol{x}}_t\right)^2} \tag{9}$$

During the normal state of the system, the sequence of the error $[D_{t-n}, \ldots, D_{t-1}, D_t]$ has a mean and standard deviation until time t that can be described as,

$$\mu_t = \left(\sum\nolimits_{i\to t} D_t\right)N_t^{-1} \tag{10}$$

$$\sigma_t = \sqrt[2]{\sum\nolimits_{i\to t}(D_i - \mu_i)^2\,N_t^{-1}} \tag{11}$$

respectively, where N is the number of data points at time t. The variation of the computed errors in the normal state falls under a confidence interval based on the mean and standard deviation of the historical distribution. Otherwise, it indicates that there is an anomalous change in the condition of the system. In this paper, we integrate these techniques in the formulation of the Boolean conditions for a behavior-rule, that is introduced in the next section, where the physical variables involve sensor readings.

Outliers are data points that significantly deviate from other observations. For example, an observable characteristic of a microenvironment's temperature is that it increases or decreases gradually over a period. A steep change in sensor reading in a short interval is considered anomalous. It could be caused by disturbances during the measurement or in the context of data communication, it could be a result of data tampering. To capture an anomalous data point, the observed data at time t is first processed using the Kalman filter method to get its corresponding estimated value as presented in Algorithm 1.

Algorithm 1 Kalman Filter-Based Estimation

Input: $z_t, x_{t-1}, c_t, P_{t-1}, R_{t-1}$
Output: x_t, P_t, R_t
(1) Calculate priori estimate $\hat{\boldsymbol{x}}_{\boldsymbol{t-1}}^{\boldsymbol{t}}$ and priori co-variance $\boldsymbol{P}_{\boldsymbol{t-1}}^{\boldsymbol{t}}$ using eq. (1) to (3)
(2) Calculate Mahalanobis MD and update tuning Parameter R_t using eq. (7) and (8)
(3) Calculate Kalman gain $\boldsymbol{K_t}$, posteriori estimate $\boldsymbol{x_t}$, and posteriori co-variance $\boldsymbol{P_t}$ using eq. (3) $\sim$ (5)

Subsequently, the estimated value is utilized for further classification of an observed data point as an outlier or not using Eq. 9 as the judging criterion, where β_c is the performance parameter. Now instead of assigning a fixed value for β_c from the start of deployment, the system learns the optimal value based on gradient descent during the early period γ of deployment. The update function for β_c is described in eq. 10 and 11 with a learning rate of α_R. Now, so that the update of β_c will be sensitive to the level of noise, we set α_R to R_t from eq. 8. As shown in Algorithm 2, the update happens when the current error D_t falls beyond the current limit criterion LM_t.

$$LM_t = \mu_t + \beta_c \sigma_t \tag{12}$$

$$\beta_{c^+} = \beta_c + \alpha_R \frac{\partial J(\beta)}{\partial \beta} \tag{13}$$

$$J(N) = (D_t - (\mu_t + \beta \sigma_t))^2 \tag{14}$$

C. SMART GREENHOUSE ECOSYSTEM

A greenhouse farm is an indoor farm where the structure is made from transparent materials in which micro-climate conditions are continuously monitored and controlled to provide a favorable growing environment for crops or plants in general. Real-time monitoring and control of the climate condition within a greenhouse farm are critical as it can affect the quality and yield of the planted crops. A closed-loop control and management system composed of embedded-IoT devices integrated with sensors and actuators is currently adopted, consequently made the greenhouse environment smarter [28].

Algorithm 2 Outlier Classification of Target Phenomenon

Input: $z_t, x_t, DT_{t-1}, DE_{t-1}, N_{t-1}, \beta_c$
Output: V = True/False(0/1), DT_t, DE_t, N_t, β_U
(1) Calculate Euclidean distance D_t using eq. (6)
(2) Calculate the following:
(3) $\boldsymbol{N_t} = N_{t-1} + 1$; $\boldsymbol{DT_t} = DT_{t-1} + D_t$
(4) $\boldsymbol{\mu_t} = DT_t/N_t$; $\boldsymbol{DE_t} = DE_{t-1} + (D_t - \mu_t)^2$
(5) $\boldsymbol{\sigma_t} = \sqrt[2]{DE_t/N_t}$
(6) Calculate limit criterion LM_t using eq. (12)
(7) If $(D_t \leq LM_t)$
(8) Assign $V = True$
(9) Else
(10) If $(N_t \leq \gamma)$
(11) Update β_c using eq. (12) $\sim$ (14)
(12) Assign V = *True*
(13) Else
(14) Assign V= *False*
(15) $\boldsymbol{DT_t} = DT_{t-1}$; $\boldsymbol{N_t} = N_{t-1}$
(16) $\boldsymbol{DE_t} = DE_{t-1}$

Consequently, the integration of IoT in this sector helped optimize growth and production [29]. On the other hand, it has placed this sector to face inherent challenges related to cybersecurity.

An example of a smart greenhouse ecosystem, which we considered is given in Figure 1. IoT nodes in this greenhouse farm have a vital role in monitoring the internal temperature, soil moisture, irrigation flow state, etc. and respond accordingly by controlling actuators such as motors, sprayers, etc. As illustrated in the figure, an IoT node may have a unique task, hence it is equipped with only the appropriate sensors and actuators to achieve it. Additionally, nodes are placed strategically to monitor a certain area on the farm. Thus, each node may behave differently from other nodes for it is possible that the crop planted on its monitored area is different from the other areas. Furthermore, the nodes communicate wirelessly to a base station where all data are aggregated and utilized by software analytics.

In this paper, we proposed a misbehavior detection based on distributed monitoring. Our approach allows IoT nodes to monitor the behavior of other nodes in proximity, regardless of whether they are equipped with the same set of sensors and actuators or not, by utilizing the Kalman filter algorithm and the statistical attributes introduced in Sections IIIA and B. In addition, the effectiveness of the explored methods relies on a reasonable assumption that the system behaves well at the early phase of its operation and misbehavior manifest later. The target node periodically reports its physical parameters (heartbeat message), e.g., temperature readings, to the monitor nodes in proximity. Subsequently, the monitor nodes

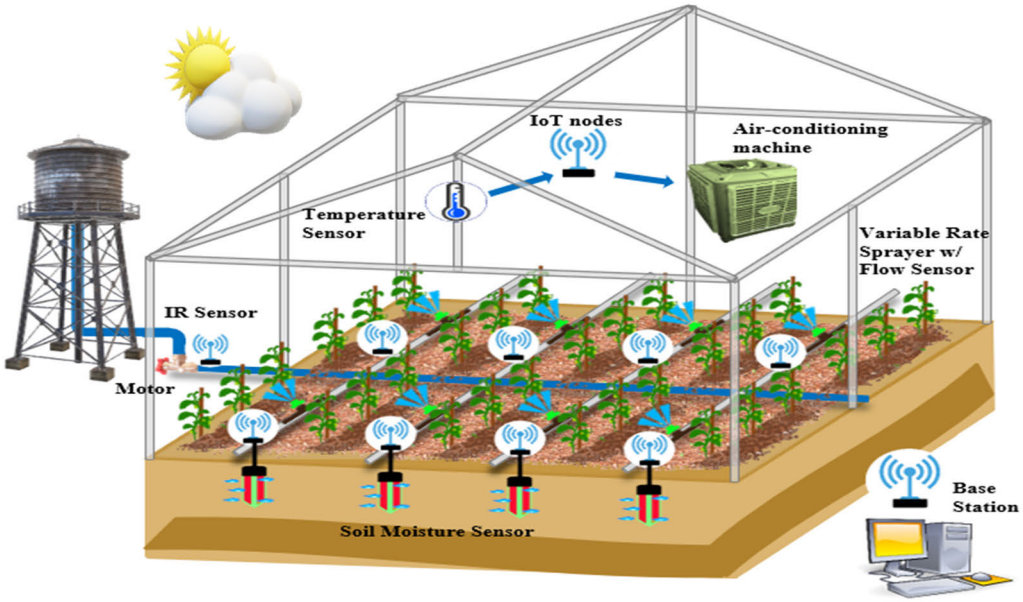


FIGURE 1. An example of smart greenhouse ecosystem.

evaluate the received operational data and network function if it deviates from the specified behavior of a node. When they find a misbehaving node, they can immediately send an alarm to designated personnel for mitigation.

IV. BEHAVIOR-RULE SPECIFICATION OF SMART GREENHOUSE FARM

Behavior-rules are checklists that an IDS software agent used as a basis in determining the state of the trusted node. Violation of such a rule implies that the system is misbehaving. This section presents the derivation of the behavior-rule of the sample smart greenhouse environment. The steps adopted in the derivation of behavior-rules are inspired by [11].

A. DERIVATION OF BEHAVIOR-RULES

The behavior-rules manifest the expected behavior of a target system; hence, it is a more systematic approach if these rules are derived based on the embedded system requirements specification which embodies the features and behavior of a system or hardware presented in a much more extensive and rigorous way. In addition, the specification enables embedded system designers to plan well its design and identify vital data assets as well as physical components needed to achieve the requirements [30]. In this paper, we assume that the available embedded system requirements of our target domain are listed in Table 1. ESR1 to ESR5 are the base requirements which target the main objective of monitoring the microenvironmental condition and automating the operations within the greenhouse farm. Furthermore, ESR6 is added to support the misbehavior detection task of the device.

