IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.Doi Number

BRIoT: Behavior Rule Specification-based Misbehavior Detection for IoT-Embedded Cyber-Physical Systems

Vishal Sharma[1], Member, IEEE, Ilsun You[1], Senior Member, IEEE, KangbinYim[1], Ing-Ray Chen[2], Member, IEEE, and Jin-Hee Cho[2], Senior Member, IEEE

[1]Dept. of Information Security Engineering, Soonchunhyang University, Asan-si-31538, South Korea
[2]Virginia Tech, Department of Computer Science, VA 24061, USA

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

This work was supported by Institute for Information &communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No.2017-0-00664, Rule Specification-based Misbehavior Detection for IoT-Embedded Cyber-Physical Systems). The work was also supported by the U.S. AFOSR under grant number FA2386-17-1-4076.

ABSTRACT The identification of vulnerabilities in a mission-critical system is one of the challenges faced by a Cyber-Physical System (CPS). The incorporation of embedded Internet of Things (IoT) devices makes it tedious to identify vulnerability and difficult to control the service-interruptions and manage the operations losses. Rule-based mechanisms have been considered as a solution in the past. However, rule-based solutions operate on the goodwill of the generated rules and perform assumption-based detection. Such a solution often is far from the actual realization of IoT runtime performance and can be fooled by zero-day attacks. Thus, this paper takes this issue as a motivation and proposes better lightweight behavior rule specification-based misbehavior detection for IoT-embedded cyber-physical systems (BRIoT). The key concept of our approach is to model a system with which misbehavior of an IoT device manifested as a result of attacks exploiting the vulnerability exposed may be detected through automatic model checking and formal verification, regardless of whether the attack is known or unknown. Automatic model checking and formal verification are achieved through a 2-layer Fuzzy-based Hierarchical Context-Aware Aspect-Oriented Petri Net (HCAPN) model, while effective misbehavior detection to avoid false alarms is achieved through a Barycentric-coordinated based center of mass calculation method. The proposed approach is verified by an unmanned aerial vehicle (UAV) embedded in a UAV system. The feasibility of the proposed model is demonstrated with high reliability, low operational cost, low false-positives, low false-negatives, and high true positives in comparison with existing rule-based solutions.

INDEX TERMS behavior rules, cyber-physical systems, IoT, specification-based intrusion detection, and zero-day attacks.

I. INTRODUCTION

Misbehavior detection techniques for Internet of Things (IoT) embedded cyber-physical systems (CPS) in general can be classified into three types: signature-based, anomaly-based and specification-based techniques [12, 28]. The proposed behavior rule specification-based misbehavior detection technique in this work falls under specification-based detection. The proposed approach disposes of signature-based detection so as to deal with zero-day attacks. It considers specification-based techniques rather than anomaly-based techniques for misbehavior detection to avoid the high cost associated with profiling and learning anomaly patterns for resource-constrained IoT devices and to avoid high false positives (treating good devices as bad devices). We argue that contemporary anomaly-based misbehavior detection methods for IoT-embedded CPSs based on profiling and machine learning through correlation and statistical analysis of a large amount of data or logs for classifying misbehavior (e.g., [2, 6-7, 10-11, 14-15, 29]) will not work for IoT-embedded CPSs because of high memory, run time, communication, and computational overhead, considering the fact that many embedded IoT devices are severely constrained in resources. Specification-based misbehavior detection provides a viable approach for misbehavior detection of embedded IoT devices because of light resource requirements for checking misbehaviors against specifications.

The goal of this work is to develop a *Behavior Rule specification-based embedded-IoT misbehavior detection technique* (called BRIoT for short) to achieve high accuracy in detecting misbehavior of an embedded IoT device in a CPS against zero-day attacks, without incurring high memory, run time, communication, or computation overhead by avoiding the high cost of profiling and learning anomaly patterns as in anomaly detection. To achieve the goal of defending against zero-day attacks, BRIoT detects "intended behaviors" specified in the "operational profile" [16] (i.e., mission specification) for every IoT device such that misbehaviors manifested as a result of attacks exploiting the vulnerability exposed may be detected through automatic model checking and formal verification. Moreover, our method to defend against zero-day attacks that try to avoid pre-established rule specification-based misbehavior detection is to identify the complete set of misbehaving states deriving from the device's operational profile that can possibly fail a mission assigned for execution. A malicious UAV can avoid being detected only if it never enters a misbehaving state, in which case the IoT device will not cause any harm to the mission execution because no failure will ever result if the IoT device never enters any misbehaving state.

In a large IoT-embedded CPS, there will be a huge number of IoT sensors/actuators and it is neither scalable nor practical to rely on a central entity to perform misbehavior detection. Since the central entity cannot physically perform misbehavior detection itself, it needs to collect misbehavior reports/logs from IoT devices. The amount of traffic generated will not only consume IoT energy but also cripple the CPS communication network. Hence, distributed misbehavior detection is the only feasible way. Since IoT devices are resource-constrained, the detection must be lightweight. For scalability, we propose a methodology to transform behavior rules to a state machine, turning behavior monitoring of an embedded IoT device into a lightweight process because it only involves checking if a monitored IoT device is in a safe or unsafe state against the transformed state machine.

The following aspects are novel in our work relative to the existing specification-based intrusion detection techniques (see Section 2 Related Work for detail): (1) design and implementation of a module for automatically modeling and deriving behavior rules from an embedded IoT device's operational profile specifications [16];(2) design and implementation of a 2-layer Fuzzy-based Hierarchical Context-Aware Aspect-Oriented Petri Net (HCAPN [33]) model to formally verify that the behavior rules generated are correct and cover all the threats (or satisfy the security requirements) and that the resulting safe and unsafe states are complete and are generated correctly with respect to the behavior rules specified;(3) design and implementation of a module for automatically transforming behavior rules into "attack behavior indicators" (ABIs) and then into a state machine for misbehavior detection at runtime;(4) design and implementation of a lightweight runtime collection module for collecting compliance degree data from runtime monitoring of an IoT device based on its derived state machine; (5) design and implementation of a lightweight statistical analysis module for effective misbehavior detection to avoid false alarms through a novel Barycentric-coordinated based center of mass calculation method; and (6) experimental verification by an unmanned aerial vehicle cyber physical system (UAV-CPS) demonstrating its superior performance over a contemporary specification-based intrusion detection solution called BRUIDS [18].

The rest of the paper is organized as follows: In Section II, we survey related work. In Section III, we discuss the system model. In Section IV, we describe in detail the design and implementation of BRIoT. In Section V, we apply BRIoT to misbehavior detection of a UAV embedded in a UAV-CPS and perform a comparative analysis of BRIoT against BRUIDS. Finally, in Section VI, we conclude the paper and outline future work.

II. RRELATED WORK

In this section, we discuss related work in three areas: anomaly-based IoT misbehavior detection, specification-based IoT misbehavior detection, and verification of specification-based IoT misbehavior detection. We compare and contrast our work with existing work.

Anomaly-based IoT Misbehavior Detection: Existing intrusion detection methods for IoT mostly are designed to detect either routing attacks or Denial of Service (DoS) attacks (see a survey in [28]). More recent works such as [29] also addressed detecting illegal memory accesses in low-power IoT. These existing works, however, are based on anomaly-based techniques applying profiling and machine learning through correlation and statistical analysis of a large amount of data or logs for classifying misbehavior (e.g., [2, 6-7, 10-11, 14-15, 29]). We believe anomaly-based detection techniques will not work for IoT-embedded CPSs because many embedded IoT devices especially battery-operated ones are severely constrained in resources. Our work is based on lightweight specification-based intrusion detection for misbehavior detection of each IoT device embedded in a CPS.

Specification-based IoT Misbehavior Detection: In the literature, specification-based misbehavior detection has been mostly applied to communication networks [4, 8, 21] and CPS security [1, 9, 17, 18, 30]. In the context of communication networks, DaSilva et al. [4] proposed traffic-based rules to detect network intruders: interval, retransmission, integrity, delay, repetition, radio transmission range and jamming. Ioannis et al. [8] proposed auditing the forwarding behavior of suspects to detect blackhole and greyhole attacks based on rule specification violations. Song et al. [21] proposed specification-based detection rules (identifying activity that is monitored) to ensure the global security requirement is obeyed for an IP configuration protocol in mobile ad networks. In the context of CPS security, Berthier et al. [1] proposed specification-based misbehavior detection to audit the network traffic among smart meters and access points for protocol compliance. Jokar et al. [9] considered specification-based misbehavior detection against physical and MAC layer attacks in ZigBee networks in smart grids. Mitchell et al. [17, 18] discussed a conceptual model of behavior rule specification-based intrusion detection for CPSs and conducted a proof-of-concept statistical analysis using pre-generated data following a probability distribution. Khan et al. [30] proposed behavior-based executable specification against false data injection attacks for industrial control systems. Our contribution relative to existing works cited above is that we pioneer the use of lightweight behavior rule specification-based misbehavior detection for resource-constrained IoT devices embedded in a CPS.

Verification of Specification-based Intrusion Detection: While specification-based detection in general induces a lower false positive rate than anomaly detection, a limitation of specification-based approaches is the difficulty of verifying that the specifications are correct and cover all the threats [1]. Toward this end, Song et al. [21] described a formal reasoning framework to first define a global security requirement and then defined the specifications of the behaviors of local nodes to assure the global security property. Utilizing the ACL theorem prover[32], they formally proved that the local detection rules (identifying local behavior that is monitored against behavior specifications) imply the global security requirement. Berthier et al. [1] followed a similar approach and proposed a formal framework comprising a model of the network, monitoring operations, protocol specifications, and security policy. The key idea of their framework is to formally verify that no network trace can violate the security policy without being detected. Utilizing ACL, they verify that all possible network traces that respect the network model, monitoring operations, and protocol specifications will also respect the security policy. Unlike the above-cited work [1, 21], we start with the "operational profile" [16] of an embedded IoT that defines the mission statement of the embedded IoT device to derive the security requirements and hence the threats of the embedded IoT device. Then we derive the behavior rules specifying the intended behavior and verify that the behavior rules are correct and cover all the threats. We develop a 2-layer Fuzzy-based HCAPN model for formal verification. Lastly, unlike [1, 21], our approach is specifically designed for intrusion detection of lightweight IoT devices embedded in a CPS with energy consideration.

III.SYSTEM MODEL

In this section, the system model, including the architecture model, threat model, and monitoring model on which the proposed IDS are based upon, is discussed in detail.