Before the formulation of the behavior rules, we first enforce a security context on each given ESR by defining the corresponding security requirements as shown in Figure 2. The enforcement of security features puts a stringent perspective on the side of the monitor node as to how a trusted node should behave. Subsequently, the potential threats associated with a specific security requirement are identified. The threat refers to possible incidents that are intentionally conducted to prevent the system from performing its main task. As presented in Figure 2, a specific security requirement could have multiple threats that influence the node to fail. Thereafter, the behavior-rules are derived to defend against the identified threats. Note that a field expert's knowledge is important in its derivation for they know the possible cause or source of threats. Such derivations are guided by the security aspects such as integrity, confidentiality, and availability. Accordingly, the derived behavior-rule set is the basis for the behavior classification of nodes.

B. EXPRESSING BEHAVIOR-RULES AS STATE MACHINE DIAGRAMS

After identifying the behavior-rules (BR), we express them as state machine diagrams that are based on Unified Modeling Language (UML), to illustrate the processes of monitoring the target nodes' finite-state sequences operating at the monitor nodes. First, we divide the behavior-rules into three categories such as "*network-related behavior*", "*operational-related behavior*" and "*compliance-checker*". To draw the state diagrams, we assigned the different behavior-rules to the appropriate category. Accordingly, BR1-BR3 constitutes the network-related behavior, operation-related behavior comprises BR4 – BR8, and BR9 falls under compliance-checker. Afterward, we add at the appropriate state the attack state indicator (ASI), which

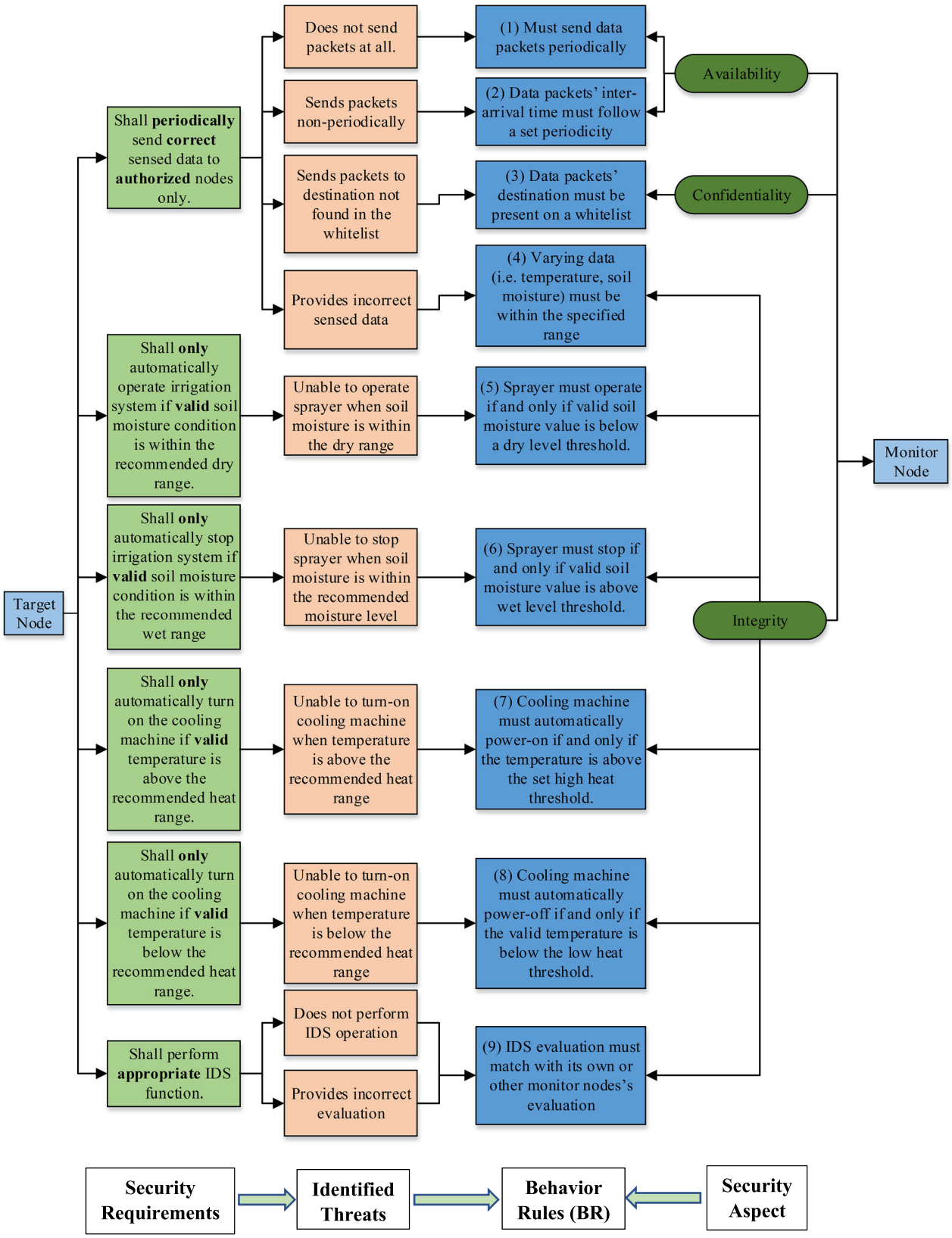


FIGURE 2. The adopted workflow on the derivation of behavior rules specific to the smart greenhouse environment.

TABLE 1. Embedded system requirements definition (ESR) for smart greenhouse monitoring and automation.

ESR #	System Requirements	Data Asset	Physical Components
ESR1	The IoT node/s should send operational data to the base station and other designated destination.	• Operational Data (See below)	• Wireless Technology (Wi-Fi)
ESR2	The IoT node/s should automatically operate an irrigation system at its assigned area when the soil moisture condition is lower than the recommended value of the planted crops.	• Soil moisture data • Water flow data • Recommended soil moisture of planted crops	• Soil moisture sensor • Water flow sensor • Variable Rate Sprayer
ESR3	The node/s should automatically cut-off irrigation system when soil moisture condition has met the recommended value for the planted crops.	• The same in ESR3	• The same in ESR3
ESR4	The IoT node/s should automatically turn-on cooling machine if the temperature within the greenhouse is higher than the recommended value for the planted crops.	• Temperature data • Recommended temperature of planted crops • Power supply state	• Temperature sensor • Power controller and sensor
ESR5	The IoT node/s should automatically turn-off cooling machine if the temperature within the greenhouse is lower than the recommended value for the planted crop.	• The same in ESR4	• The same in ESR4
ESR6	An IoT node within the smart greenhouse network infrastructure may be able to monitor behavior of other nodes.	• Evaluation Result	• Embedded platform (Arduino, Raspberry-Pi, etc.)

holds information if a corresponding behavior-rule is violated or not. The ASIs are Boolean variables that hold a value either True (1) or False (0) where the binary outcome 1 simply means that the node breaks the corresponding rule. Figure 3 shows the network-related state diagram, whereas Figures 4 and 5 describe the operational-related behavior of nodes handling temperature and soil moisture data, respectively, while Figure 6 illustrates the processes of compliance-checker. Additionally, functions that are imperative for the monitoring task are also included in the formulation of the state diagram. Consequently, the drawn state diagrams can be the basis for the development of the IDS software agents operating at the monitoring nodes.

The *network-related* state diagram shows the process of monitoring the network throughput of a target node. If the monitor node receives the first byte of the first heartbeat packet, the initial state transits to the Receive Data state. Subsequently, it will stay in that state until a heartbeat packet has been completely received or a waiting period expires. In the case of the latter, it immediately transits to the Update Do Time state. On the other hand, if the data packet is completely received within the expected waiting duration, the current state transits to the *Arrival Validation* state. In this state, the packet's inter-arrival time is then checked if it complies with the predefined average periodicity. The packet's destination is also verified if it is included on a whitelist (a list of legitimate receivers). Afterward, the state transits to *Update Do Time* and then returns to *Receive Data* state.

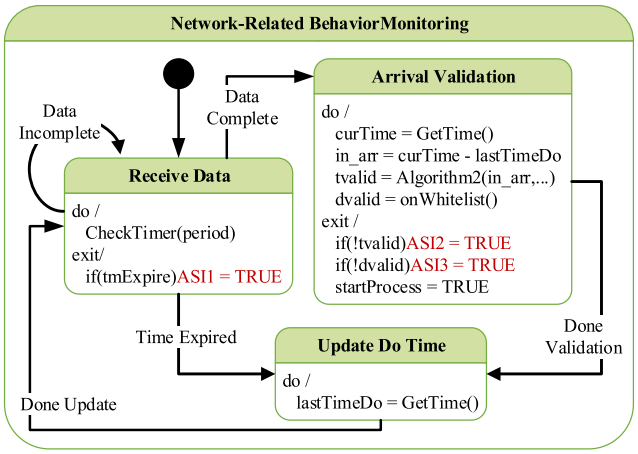


FIGURE 3. Network-related state machine diagram.