A. ARCHITECTURE MODEL

An embedded IoT device can be a sensor, an actuator, a controller, or a combination of the above such as a UAV. The architecture model depends on the specific type of IoT device under consideration. We illustrate it with an embedded UAV device in a UAV-CPS as considered in [18] with the addition of the misbehavior detection module (labeled as BRIoT) and the external architecture, where the information is served with both the distantly placed monitoring station and the other UAVs, as shown in Fig.1.

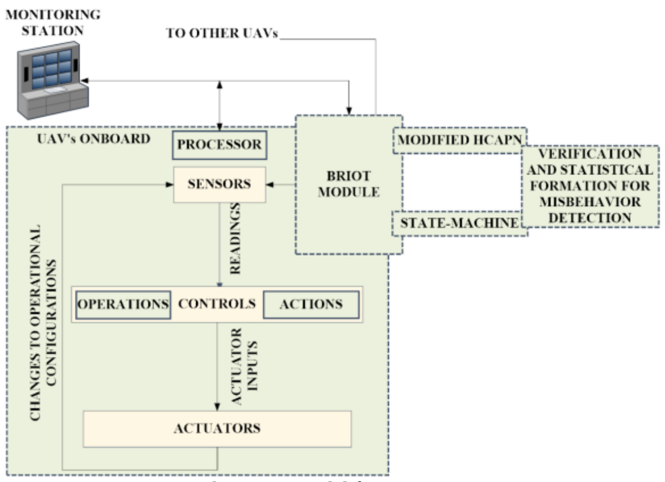


Fig. 1: Architecture Model for a UAV-CPS.

B. THREAT MODEL

We first understand the meanings of threats and attackers with the following definitions:

Definition 1: A threat is a negative event that can lead to an undesired outcome, such as damage to or a loss of an asset. Threats can use or become more dangerous because of a vulnerability (which is simply a weakness in the system).

Definition 2: A threat agent or an attacker is a person, actor, entity, or organization that is initiating a threat event.

In this paper, our primary interest is on attacks of embedded IoT devices performing basic sensing, actuating, navigating, and networking functions. Our threat model considers all threats that target integrity, confidentiality and availability aspects of IoT-embedded CPS security. The known attacks that have been investigated in the literature are summarized in Table 1. Unlike most existing IoT intrusion detection approaches which design specific intrusion detection functions to detect or prevent specific known attacks [28], we take an entirely different approach. That is, we use the design concept of "operational profile" [16] during the testing and debugging phase of an embedded IoT device when the IoT software is built to define the embedded IoT device's security requirements, from which the threat model is derived. The threat model comprises a list of threats that would fail an embedded IoT device's mission assignment. The threat model leads to a set of behavior rules against which the misbehavior of an IoT device would be detected automatically at runtime, regardless of if the attack is known (e.g., as listed in Table 1) or unknown. In our work, we formally verify that the behavior rules generated are correct and cover all the threats (or satisfy the security requirements).

C. MISBEHAVIOR MONITORING MODEL

Our behavior-rule based IDS approach relies on the use of monitoring nodes. We assume that a monitoring node performs misbehavior detection on a target node. One possible design is to have a sensor (actuator) monitor another sensor (actuator respectively) within the same CPS. This may require each sensor (actuator) to have multiple sensing functionalities. Note that a malicious embedded IoT device cannot bypass misbehavior detection because our approach is based on a device being monitored by a peer device (or more than one peer IoT devices to increase the detection strength). If a peer monitoring IoT device is itself malicious and performs attacks, its misbehavior would be detected by another peer IoT device. Further, whenever an IoT device is identified as malicious, its monitoring duty would be reassigned to another IoT device. Therefore, no malicious IoT device can bypass detection in our approach. Another possibility is that each IoT device is built on top of secure computational space (e.g., [40]) such that each target IoT device can execute misbehavior detection code in a secure computation space and self-monitor itself, even if the operating kernel has been compromised. In this case, once a node identifies itself as misbehaved based on the behavior rule specification, it can take itself off the mission or even self-shutdown.

Table 1: "Known" Attacks that Target Integrity, Confidentiality and Availability Aspects of IoT-CPS Security

Attack Type	Security Aspect
command spoofing attack [20], data spoofing attack [12], bad-mouthing/ballot-stuffing attack [3], capture attack [13], GPS spoofing attack [11, 27]	integrity
data exfiltration attack [13]	confidentiality
DoS or jamming attack [12], black/grey hole attack [12], energy exhaustion attack [25]	availability

IV. BEHAVIOR RULE SPECIFICATION-BASED MISBEHAVIOR DETECTION FOR EMBEDDED IOT

We first explain the workflow of BRIoT, as illustrated in Fig. 2. The automatic derivation of behavior rules is done at static time (or compile time) given a target IoT device's operational profile as input. Each behavior rule is then converted into a corresponding "attack behavior indicator"(ABI) being expressed as a Boolean expression to be evaluated true (1) or false (0), indicating whether the corresponding behavior rule is violated or not. All ABIs thus generated (corresponding to all behavior rules) are encoded in XML format and are fed as input to a HCAPN tool which does automatic model checking and formal verification also at static time. Once the behavior rules are formally verified and proven correct, we transform the corresponding ABIs into a C-language state machine for misbehavior detection of the specified target IoT device. This part is also performed at static time. Then we preload the state machine into the memory of a monitoring node and assign the monitoring node the duty of monitoring and detecting misbehavior of the target IoT device. This misbehavior detection part is performed at runtime. During runtime, misbehavior data detected if any are collected by the monitoring node via anon-board lightweight data collector. Subsequently the data collected are fed into a lightweight statistical analyzer (also on-board as it is lightweight) to judge if the target IoT device is malicious.