As shown in Figure 3, the state diagram evaluates three attack state indicators that correspond to the first three behavior-rules. The details of the three ASIs are as follows:

- ASI 1 evaluated as *True* (1) indicates that a target node has delayed in sending a complete heartbeat packet than the required periodicity. This attack case is captured

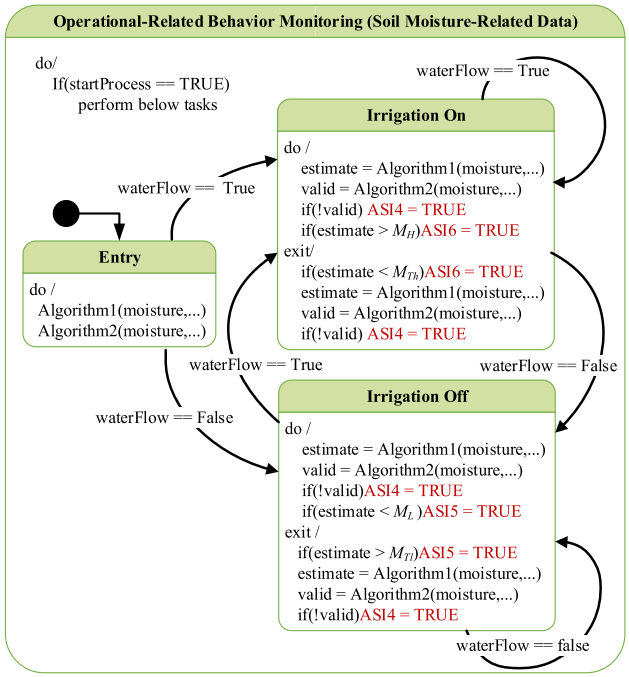


FIGURE 4. Operational-related state machine diagram handling soil-moisture data and irrigation system.

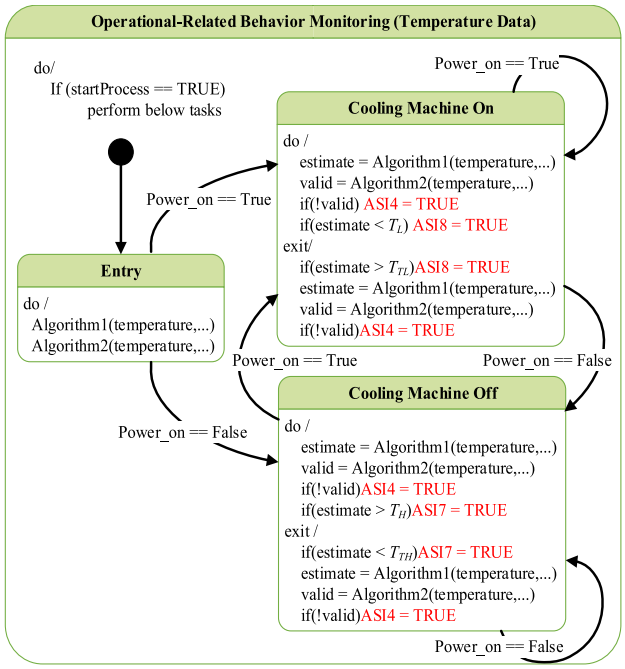


FIGURE 5. Operation-related state machine diagram handling temperature data and cooling machine.

when the preset waiting period during the Receive Data state expires.

- ASI 2 evaluated as *True* (1) indicates that the target node has sent a complete heartbeat packet too early than the required periodicity or it is successively sending heartbeat packets. This attack case is captured by using algorithm 2 where z_t and x_t are set as the inter-arrival between adjacent packets and required periodicity, respectively.
- ASI 3 evaluated as *True* (1) indicates that the target node is transmitting a heartbeat packet to an unauthorized node. This attack case is detected by checking if the destination is included on the list of authorized nodes called a whitelist. This list may be distributed during initialization (at power-up) and may be updated at runtime.

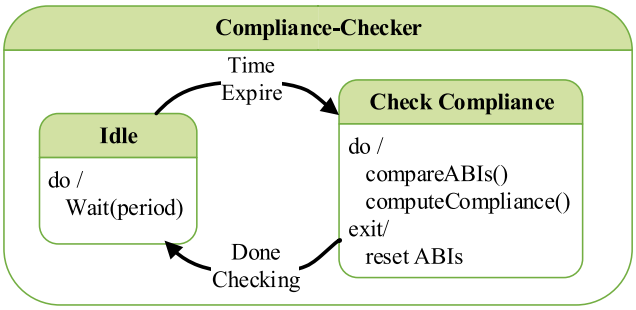


FIGURE 6. Compliance-Checker state diagram.

Furthermore, the operational-related state diagrams show the process of monitoring the main task of a target node. In our target scenario, there are two types of nodes: (1) a node's main task is to monitor and control the micro-temperature within the greenhouse farm, and (2) a node's main task is to monitor and control the soil-moisture level within its area. Hence, we have drawn separate state diagrams for each target node. If a monitor node receives the first complete heartbeat packet, the initial state transits to the Entry state. In this state, it simply computes the first estimate and initial statistical attributes using algorithms 1 and 2, respectively. Afterward, it immediately transits to the next state depending on the status of the associated actuators as illustrated in Figures 4 and 5.

As presented in Figures 4 and 5, the operational-related state diagrams evaluate the attack state indicators of behavior-rules 4 – 8. The details of the ASIs are as follows:

- ASI 4 evaluated as True (1) indicates that the operational data (temperature or soil-moisture) received by the monitor node significantly deviates from the model. The possible cause of this is that the packet may have been tampered with while on transit to the monitor node or the physical setup was intentionally and maliciously altered. This attack case is detected through algorithms 1 and 2 which monitor the historical distribution of data.
- ASI 5 evaluated as *True* (1) indicates that a target node is unable to activate the variable-rate sprayer even though the soil the moisture level has already crossed under the predefined extreme limit $\boldsymbol{M_L}$. In addition, it may also indicate that the target node has activated the sprayer where moisture level has not yet exceeded the activation value $\boldsymbol{M_{TL}}$ such that $\boldsymbol{M_{TL} > M_L}$. This attack case is

captured by checking the computed estimate of sensed soil moisture data within the *Irrigation off* state if it satisfies the given Boolean condition shown in Figure 4.

- ASI 6 evaluated as *True* (1) indicates that a target node is unable to stop the sprayer despite that the soil moisture level already falls over the predefined extreme limit $\boldsymbol{M_L}$. Additionally, it may also indicate that the target node has stopped the sprayer earlier where the moisture level is still below $\boldsymbol{M_{TH}}$ such that $\boldsymbol{M_H > M_{TH}}$. In contrast to ASI 5, this attack state is captured by checking the computed estimate of sensed moisture data within the *Irrigation on state* if it satisfies the given Boolean condition shown in figure 4.
- ASI 7 evaluated as *True* (1) indicates that the target node is unable to turn on the cooling machine even though the temperature is already over the extreme limit $\boldsymbol{T_H}$. Likewise, it may also indicate that the target node has prematurely turned-on the cooling machine. Thus, the temperature has not yet exceeded the trigger limit $\boldsymbol{T_{TH}}$ such that $\boldsymbol{T_H > T_{TH}}$. This attack case is captured by checking the estimated temperature within the *Cooling Machine Off* state if it satisfies the given Boolean condition shown in Figure 5.
- ASI 8 evaluated as *True* (1), not only it indicates that the target node is unable to turn off the cooling machine despite that the temperature has already fallen under the extreme limit $\boldsymbol{T_L}$ but also it has prematurely turn-off the machine. The latter means that temperature is still above $\boldsymbol{T_{TL}}$ such that $\boldsymbol{T_{TL} > T_L}$. This attack case is captured by also checking the estimated temperature data within the *Cooling Machine On* state if it satisfies the given Boolean condition shown in figure 5.

Meanwhile, the *compliance-checker* state diagram shown in figure 6 describes the process of evaluating the compliance of the target node based on the obtained values of ASI 1-8. Under this category, the attack state indicator for BR 9 is also assessed. BR 9 considers the case where the target node also acts as a monitor node. Hence, an ASI 9 evaluated as *True* (1) indicates that the target node provides a false evaluation towards a well-behaving node (called bad-mouthing attack) or a misbehaving node (called ballot-stuffing attack). This can be captured by checking the discrepancy of the target node's evaluation with its assessment towards the other nodes that they both monitors.

V. FORMAL VERIFICATION THROUGH MODEL CHECKING

While state diagrams are formed to illustrate the process of monitoring the reactive behavior of the target node if it is following the expected state transitions guided by the derived behavior-rules, it can also be the foundation for embedded software development. To verify the functional correctness and guarantee that behavior rules are completely covered before software development, formal verification is necessary.

In this paper, we utilized an integrated environment tool for system modeling, simulation and verification called UPPAAL [31]. UPPAAL has a user-friendly graphical interface that allows users to easily model systems as timed automata and assess the required specifications by representing it as computation tree logic (CTL) formulas. The model checker uses state-space exploration to determine whether a certain path to an automaton (node) exists or not during run time. Such features can be checked by defining a CTL formula that asserts the reachability property. Accordingly, we represent each state of the UML diagram as an automaton and drives its transition to the next state by synchronous signals and guard conditions as shown in Figure 7. However, due to the floating-point limitation of UPPAAL, we assume that algorithm1 and algorithm 2 are already established. This means that the operational data such as temperature, soil moisture level, etc. that are used in the guard conditions are the output of the two algorithms. As presented in the figures, we also represent the different ASIs as an automaton to assess the reachability property during normal and malicious events. In this form, the composition of the CTL formulas for the target property would make it very simple. Hence, the query on such property towards the ASI_x node provides a conclusive proof of correctness and completeness in the assessment of behavior rules. Other properties, such as safety, i.e., deadlock, can also be associated with the functional correctness ensuring that the system will not commit deadlock both in normal operations and malicious events. Note that the model for the operational-related task that handles soil-moisture data is like Figure 7c but only differs on the guard conditions.

Table 2 summarizes the verification results of our target properties. The composed CTL formulas assert the ASI node's reachability when a malicious event occurs. As shown in the table, a safety property is satisfied, which indicates that the process will not be deadlocked during the whole operation of the system. Meanwhile, when a reachability property is verified with a normal target node in place, all queries, $\boldsymbol{P_2}$ to $\boldsymbol{P_8}$, are not satisfied. This result is positive since normal nodes do not deviate from their specified behavior. Hence, there will be no path to the ASIs automaton. In contrast, when a malicious model is tested, all queries are satisfied. Thus, we can conclude from these results the proof of the correctness of our monitoring operations and the complete evaluation of behavior rules. Moreover, the correctness and completeness features can be associated as a satisfaction to the security requirements of the system.

VI. THREAT MODEL

A. FIRMWARE TROJAN

A smart greenhouse, or smart agriculture in general, is highly integrated with interconnected microprocessor-based embedded devices that perform sensing, actuation, and communication functionality. As such, due to the globalization of embedded system development, a malicious individual can insert a firmware Trojan at any phase of the fabrication

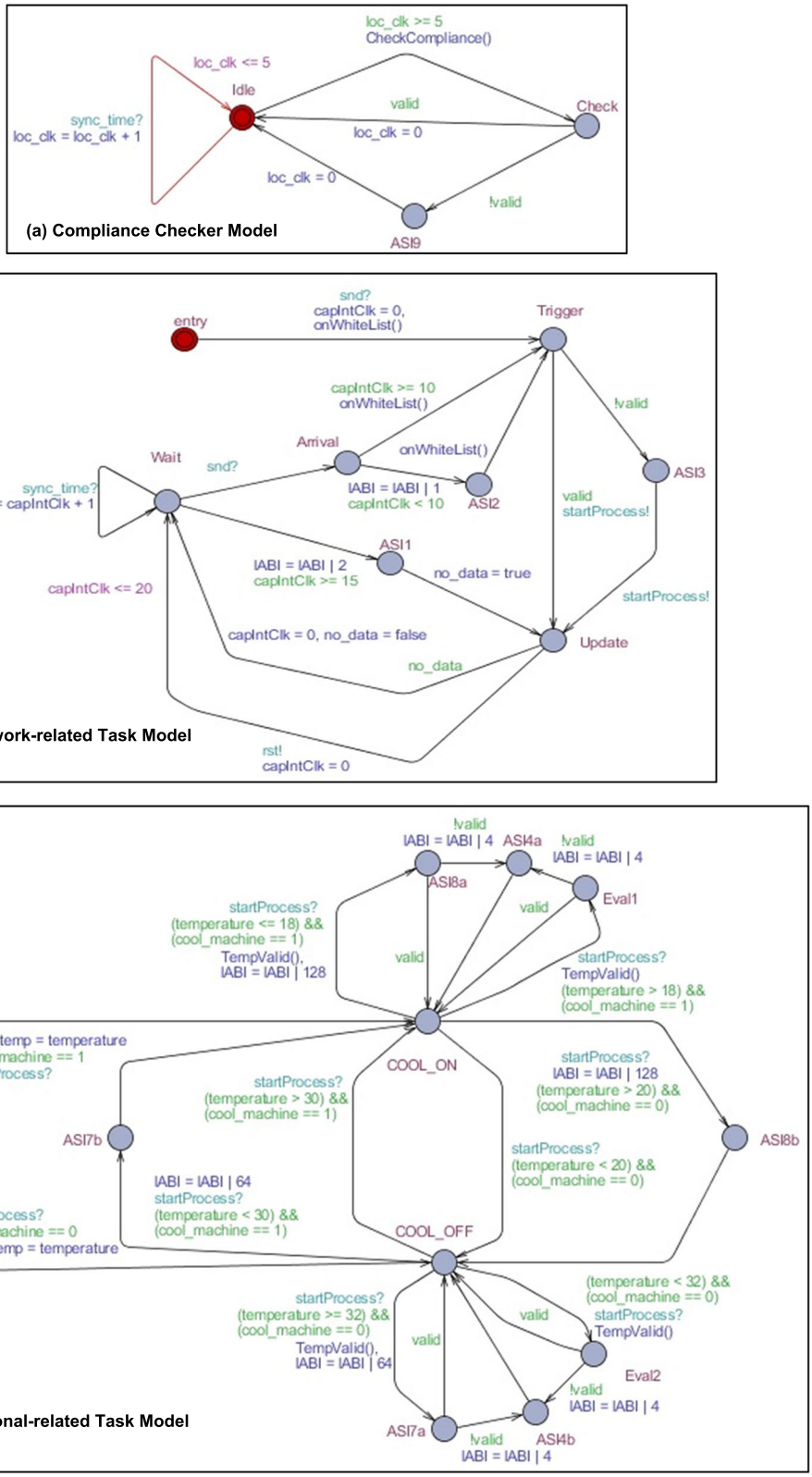


FIGURE 7. UPPAAL model of the different modules of IDS task. The operational-related task model for soil moisture is the same to (c) with minor changes on the guard conditions.

process [32]. In addition, the trojan may also be realized during the firmware update where the malicious code was successfully inserted into the updated version. Firmware trojans, also called hardware trojans, are designed to disable, manipulate functions, or degrade the performance of embedded devices in some future. In other words, it stays dormant

TABLE 2. Summary of queries to the requirement specification (legend: S = safety; R = reachability; N = normal; M = malicious).

ID	Property	Type	Query (UPPAAL)	Satisfied? (N)	Satisfied? (M)
P_1	The system is deadlock free	S	A[] not deadlock	Yes	Yes
P_2	The data receiver module eventually reaches ASI1	R	E<> Network.ASI1	No	Yes
P_3	The data receiver module eventually reaches ASI2	R	E<> Network.ASI2	No	Yes
P_4	The operation-related IDS task eventually reaches ASI4a or ASI4b	R	E<> Temperature.ASI4a E<> Temperature.ASI4b E<> Moisture.ASI4a E<> Moisture.ASI4b	No	Yes
P_5	The operational-related IDS task eventually reaches ASI5a or ASI5b (soil moisture data)	R	E<> Moisture.ASI5a E<> Moisture.ASI5b	No	Yes
P_6	The operational-related IDS task eventually reaches ASI6a or ASI6b (soil moisture data)	R	E<> Moisture.ASI6a E<> Moisture.ASI7b	No	Yes
P_7	The operation-related IDS task eventually reaches ASI7a or ASI7b (temperature data)	R	E<> Temperature.ASI7a E<> Temperature.ASI7b	No	Yes
P_8	The operation-related IDS task eventually reaches ASI8a or ASI8b (temperature data)	R	E<> Temperature.ASI8a E<> Temperature.ASI8b	No	Yes
P_8	The compliance checker task eventually reaches ASI9	R	E<> Compliance.ASI9	No	Yes

within the chip's physical layout until some external or internal activation condition, like counters or events, is met [33]. It is for this reason that firmware trojans are difficult to detect during the testing phase. Furthermore, the attack mode of a trojan may either be always-on or randomized.

B. FLOODING ATTACK

Flood attack, also known as Denial of Service (DoS) attack, involves sending a massive volume of packets to a target victim in order to exhaust its resources and disrupts the processing of other legitimate data. In this paper, we see this attack as one of the direct effects of the firmware trojan squatting at the wireless communication module's system-on-chip or at the main module's microcontroller. We focus on this attack in such a way that valid messages are repeatedly transmitted in a short interval.

C. DATA TAMPERING ATTACK

Data tampering is one of the major threats in many IoT-assisted systems. In this case, the adversary manipulates data with the intent not only to mislead the receiver on the condition of the system or device being monitored but also to inflict an adverse effect to applications or business processes that strongly relies on the analysis of these data.

This attack may either be originated internally or from an external entity. The former could also be a direct effect of firmware trojan, while the latter is achieved through a man-in-the-middle attack. In either case, we establish that an adversary modifies the data (i.e., temperature and soil-moisture) by adding an offset. Thus, the data received by the monitor node can be described in eq. 12, where ∇d is the deviation of sensed data due to noise at different levels (low, moderate, high) and l_d is the malicious offset.

$$m_t = m_{rt} \pm \nabla d \pm l_d \tag{15}$$

VII. EXPERIMENTAL VALIDATION THROUGH SIMULATION

To validate our proposed specification-based misbehavior detection, we conducted an experiment following the workflow illustrated in Figure 7. We first construct our simulation environment, followed by the development of the firmware which is composed of two tasks (Main task and IDS task). In this case, we adopt the agile software development method. The remainder of this section discusses in detail our experiment.

A. SIMULATION ENVIRONMENT

To proceed with this study, a simulation scenario of a smart greenhouse system that is composed of embedded IoT devices was constructed using Proteus Design Suite (Proteus) simulation tool. This tool was primarily used to design electronic schematics but later extended to simulate electronic operations covering from analog to micro-controller based embedded system applications. Hence, firmware developed for microcontrollers supported by this tool can be tested and

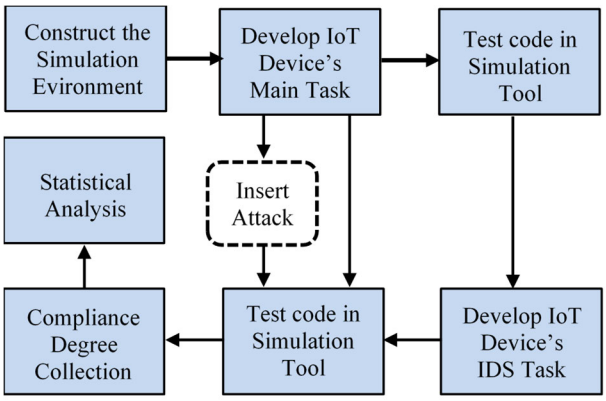


FIGURE 8. Experimentation workflow for the validation of the proposed misbehavior detection.