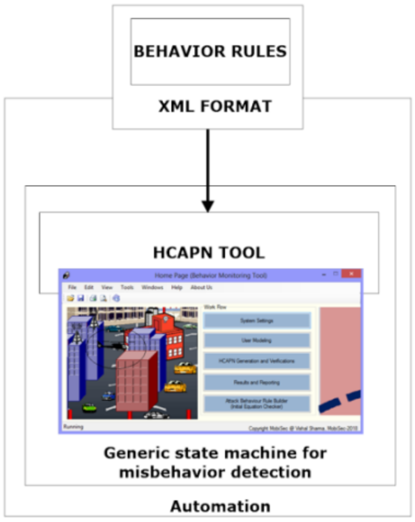

BEHAVIOR RULES
XML FORMAT
HCAPN TOOL
Home Page (Behavior Monitoring Tool)
System Settings
User Modeling
HCAPN Generation and Verifications
Results and Reporting
Attack Behaviour Rule Builder
Generic state machine for misbehavior detection
Automation

Fig.2: Workflow of BRIoT.

The HCAPA tool in Fig. 2 is developed to ease automation of model checking and formal verification. The tool uses basic coding principles which are extended to fit into the need of the proposed 2 layers statistical HCAPN model. The tool not only helps parse the user's or expert's inputs but also checks whether the developed rules are formally verifiable or not. It further allows visualization of the final model in the form of workflow through Petri Net visualization. The tool helps generate the reports and obtain results to check the basic principles of HCAPN model. In addition, it provides a high flexibility to model different behavior rules and attack behaviors. Fig. 3 provides an overview of our BRIoT design. In the following, we detail our BRIoT design in three major areas: automatic modeling and verification of behavior rule specification for an embedded IoT device through HCAPN (Section IV.A), automatic transformation of a behavior rule set to a state machine for misbehavior detection (Section IV..B), and lightweight runtime collection of compliance degree data and statistical analysis (Section IV.C).

A. AUTOMATIC MODELING AND VERIFICATION OF BEHAVIOR RULE SPECIFICATION FOR AN EMBEDDED IOT DEVICE THROUGH HCAPN

We propose to use "operational profile" [16], which essentially is a mission assignment statement generated according to the probabilities with which events will happen to an embedded IoT device during its mission execution as input to BRIoT. A mission assignment statement explicitly defines a set of security requirements for the mission to be successful, from which a set of threats as well as a set of behavior rules to cope with the threats may be automatically derived. If the "operational profile" of an embedded IoT device (by software engineers who developed the IoT device) which defines the security requirements is available, then it can be modeled to automatically identify the complete set of threats and consequently derive the complete set of behavior rules. Otherwise, the system designer would be guided to define the "anticipated operational profile" as input.

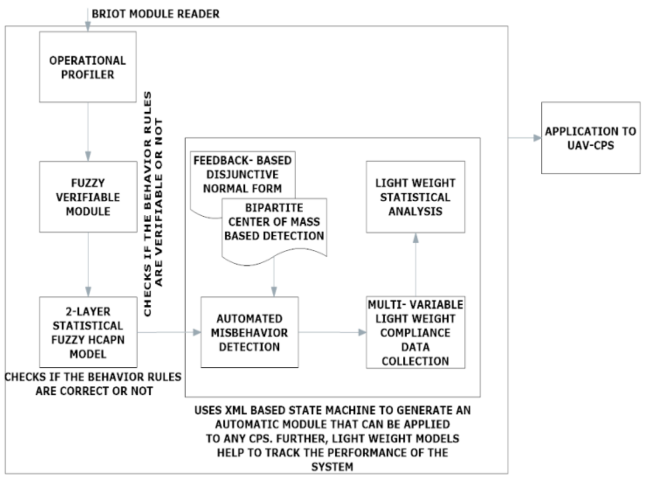

BRIOT MODULE READER
OPERATIONAL PROFILER
FUZZY VERIFIABLE MODULE
2-LAYER STATISTICAL FUZZY HCAPN MODEL
CHECKS IF THE BEHAVIOR RULES ARE VERIFIABLE OR NOT
CHECKS IF THE BEHAVIOR RULES ARE CORRECT OR NOT
FEEDBACK- BASED DISJUNCTIVE NORMAL FORM
BIPARTITE CENTER OF MASS BASED DETECTION
LIGHT WEIGHT STATISTICAL ANALYSIS
AUTOMATED MISBEHAVIOR DETECTION
MULTI- VARIABLE LIGHT WEIGHT COMPLIANCE DATA COLLECTION
USES XML BASED STATE MACHINE TO GENERATE AN AUTOMATIC MODULE THAT CAN BE APPLIED TO ANY CPS. FURTHER, LIGHT WEIGHT MODELS HELP TO TRACK THE PERFORMANCE OF THE SYSTEM
APPLICATION TO UAV-CPS

Fig.3: Overview of BRIoT design.