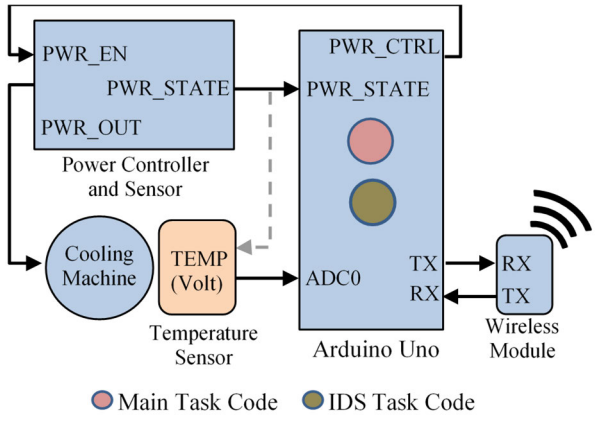


FIGURE 9. Block diagram of IoT device that monitors the temperature and controls the power supply for cooling machine.

co-simulated with other electronic components connected to it.

Figures 8 and 9 show the block diagram of two IoT devices found in the target smart greenhouse system. Note that the broken arrows are not included in the real schematic of the system and are only intended to assist the connected components in the simulation of its modeled behavior. In this case, we use the Arduino Uno as the main controller and connect the appropriate sensors and actuators. The controllers are uploaded with the main task code as well as the IDS task code. Moreover, we model the behavior of the moisture level and temperature using eq. 12 with $\boldsymbol{l_d}$ equal to zero. In accordance with the status of the actuators (i.e., sprayer, cooling machine), we also model the change of $\boldsymbol{m_{rt}}$ as,

$$\boldsymbol{m_{rt+1}} = \begin{cases} \boldsymbol{m_{rt}} - \boldsymbol{\Delta c}, \boldsymbol{flow_{state}} = 1, \boldsymbol{pwr_{state}} = 0 \\ \boldsymbol{m_{rt}} + \boldsymbol{\Delta c}, \boldsymbol{flow_{state}} = 0, \boldsymbol{pwr_{state}} = 1 \end{cases} \tag{16}$$

where $\boldsymbol{m_{rt}}$ is the base voltage output of soil moisture sensor or temperature sensor and $\boldsymbol{\Delta c}$ as the rate of voltage change per 5 seconds. Table 3 summarizes the values of the different parameters used during the simulation.

B. COMPLIANCE DEGREE COLLECTION

During the simulation runtime, the compliance degree of each target node is computed at the end of every unitformed-aggregation time-space $\boldsymbol{A_t}$. To minimize energy consumption, we divide $\boldsymbol{A_t}$ by n-uniform discrete intervals where the monitor nodes check the value of the attack state indicators which reflects the behavioral-state of the target node. Accordingly, at the end of each aggregation time space, the compliance degree is computed as described in eq. 14, where $\boldsymbol{B_r}$ is the number of times the node is in the well-behaved state. Now, at the end of mth aggregation time, a compliance degree history $\boldsymbol{c_1}, \boldsymbol{c_2}, \boldsymbol{c_3}, \ldots, \boldsymbol{c_m}$ are collected.

$$\boldsymbol{p_x} = \boldsymbol{B_r n^{-1}} \tag{17}$$

C. ATTACK INSERTION AND ATTACKER TYPE

To evaluate the effectiveness of our proposed misbehavior detection, we intentionally insert malicious behavior in the code that corresponds to the attacks described in Section V. Synthetic flooding attack events are inserted in such a way that compromised IoT device transmit data every 100 ms for 1 second. Moreover, data tampering attack adopts eq. 12 where offset $\boldsymbol{l_d}$ is randomly generated from the range [1, 10].

Furthermore, we also considered the following attacking mode for malicious SGF nodes:

1. *Always-On Mode:* In the event where the base activation counter of firmware trojan is satisfied, the compromised SGF node, thereafter, continuously attacks when it has an opportunity. This means that an adversary will limit its attack when the disturbance, such as environmental noise, is low. On the contrary, it attacks more often as the noise level increases. Hence, we set the malicious SGF node to discretely attack within all aggregation time-space based on an attack condition given as $\boldsymbol{\nabla d} > \boldsymbol{\nabla d} * \boldsymbol{W}$ where $\boldsymbol{W}$ is 50%, 25%, and 20% when in low, moderate, and high noise, respectively. We patterned this attack scenario based on an analogy that criminals will not commit a crime when law enforcement officers are on alert and in contrast, they perform crimes when officers are not.
2. *Alternating Mode:* In the event where the base activation counter of the firmware trojan is satisfied, the compromised IoT device thereafter, alternately attacks. Hence, we set that the malicious SGF nodes to attack within every aggregation time-space and adopt the attack decision from Always-on mode as a preliminary condition and then finally decide to attack depending on the probabilistic condition. This means that when the first condition is satisfied, the compromised IoT device will finally attack when a randomly generated number [0 1] is less than an attack probability $\boldsymbol{P_a}$ (50%).
3. *Unobtrusive Mode:* In the event where the base activation counter of the firmware trojan is satisfied, the compromised IoT device, thereafter, unobtrusively attack

TABLE 3. Values of the parameters used In the simulation.

Param.	Values IoT1 – IoT2	Remarks
$[X_{TH}, X_{TL}]$	[30,20] – [90,70]	Triggering points of temperature and moisture level where X is T or M
$[X_H, X_L]$	[32,18] – [92,68]	Extreme points of temperature and moisture level when X is T or M.
x_0	32 – 75	Initial value of Kalman-Filter estimate.
R_{v0}	0.1 - 0.1	Initial value of the tuning parameter / measurement noise.
Q	0.01 - 0.01	Process noise in Kalman-Filter
ζ	(-0.39, 0.39) - (-0.2, 0.4)	Phenomenon's change of value per second depending on the status of cooling machine (1, 0) and sprayer (0, 1), respectively.
β_{c0}	1 - 1	Initial value of the performance parameter in Eq. 9
γ	50 – 50 secs	End time of training.
α_R	0.4 - 0.4	Learning rate in eq. 10
∇d	(10, 20, 30)% - (10, 20, 30)%	Sensed data's deviation due to (LOW, MODERATE, HIGH) noise with respect to true value.
l_d	[1, 10] – [1, 10]	Malicious offset in Eq. 12
Δc	(0.02, 0.02)V – (0.02, 0.04)V	Rate of voltage change per 5 secs depending on the status of cooling machine (1, 0) and sprayer (0,1), respectively.
A_t	50 sec	Aggregation time-space.
n	5 sec	Compliance check interval

IoT1: Monitors the node that handles temperature and cooler

IoT2: Monitors the node that handles soil-moisture level and sprayer

in order not to be easily detected. Hence, we set that the malicious IoT device attacks at every other aggregation period and follow the same attack decision from Always-on mode.

D. STATISTICAL ANALYSIS OF MISBEHAVIOR DETECTION

The estimation of compliance degree is more likely to be imperfect because of environmental noise, communication errors, or even software bugs. Thus, in this work, we adopt the model from [10] where the compliance degree of the embedded IoT node is a random variable C following the beta probability distribution function $\boldsymbol{G}(.) = \boldsymbol{Beta}(\boldsymbol{\alpha}, \boldsymbol{\beta})$. Accordingly, the average compliance degree of the target node over a bounded period can be computed using the G's mean equation given by $\boldsymbol{E}(\boldsymbol{P}) = \boldsymbol{\alpha}/(\boldsymbol{\alpha} + \boldsymbol{\beta})$. We set a fixed value of α as 1 and parameterized β using the maximum likelihood estimate, given by $\boldsymbol{\beta} = \boldsymbol{m}/\sum_{i=1}^{m} \boldsymbol{log}(\boldsymbol{1}/(\boldsymbol{1}-\boldsymbol{c_i}))$ where $\boldsymbol{c_i}$ is the target's node compliance degree history $\boldsymbol{c_1}, \boldsymbol{c_2}, \boldsymbol{c_3}, \ldots, \boldsymbol{c_m}$ collected during runtime. These assignments reduced the run time complexity of solving the values of the beta distribution's parameters α and β to $\boldsymbol{O(n)}$ which is extremely lightweight [10]. Consequently, the decision on which target node is considered as a misbehaving node is based on a binary criterion such that the node's average compliance degree is less than or equal to a predefined minimum compliance threshold $\bar{\boldsymbol{C}}_{\boldsymbol{T}}$.