The automatic model verification of the behavior rules is conducted by verifying that the behavior rules generated are correct and covers all the threats (or satisfies the security requirements). The basic idea is to prove that the behavior rules can guarantee all security requirements are not violated, so any violation of the security requirements implies violations of the behavior rules. This means all attacks that violate the security requirements will be detected by the behavior rules.

The formal proof is made possible by expressing the behavior rules generated and the security requirements derived in a HCAPN [33] model such that "any violation of the security requirements implies violations of the behavior rules" is expressed as Boolean expressions in HCAPN. The model verification begins with generating a HCAPN model from the generated behavior rules. The newly generated HCAPN is a fuzzy-based statistical 2-layer model that is lightweight on memory and running time.

More specifically, a system comprising a set I of IoT devices is considered, with the cardinality $|I|$ denoting the number of IoT devices. Each IoT device must execute certain operations leading to a behavior set B generated automatically through the operational profile and must be verified before deployment. The verification is accounted with a behavior recording variable V**,** a Boolean variable that tells whether the behavior set B is verifiable or not. If verifiable, it marks whether the verified behavior is correct or incorrect by using another Boolean expression (G).The correctness variable, G, is accounted through HCAPN observations and can be written as G=*Fuzzy(H(.))* where (.) denotes the functional inputs to the HCAPN model defined as in [33]. By extending the initial model, V can be expressed as a fuzzy function [42] related to the behavior variables from the behavior set B, the degree of dependence of behavior represented by a set D, and a statistical weightage set W generated based on the dependence value, such that V=*Fuzzy(B, D, W)*.Here, V can operate on a vector of behavior rules or an individual rule depending on the initial observations as well as the supporting model available from an expert (E). For an expert, the verification function can be modeled as V_E=*Fuzzy(B, D, W)*$_E$. The values from an expert remain unchanged for a specified duration. However, for observations of the CPS, timing represents a key role because it becomes important to consider an instance-based (timely) fuzzy function written as V_T=*Fuzzy(B, D, W)*$_T$.

The proposed approach considers users and experts for operations, where users are the track-able devices with behavior rules whose evaluations are to be verified, whereas experts are the original sources available for testing, validating and defining the correct system. Although, the proposed model can work as an independent unit, we model it around the expert's observations for proving correctness. Usually, it is questionable that the availability of expert's values can directly provide correctness of the observed or recorded values. Therefore, an additional methodology is required. To answer this, an expert can provide base values for a given CPS. In practice, a user may encounter a different set of metrics, which could be dynamically verified and adjusted to form a base for timely detection of misbehavior patterns. Moreover, with verifications after certain time is elapsed, user's values can replace expert's values, thereby allowing the approach to settle into strong priori-probabilities.

The use of fuzzy logic [42] for deciding the outcomes of V helps to observe a Boolean value from the unevenly observable crisp values of B, D, and W. To find D, the initial observational values for the behavior rule set B of a given device in I are taken, such that a correlation coefficient (r_{UE}) [34] is identified for the user's as well as the expert's sets as follows:

$$r_{UE} = \frac{\vartheta \sum_{i=1}^{\vartheta} \alpha_i \beta_i - \sum_{i=1}^{\vartheta} \alpha_i \sum_{j=1}^{\vartheta} \beta_j}{\sqrt{\vartheta \sum_{i=1}^{\vartheta} \alpha_i^2 - (\sum_{i=1}^{\vartheta} \alpha_i)^2}\sqrt{\vartheta \sum_{j=1}^{\vartheta} \beta_j^2 - (\sum_{j=1}^{\vartheta} \beta_j)^2}}, \vartheta_U \neq \vartheta_E (1)$$

where ϑ is the total number of variables from B with uniqueness for user and expert in totality ($\vartheta = \vartheta_U + \vartheta_E$), $\alpha_i$is the number of occurrences of variable i in the behavior profiling set by the user, and β_i is the number of occurrences of variable i in the behavior profiling set by the expert. Now, based on these observations, the dependence of the kth behavior rule for the user can be evaluated as:

$$D_{U,k} = r_{UE} \cdot \frac{\sum_{j=1}^{n_{1,k}} \alpha_j}{\sum_{i=1}^{\vartheta} \alpha_i \beta_i}, n_1 \leq \vartheta_U,\ r_{UE} \neq 0 \tag{2}$$

where $n_{1,k}$ is the number of variables for a given behavior rule $k \in B$ of a user's input. Similarly, the dependence of the kth behavior rule for an expert can be written as:

$$D_{E,k} = r_{UE} \cdot \frac{\sum_{j=1}^{n_{2,k}} \beta_j}{\sum_{i=1}^{\vartheta} \alpha_i \beta_i}, n_2 \leq \vartheta_E,\ r_{UE} \neq 0 \tag{3}$$

where $n_{2,k}$ is the number of variables for a given behavior rule $k \in B$ of an expert's input. The ratio of dependence for a given behavior rule $k \in B$ can be given as $R_{UE,k} = \frac{\sum_{j=1}^{n_{1,k}} \alpha_j}{\sum_{j=1}^{n_{2,k}} \beta_j}$.