The effectiveness of the adopted statistical analysis method can be measured by the false negative probability $\boldsymbol{P_{fn}}$ and false positive probability $\boldsymbol{P_{fp}}$. $\boldsymbol{P_{fp}}$ represents the likelihood that a well-behaved node is treated as misbehaving node, i.e., its average compliance degree is less than or equal to the minimum threshold $\bar{\boldsymbol{C}}_{\boldsymbol{T}}$. On the other hand, $\boldsymbol{P_{fn}}$ represents the likelihood that a misbehaving node is treated as well-behaved node, i.e.,., its average compliance degree is greater than the minimum threshold $\bar{\boldsymbol{C}}_{\boldsymbol{T}}$. Hence, we can

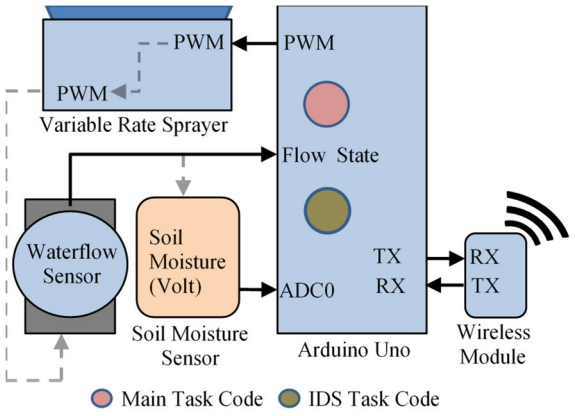


FIGURE 10. Block diagram of IoT device that monitors the soil-moisture level and controls the variable rate sprayer.

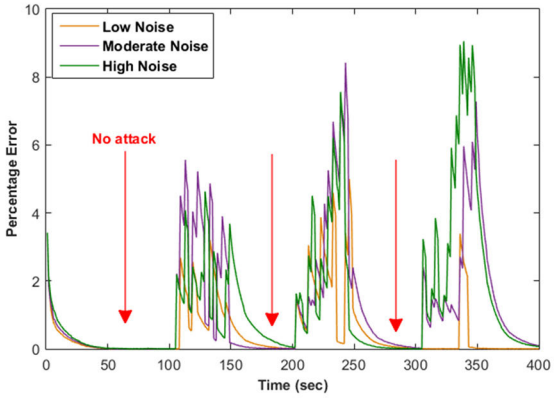


FIGURE 11. Percentage Error between estimated value and true temperature value.

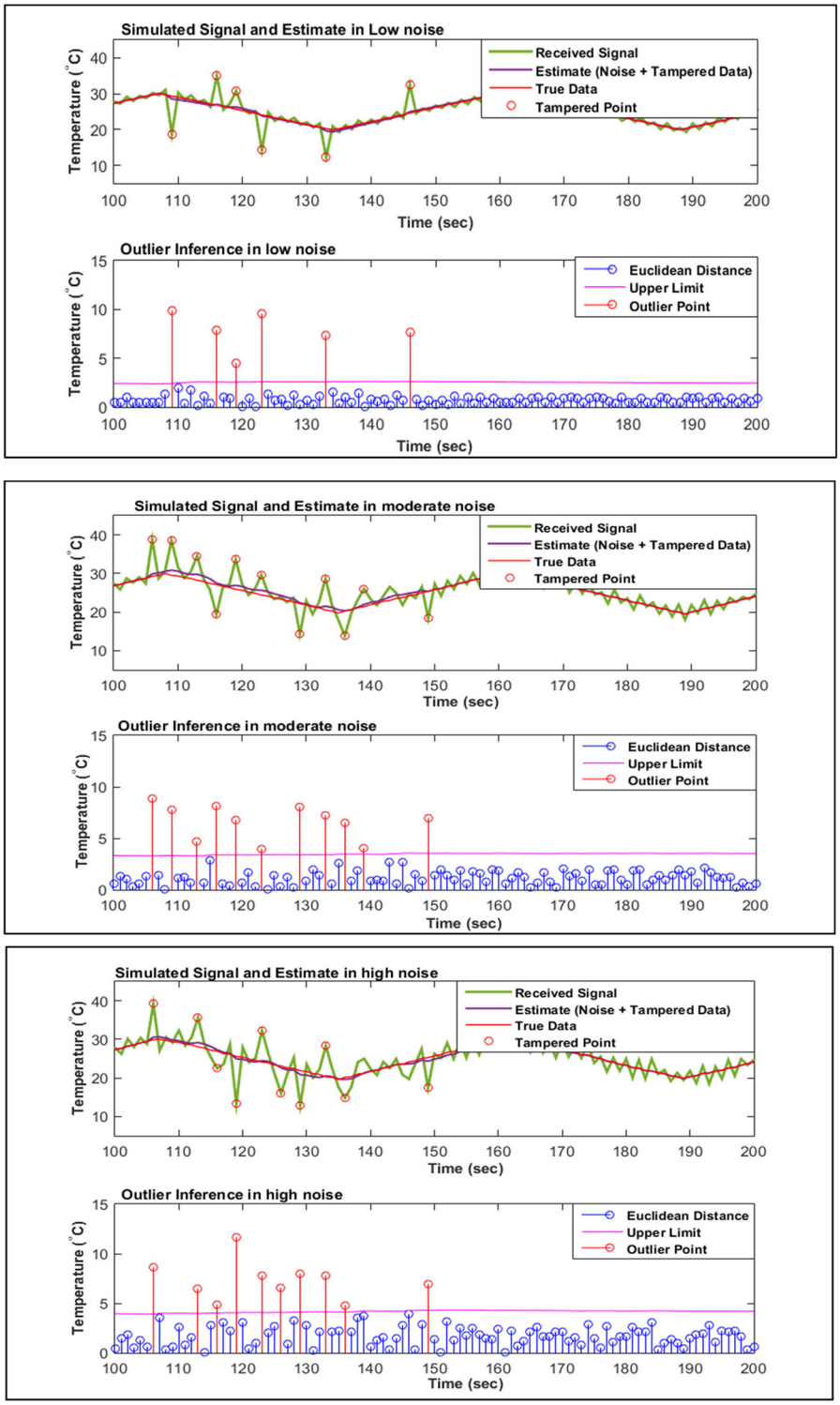


FIGURE 12. Estimation of microenvironment's temperature affected by noise and tampering is shown in top figures of each block. Meanwhile, bottom figures present the detected tampered data points based on a statistically derived limiting criterion.

compute $P_{fn} = Pr\left(C > \bar{C}_T\right) = 1 - G\left(C_T\right)$ given that the SGF node is "misbehave" and $P_{fp} = Pr\left(C < \bar{C}_T\right) = G(\bar{C}_T)$ given that the SGF IoT node is "well-behave" during the experimental run.

VIII. SIMULATION RESULT

In this section, we present the effectiveness of the Kalman-filter algorithm in the estimation of the operational data that are transmitted by the target node. We also show the effect

on the estimation when data are tempered by the attackers in different attack modes. Furthermore, we evaluate the performance of our proposed misbehavior detection technique in terms of ''effectiveness'' measured by detection rate, false positive probability, and false negative probability as well as the ''efficiency'' measured by memory utilization and computation time.

A. KALMAN-FILTER ESTIMATION AND OUTLIER INFERENCE

Figure 11 shows the estimated value of the temperature from the Kalman-filter method, the microenvironment temperature, and the signal received by the monitoring node, which is affected by noise and data tampering that is based on the unobtrusive attack mode. Moreover, as can be seen from the bottom graph of Figure 11, the corresponding points of the tampered data fall above the limiting criterion based on eq. 9 and algorithm 2. While in Figure 10, the percentage error between the true value and the calculated estimate of a received data is presented. As seen in the figure, the maximum percentage error is less than 10%. This error was incurred as a result of data tampering while the influence of noise is minimal. As observed in the figure, the estimated value eventually converges to the true value once there are no attack events. The same trend is observed in the always-on mode and alternate-mode attackers. This means that the proposed technique can detect the alteration of data and shows resiliency to the noise environment as well as to tampering attack.

In the experiment, the periodicity of data transmission is set at every 1 second. Thus, from the receiver's point of view, it follows that the expected arrival of the next message will have an average of 1 second. Figure 12 illustrates the timing reception of adjacent messages. As can be seen in the figure, the inter-arrival time of messages with respect to the expected arrival (1 sec) has a maximum variation of 1 ms when in a normal state. However, when a flooding event occurs, the absolute distance from the actual inter-arrival time to the expected time will be significantly large and thus violates the limiting criterion as illustrated in the figure. Hence, a compromised IoT device launching flooding attack can easily be detected. Note that the distance value of malicious points is scaled down for better visibility inside the graph.

B. EFFECTIVENESS PERFORMANCE EVALUATION OF IDS

We measure the effectiveness of our proposed approach by the following performance metrics: (a) detection rate: the probability of correctly identifying a misbehaving node; (b) false negative probability (P_{fn}): computed as 1 – detection rate; (c) false positive probability: the probability of misidentifying a well-behaved node as a misbehaving node; (d) AUROC: area under a receiver operating characteristics (ROC) curve formed by the detection rate vs false positive probability. Figure 13 presents the ROC curves for the case in which the misbehaving SGF nodes are in always-on and

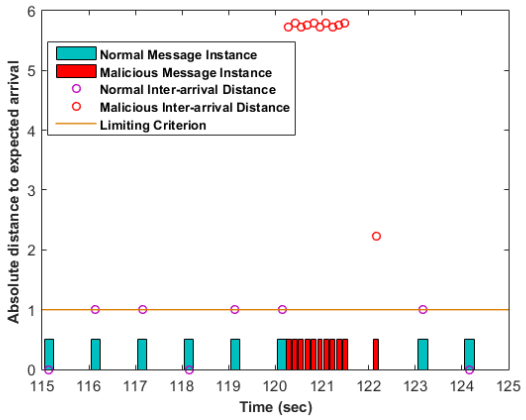


FIGURE 13. Illustration of inter-arrival time of messages during normal state and when a flooding attack is launched.

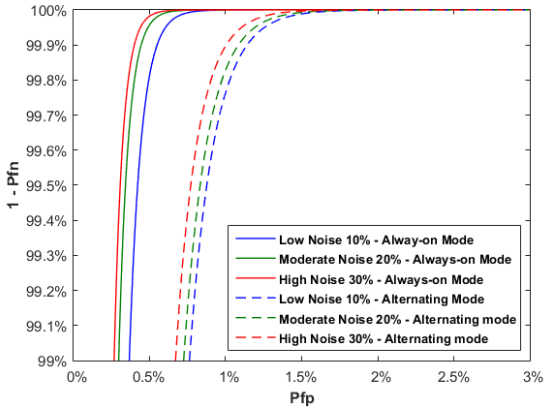


FIGURE 14. ROC curve of IoT device under different level of environment noise at *Always-on* and *Alternating* attack mode. The y-coordinate is the detection rate (1 - P_{fn}) and x-coordinate false positive rate (P_{fp}).

alternating attack mode. Moreover, each ROC curve has a noise level of 10%, 20%, 30% with respect to the true data. We see that in all noise levels, the AUROC is close to 100% since the false positive rate (P_{fp}) and false negative rate (P_{fn}) are close to zero. Furthermore, an interesting result shows that even in high noise, it can effectively detect a misbehaving IoT node. We can observe a slightly better accuracy compared with the lower noise level. This is because the effect of noise on the data is filtered out before the assessment of the target node's behavior.

Meanwhile, Figure 14 displays the ROC curves of an unobtrusive attacker at different noise levels. As can be seen in the figure, the proposed technique performs poorly in detecting an unobtrusive attacker. This is because the compromised IoT device is attacking very carefully to avoid detection. We can equate this mode as an insidious attacker that has known evaluation period of the compliance degree.

C. EFFICIENCY PERFORMANCE EVALUATION

We evaluate the efficiency of our proposed approach by the following metrics: (a) the memory consumption: the amount of memory utilized; (b) the computation overhead:

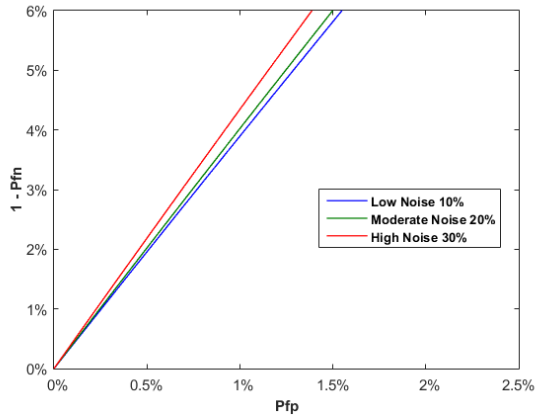


FIGURE 15. ROC curve of IoT device under different level of environment noise at *Unobtrusive* mode. The y-coordinate is the detection rate (1 - P_{fn}) and x-coordinate false positive rate (P_{fp}).

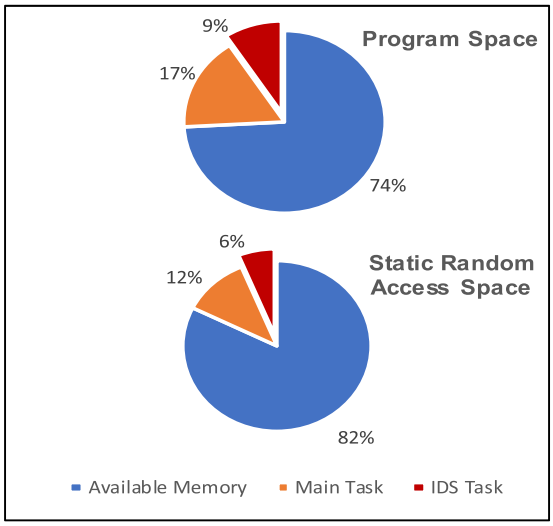


FIGURE 16. Memory Utilization of the main task and ids agent software.

the amount of computation time from data reception to the misbehavior detection.

Our proposed IDS agent is written for the Arduino-Uno platform. In this paper, we wrote the programs that include the main task of the device and the misbehavior detection task. The used platform has two pools of memory: program space of 32256 bytes and static random-access memory (SRAM) of 2048 bytes. Figure 15 shows the memory utilized by the main task and IDS task programs. The program space basically holds a set of instructions (or called sketch) that corresponds to the written program, while the SRAM is where a data holders (variables) are created and manipulated during run time. As can be seen in Figure 14, the footprint of the IDS instructions is approximately 9% of the program space and utilizes approximately 6% of the SRAM's space, which holds information of one target node. Hence, the IDS agent can accommodate approximately 13 more target nodes. However, it is recommended not to use up the whole SRAM to prevent unexpected failures. Moreover, the computation time can be obtained by capturing the time at two locations: upon complete reception of data and immediately after behavior assessment. The processes operating within this period include the Kalman-filter estimation, outlier classification, and the ASIs evaluation. Accordingly, based on our experiment, the average computation time needed to reach a decision is 3 milliseconds.

IX. CONCLUSION

The incorporation of ICT and embedded IoT in the agricultural sector poses risks to the quality and quantity of its products. This work contributes to the protection of resource-constrained embedded IoT, especially against zero-day attacks, in a smart greenhouse farm by proposing a behavior-rule specification-based distributed misbehavior detection technique where behavior-rules specific to the target environment were derived. A lightweight estimation algorithm based on Kalman-filter was also used to preprocess the received data to remove the effect of the different noise levels. Additionally, its integration resulted in the expansion of the monitor nodes' monitoring space in such a way that both monitor and target nodes do not need to have the same physical characteristics and access to the same target phenomenon. Moreover, an IDS agent was developed that can run on Arduino-Uno, and subsequently validated by extensively simulating in the Proteus simulation tool. From the experiments, it is shown that the proposed approach is effective in detecting a misbehaving node that launches an attack aggressively. However, we did not conduct a comparison against contemporary machine learning algorithms like KNN or SVM, which are popular classification schemes because of the limited memory storage of our selected platform. It is known that these schemes have a large memory footprint.

In future work, we will deploy our approach in a real greenhouse environment. We also plan to test our approach to different domains like healthcare or smart factory.

REFERENCES

[1] S. Linsner, R. Varma, and C. Reuter, "Vulnerability assessment in the smart farming infrastructure through cyberattacks," in *Proc. 38th GIL-Jahrestagung Wien, Digitalisierung Landwirtschaftliche Betriebe Kleinstrukturierten Regionen–ein Widerspruch Sich?*, Feb. 2019, pp. 119–124.

[2] L. Barreto and A. Amaral, "Smart farming: Cyber security challenges," in *Proc. Int. Conf. Intell. Syst. (IS)*, Sep. 2018, pp. 870–876.

[3] G. Suciu, C.-I. Istrate, and M.-C. Ditu, "Secure smart agriculture monitoring technique through isolation," in *Proc. Global IoT Summit (GIoTS)*, Jun. 2019, pp. 1–5.

[4] I. V. Kotenko, I. Saenko, and A. Branitskiy, "Applying big data processing and machine learning methods for mobile Internet of Things security monitoring," *J. Internet Services Inf. Secur.*, vol. 8, no. 3, pp. 54–63, Aug. 2018.

[5] M. S. Farooq, S. Riaz, A. Abid, K. Abid, and M. A. Naeem, "A survey on the role of IoT in agriculture for the implementation of smart farming," *IEEE Access*, vol. 7, pp. 156237–156271, 2019.

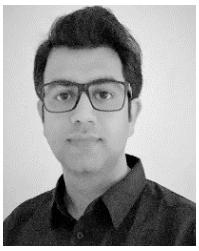
[6] T. Gundu and V. Maronga, "IoT security and privacy: Turning on the human firewall in smart farming," in *Proc. 4th Int. Conf. the*, vol. 12, 2019, pp. 95–104.

[7] M. Borrelli, V. Coric, C. Gnauer, J. Wolfgeher, and M. Tauber, "Security threats and risk analysis of an IoT Web service for a smart vineyard," *ERCIM News*, May 2018. [Online]. Available: https://gil-net.de/Publikationen/139_119.pdf