Now, D for each behavior rule k (subscript k is omitted below) can be marked as:

$$D = \begin{cases} D_U = D_E, if\ R_{UE} = 1, equal\ variables \\ R_{UE}, if\ proportionate\ and\ D_U \leq D_E \\ \frac{\gamma_U D_U + \gamma_E D_E}{\gamma_U + \gamma_E}, \gamma_U + \gamma_E \neq 0\ , importance\ based\ measurement \\ \gamma_{U,t-1} + \sum_{i=1}^{n_1} \gamma_{U,t,i} \alpha_i + \sum_{j=1}^{n_1} \beta_j - \alpha_j\ , continuous\ monitoring \\ and\ noncompliance\ with\ the\ above\ three \end{cases} \tag{4}$$

Here, γ_U and γ_E are the importance coefficients indicating importance of a behavior rule for the user and the expert, respectively. They are derived from the behavior set B and its contained variables. Specifically, they can be derived based on a linear model [35] [36] used in the formulation of D, such that $\sum_{i=1}^{n_1} \mu^2$ is minimum, where $\mu = \gamma_E \sum_{j=1}^{n_2} \beta_j - \gamma_U \sum_{j=1}^{n_1} \alpha_j$. With $\gamma_E = 1, \mu$ can be computed as $\overline{\beta_j} - \gamma_U \overline{\alpha_j}$. If D is higher then it becomes easier to detect the possibility of behavior rules being verifiable, which otherwise is difficult for isolated variables in the behavior rules. Setting D equal to the user to expert ratio of dependence, i.e., $D = R_{UE}$ is convenient to use under the given constraints as it allows verification between the user's and the expert's inputs. The fourth sub-value for D helps to evaluate a continuously changing system. However, this requires setting certain thresholds on the number of new variables in behavior profiling. An unlimited number of new variables may cause additional overheads as it becomes tedious to find dependence for all additional variables with limited knowledge. Here, knowledge refers to the available content from the expert and device profile available from the manufacturer.

For W, a memory coefficient (φ) is considered for each behavior rule, which helps to depict the statistical requirement (mean occurrences) of a behavior rule and is uniformly distributed with the value given based on CDF, such that:

$$\varphi_i = \prod_{j=1}^{n_1} \frac{\alpha_{i,j} - \min(\alpha) + 1}{\max(\alpha) - \min(\alpha) + 1} \tag{5}$$

For relative memory, the observations change to:

$$\varphi_{UE,i} = \prod_{j=1,q=1}^{n_1,n_2} \frac{||\alpha_{i,j} - \beta_{i,q}|| - \min(\alpha,\beta) + 1}{\max(\alpha,\beta) - \min(\alpha,\beta) + 1} \tag{6}$$

where the choice between the two is subject to system constraints and applicability. Now, W can be accumulated through a Wannier function [37], such that:

$$W_i = \varphi_i\ .f\left(\alpha, \beta, \varphi_{UE,i}\right). f(\alpha, \beta, \varepsilon) \tag{7}$$

where by definition [37],

$$f\left(\alpha, \beta, \varphi_{UE,i}\right) = \frac{1}{\sqrt{|B|}} \sum_{j=1,q=1}^{n_1,n_2} e^{-j\,||\alpha_{i,j} - \beta_{i,q}||} \left(e^{j\,||\alpha_{i,j} - \beta_{i,q}||} . \varphi_{UE,i}\right) \tag{8}$$

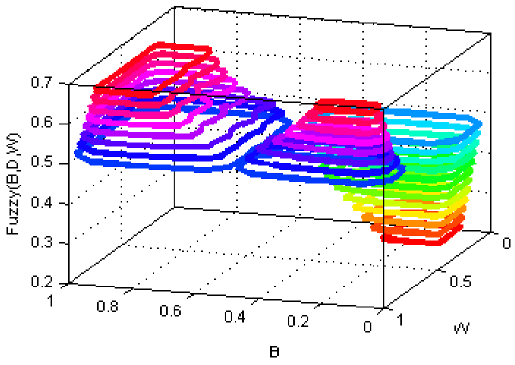
and

$$f(\alpha, \beta, \varepsilon) = \begin{cases} 1, if\ residuals\ are\ unavailable \\ \frac{1}{\sup\limits_{0<j\leq q} \varepsilon} \sum\limits_{j=1,q=1}^{n_1,n_2} ||\alpha_{i,j} - \beta_{i,q}||, q > j, if\ residuals\ (\varepsilon)\ are \\ non-localized \\ \frac{1}{\sqrt{|B|}} \sum\limits_{j=1,q=1}^{n_1,n_2} e^{-j\,||\alpha_{i,j} - \beta_{i,q}||} \left(e^{j\,||\alpha_{i,j} - \beta_{i,q}||} . \varepsilon_{i,j}\right), i\ fresiduals\ (\varepsilon) \\ are\ localized \end{cases} \tag{9}$$

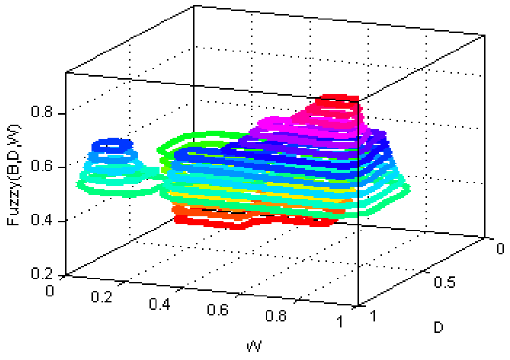
Here, the localization of residuals refers to the identification of errors with respect to a behavior rule. The above formulations form the base of fuzzy evaluations and help decide whether the available values for a behavior rule make it verifiable or not.