[8] C. Gritti, M. Önen, R. Molva, W. Susilo, and T. Plantard, "Device identification and personal data attestation in networks," *J. Wireless Mobile Netw., Ubiquitous Comput., Dependable Appl.*, vol. 9, no. 4, pp. 1–25, Dec. 2018.
[9] C. Cotroneo. *Why the Indoor Farming Movements Is Taking Off*. Accessed: 2020. [Online]. Available: https://www.treehugger.com/indoor-farming-study-obstacles-growth-potential-4859467
[10] I. You, K. Yim, V. Sharma, G. Choudhary, I.-R. Chen, and J.-H. Cho, "Misbehavior detection of embedded IoT devices in medical cyber physical systems," in *Proc. IEEE/ACM Int. Conf. Connected Health, Appl., Syst. Eng. Technol.*, Sep. 2018, pp. 88–93.
[11] V. Sharma, I. You, K. Yim, I.-R. Chen, and J.-H. Cho, "BRIoT: Behavior rule specification-based misbehavior detection for IoT-embedded cyber-physical systems," *IEEE Access*, vol. 7, pp. 118556–118580, 2019.
[12] M. Bhargavi, M. N. Kumar, N. V. Meenakshi, and N. Lasya, "Intrusion detection techniques used for Internet of Things," *Int. J. Appl. Eng. Res.*, vol. 14, no. 24, pp. 4462–4466, 2019.
[13] G. Efstathopoulos, P. R. Grammatikis, P. Sarigiannidis, V. Argyriou, A. Sarigiannidis, K. Stamatakis, M. K. Angelopoulos, and S. K. Athanasopoulos, "Operational data based intrusion detection system for smart grid," in *Proc. IEEE 24th Int. Workshop Comput. Aided Model. Design Commun. Links Netw. (CAMAD)*, Sep. 2019, pp. 1–6.
[14] C. Liang, B. Shanmugam, S. Azam, M. Jonkman, F. De Boer, and G. Narayansamy, "Intrusion detection system for Internet of Things based on a machine learning approach," in *Proc. Int. Conf. Vis. Towards Emerg. Trends Commun. Netw. (ViTECoN)*, 2019, pp. 1–6.
[15] I. You, K. Yim, V. Sharma, G. Choudhary, I.-R. Chen, and J.-H. Cho, "On IoT misbehavior detection in cyber physical systems," in *Proc. IEEE 23rd Pacific Rim Int. Symp. Dependable Comput. (PRDC)*, Dec. 2018, pp. 189–190.
[16] M. M. Ozcelik, E. Irmak, and S. Ozdemir, "A hybrid trust based intrusion detection system for wireless sensor networks," in *Proc. Int. Symp. Netw., Comput. Commun. (ISNCC)*, May 2017, pp. 1–6.
[17] H. Hui, X. An, H. Wang, W. Ju, H. Yang, H. Gao, and F. Lin, "Survey on blockchain for Internet of Things," *J. Internet Services Inf. Secur.*, vol. 9, no. 2, pp. 1–30, May 2019.
[18] R. Samrin and D. Vasumathi, "Review on anomaly based network intrusion detection system," in *Proc. Int. Conf. Electr., Electron., Commun., Comput., Optim. Techn. (ICEECCOT)*, Dec. 2017, pp. 141–147.
[19] N. Thanh Van, T. Ngoc Thinh, and L. Thanh Sach, "An anomaly-based network intrusion detection system using deep learning," in *Proc. Int. Conf. Syst. Sci. Eng. (ICSSE)*, Jul. 2017, pp. 210–214.
[20] R. B. Basnet, R. Shash, C. Johnson, L. Walgren, and T. Doleck, "Towards detecting and classifying network intrusion traffic using deep learning frameworks," *J. Internet Serv. Inf. Secur.*, vol. 9, no. 4, pp. 1–17, 2019.
[21] S. Otoum, B. Kantarci, and H. T. Mouftah, "Detection of known and unknown intrusive sensor behavior in critical applications," *IEEE Sensors Lett.*, vol. 1, no. 5, pp. 1–4, Oct. 2017.
[22] V. Korzhuk, A. Groznykh, A. Menshikov, and M. Strecker, "Identification of attacks against wireless sensor networks based on behaviour analysis," *J. Wireless Mob. Netw., Ubiquitous Comput. Dependable Appl.*, vol. 10, no. 2, pp. 1–21, 2019.
[23] L. Santos, C. Rabadao, and R. Goncalves, "Intrusion detection systems in Internet of Things: A literature review," in *Proc. 13th Iberian Conf. Inf. Syst. Technol. (CISTI)*, Jun. 2018, pp. 1–7.
[24] W. Zhang, B. Bu, and H. Wang, "An intrusion detection method of data tampering attack in communication-based train control system," in *Proc. IEEE Intell. Transp. Syst. Conf. (ITSC)*, Oct. 2019, pp. 345–350.
[25] Y. Kim and H. Bang, "Introduction to Kalman filter and its applications," in *Introduction and Implementations of the Kalman Filter*. Rijeka, Croatia: Intech, 2018.
[26] M. Shuai, K. Xie, G. Chen, X. Ma, and G. Song, "A Kalman filter based approach for outlier detection in sensor networks," in *Proc. Int. Conf. Comput. Sci. Softw. Eng.*, vol. 4, 2008, pp. 154–157.
[27] A. F. Echeverri, H. Medeiros, R. Walsh, Y. Reznichenko, and R. Povinelli, "Real-time hierarchical Bayesian data fusion for vision-based target tracking with unmanned aerial platforms," in *Proc. Int. Conf. Unmanned Aircr. Syst. (ICUAS)*, Jun. 2018, pp. 1262–1270.
[28] T. V. H. Pham, "Overview of IoT development in agriculture and applications in Vietnam," VNU Univ. Eng. Technol., Hanoi, Vietnam, Tech. Rep., 2018.
[29] S. R. Prathibha, A. Hongal, and M. P. Jyothi, "IOT based monitoring system in smart agriculture," in *Proc. Int. Conf. Recent Adv. Electron. Commun. Technol. (ICRAECT)*, Mar. 2017, pp. 81–84.
[30] J. Jaalinoja, "Requirements implementation in embedded software development," VTT Publications, Espoo, Finland, Tech. Rep. VTT-PUBS-526, 2004.
[31] *UPPAAL: Design Verification for Embedded Systems*. Accessed: Jun. 30, 2020. [Online]. Available: http://uppaal.com
[32] C. Konstantinou, A. Keliris, and M. Maniatakos, "Taxonomy of firmware trojans in smart grid devices," in *Proc. IEEE Power Energy Soc. Gen. Meeting (PESGM)*, Jul. 2016, pp. 1–5.
[33] M. Tehranipoor and F. Koushanfar, "A survey of hardware trojan taxonomy and detection," *IEEE Des. Test. Comput.*, vol. 27, no. 1, pp. 10–25, Jan. 2010.

PHILIP VIRGIL ASTILLO received the B.S. degree in computer engineering from the University of San Carlos, Cebu, Philippines, in 2009, and the M.Eng. degree in computer engineering from the University of San Carlos, in 2011. He is currently pursuing the Ph.D. degree in information security engineering with Soonchunhyang University, South Korea. From 2009 to 2015, he worked as a Lecturer with the University of San Carlos, where he was a Research Assistant in the Phil-LiDAR Program from 2014 to 2015. From 2015 to 2016, he was a Research Assistant with Sensor Laboratory, Clemson University, Clemson, SC, USA. His research interests include sensor development, embedded system design and development, mobile Internet security, 5G security, IoT security, and intrusion detection system.

JIYOON KIM (Student Member, IEEE) received the M.S. degree in information security engineering from Soonchunhyang University, South Korea, where he is currently pursuing the Ph.D. degree with the Department of Information Security Engineering. His current research interests include mobile Internet security, 5G security, and formal security analysis.



VISHAL SHARMA (Member, IEEE) received the B.Tech. degree in computer science and engineering from Punjab Technical University, in 2012, and the Ph.D. degree in computer science and engineering from Thapar University, in 2016. He is currently working as a Lecturer (Assistant Professor) with the School of Electronics, Electrical Engineering and Computer Science (EEECS), Queen's University Belfast (QUB), Northern Ireland, U.K. Before coming to QUB, he was a Research Fellow with the Information Systems Technology and Design (ISTD) Pillar, Singapore University of Technology and Design (SUTD), Singapore, where he worked on the future-proof blockchain systems, funded by SUTD-MoE. From November 2016 to March 2019, he worked with the Information Security Engineering Department, Soonchunhyang University, South Korea, in multiple positions, i.e., as a Postdoctoral Researcher (November 2016–December 2017) and as a Research Assistant Professor (January 2018–March 2019). He also held a joint postdoctoral position with Soongsil University, South Korea. He was affiliated with the Industry-Academia Cooperation Foundation and the Mobile Internet Security Laboratory at

Soonchunhyang University. Before this, he worked as a Lecturer with the Computer Science and Engineering Department, Thapar University, India. He is a professional member of the ACM and the Past Chair for the ACM Student Chapter-TIET Patiala. He has authored/coauthored more than 100 journal/conference papers and book chapters and has co-edited two books with Springer. He has served/serving as a Guest Editor for MIS, IJDSN, WCMC, MDPI (Sensors, Drones, Future Internet), and Autosoft journals. He is a recipient of three best paper awards from the International Conference on Communication, Management and Information Technology (ICCMIT), Warsaw, Poland, in April 2017; CISC-S'17 South Korea in June 2017; and IoTaas Taiwan in September 2017. He was the Track Chair of MobiSec'16 and AIMS-FSS'16, and a PC member and reviewer of MIST'16 and MIST'17, respectively. He has served as the TPC member of ETIC- 2019, WiMO-2019, ITNAC-IEEE TCBD'17, ICCMIT'18, CoCoNet'18, and ITNAC-IEEE TCBD'18. Furthermore, he serves as a Reviewer for various ACM/IEEE Transactions and other journals. He also serves as the ATE for the *IEEE Communications Magazine* and an AE for the IET-CAAI TRIT. His areas of research and interests are 5G networks, Blockchain systems, aerial (UAV) communications, CPS-IoT, and mobile Internet systems.

ILSUN YOU (Senior Member, IEEE) received the M.S. and Ph.D. degrees in computer science from Dankook University, Seoul, South Korea, in 1997 and 2002, respectively, and the Ph.D. degree from Kyushu University, Japan, in 2012. He is currently working as a Professor with the Department of Information Security Engineering, Soonchunhyang University, South Korea. His main research interests include Internet security, authentication, access control, and formal security analysis. He is a Fellow of the IET. He is the EiC of the *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications* (JoWUA), and the *Journal of Internet Services and Information Security* (JISIS). He is on the Editorial Board of *Information Sciences*, *Journal of Network and Computer Applications*, *International Journal of Ad Hoc and Ubiquitous Computing*, *Computing and Informatics*, *Intelligent Automation and Soft Computing*, and so on.