Different mechanisms are used to generate normalized inputs for B, D and W to formulate the fuzzy sets for generating inference rules. To map B, it is replaced by the periodicity of the behavior rule (B'), which is normalized using $\frac{B' - \min(B')}{\max(B') - \min(B')}$. D and W are evaluated for Bayesian belief, such that their normalized values are given by $D^{(N)}{}_i = \frac{L(D_{U,i}).P(D_E)}{P(D_{U,i})}$ and $W^{(N)}{}_i = \frac{L(W_{U,i}).P(W_E)}{P(W_{U,i})}$, respectively, where L and P denote the likelihood and the probability, respectively. Under relaxed constraints and low-complex evaluations, these are obtained as $\overbrace{0 < k \leq j}^{sup}_{0 < m \leq q} [\frac{D_{U,k}}{\max(D_U)} . \frac{D_{E,m}}{\max(D_E)}]$ and $\overbrace{0 < k \leq j}^{sup}_{0 < m \leq q} [\frac{W_{U,k}}{\max(W_U)} . \frac{W_{E,m}}{\max(W_U)}]$.

Based on the expert's recommendations as well as the devices' readings, limits are set for the membership values observed for the fuzzy set, thus inferencing an output for taking a decision on V. For this, Low, Medium, and High are marked for B', and Very Low, Low, Medium, High, and Very High are marked for both D and W. Usually, the value range is based on beliefs; however, in the proposed approach, these are driven by the max-mean approach. Therefore, the limits on the membership values (0, 1) for *B'* are (0, 0.25, 0.5), (0.25, 0.5, 0.75), and (0.5, 0.75, 1.0) and for *W* and *D* are (0, 0.125, 0.25), (0.15, 0.25, 0.35), (0.30, 0.45, 0.60), (0.55, 0.7, 0.85) and (0.75, 0.875, 1.0). Now, using inference criteria based on the urgency of a behavior rule, the following fuzzy-observations are attainable for *V*: Low, Medium, High, Very High and Extreme with membership values of the order, (0, 0.2, 0.4), (0.35, 0.5, 0.65), (0.6, 0.7, 0.8), (0.75, 0.825, 0.9) and (0.85, 1.0).Fig. 4 shows how to trace inference rules for their mapping.



(A)Fuzzy (B, D, W) vs. W and B.



(B) Fuzzy (B, D, W) vs. W and D.

Fig.4: A graphical illustration of the fuzzy observations with variations in fuzzy function with respect to *B', W, D.* The plots help to understand the impact of rules on the observation of identifying the verifiability for given behavior rules .In both diagrams, the interest is given to a V=Fuzzy (B, D, W) value higher than the medium value defined by the expert or the user.

A general procedure for fuzzy evaluations involves converting fuzzy observations to crisp values for finalizing the value of a function under evaluation. However, in the given system, the primary concern is about the belief to consider the verification of a behavior rule. Thus, a Boolean variable is assigned directly to the fuzzy observations, such that any value leading to a medium or lower is marked with 0, and 1 otherwise. Now, based on these, the final set of behavior rules is obtained to further check for correctness.

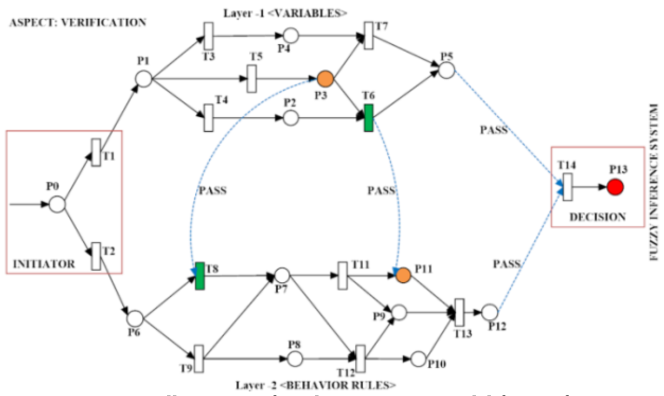


Fig.5: An illustration of a 2-layer HCAPN model for verifying behavior rule correctness.

Let B_d, W_d, and D_d, be the derived sets for the evaluated behavior rules, which are to be formally checked for their correctness. To handle this task, HCAPN's 2-layer statistical format is used (Fig. 5), which is a variant of the original HCAPN. At first, the system is accounted for the number of places, passes, and association for building transitions. Later, the number of tokens required to evaluate the reachability of HCAPN are generated. Finally, the statistical evaluations of HCAPN help verify the correctness of the shortlisted (decided) behavior rules. The details are as follows:

1. ***Number of layers:*** The initial HCAPN model [33] is efficient in resolving multi-variable dependencies as well as support variable evaluations and formal analysis of network entities. However, the initial version accounts each entity into the place and builds a transition for each of them leading to a complex scenario that is heavyweight on memory as well as run-time. The conditions fail when the real-time evaluations involve undecided variables accounting verification. Thus, to make it lightweight, we adopt a 2-layer HCAPN model with statistical decidability, which reduces the complexity by lowering the number of places, passes, transitions, and tokens for generating the required observations.
2. ***Number of places:*** Two sets of places, $\mathrm{N}_{\mathrm{U}}^{\mathrm{P}}$ and $\mathrm{N}_{\mathrm{L}}^{\mathrm{P}}$, for the lower layer and the upper layer of HCAPN, respectively, are decided based on the number of variables and the number of behavior rules. All tracking variables, ϑ_U,are marked as places in the HCAPN's upper layer and all behavior rules,B_d, are taken for places in the lower layer, such that:
$$\mathrm{N}_{\mathrm{U}}^{\mathrm{P}} = \{\vartheta_U \mid \vartheta_U > 0, \vartheta_U \text{ is the variables formulating } B_d\},$$
$$\mathrm{N}_{\mathrm{L}}^{\mathrm{P}} = \{x \mid x \epsilon B_d\} \qquad (10)$$
3. ***Number of transitions:*** The transitions for the upper layers involve the evaluation formulas using the variables from the places and are represented by a set $\mathrm{T}_{\mathrm{U}}^{\mathrm{P}}$ (e.g. 3 for ABI 1 - Tables 1-5). In the lower layer, the transitions are marked by security aspects, which are denoted by a set $\mathrm{T}_{\mathrm{L}}^{\mathrm{P}}$ (e.g. 13 for given behavior rules - Tables 1-5), such that,

$$\mathrm{T}_{\mathrm{U}}^{\mathrm{P}} = \{a|\ a \text{ is the equation involving variables from places}\},$$
$$\mathrm{T}_{\mathrm{L}}^{\mathrm{P}} = \{b|b \text{ is the referral aspects}\} \qquad (11)$$

The referral aspects can be any property, condition or additional rules. In this work, the referral aspects refer to security aspects, which are accounted for based on the behavior rules for devices in a CPS.

4. ***Number of passes:*** The number of passes is an integral part of HCAPN which provides flexibility of multi-party verifications without rebuilding new Petri nets for the dependent variables. In this work, the number of passes is not directly generated based on the rules of places/passes. Rather, five main strategies are used which are further based on two main properties, namely, active passing and passive passing. In the passive passing, the number of passes between the upper and the lower layer Petri nets is pre-decided and any additional inclusion of passes or change in transition leads to the fresh derivation of HCAPN. In the active passing, the number of passes is decided on-demand; however, such a situation generates an optimization problem which accounts for settling a tradeoff between the excessive passes and operational time. The excessive passes can lead to far more accurate results even for a complex scenario, but at a cost of time and memory. While keeping time in constraint, the number of passes can still be functional, but only under certain conditions leading to the verification of strict behavior rules only. Irrespective of the mode of operation, the following solutions can be used for deciding the number of passes in the 2-layer statistical HCAPN model for behavior rule verification:
   a) ***In case of loops:*** The active passing can especially be used to remove loops during evaluations of behavior rules. Any adversary, who tends to avoid the verification to prevent its detection as misbehavior, can try to fool the system by sending similar types of data from the same devices again and again. This may result in a loop over a particular variable as the behavior rule for the verification remains the same. To avoid such a situation, the context can be shifted while avoiding loops over the involved places and transitions, thereby preventing missing verification for a non-included behavior rule.
   b) ***In case of relationships:*** In case of a direct relationship between the variables and behavior rules, a pass is needed between the two layers of Petri nets. However, the choice of positioning of passes and extending a pass from a particular variable to a particular behavior rule is again an issue related to optimization.
   c) ***In case of deviation in observations:*** There are certain situations, where the system generates a large number of false positives because of numerous connectivity or excessive tokens, which lead to a deviation of the system from generating desired results. In such a case, the passes are marked between the variables and the behavior rules to avoid false positives. Moreover, in such situations, the passes can be considered from formulae from the upper layer to the aspects of the lower layer via additional places.
   d) ***In case of high operational time:*** As expressed in the first part, high operational time for evaluating the correctness of behavior rules has to be avoided in a solution pertaining to the identification of misbehavior in a CPS. Thus, additional places and transitions need to be removed and new passes must be generated to increase the performance without compromising the verification procedure.
   e) ***In case of large traversals of places:*** This is similar to loops, the places which are traversed several times must have a common variable or behavior rule, which can be overlooked, however, only at the cost of false negatives. In case the system shows an increase in false positives, such traversals should be allowed even if the computational time increases. The time cost in such a situation can be saved by skipping variables based on periodicity.

In a general HCAPN model, the number of layers may vary, so is the number of passes. However, there is an upper limit for the number of passes to avoid additional overheads. In the case of a 2-layer HCAPN model, the number of passes is marked by the general distribution of the number of variables and the behavior rules. The upper limits remain at *X(X-1)/2*, where *X* denotes the sum of the places and tokens. However, such a situation causes more tokens and hinders the timely verification of the behavior rule correctness. To resolve this, a law of *K* by *K* is formulated which means finding the value of *K* such that *K* variables are always in demand by exactly *K* behavior rules. The value of *K* should be minimized subject to the verification of behavior rules. Additionally, the value of *K* should also be maximized subject to the minimization of the evaluation time. The value of *K* remains to be the number of passes required for building the 2-layer HCAPN model. To solve this, the Walsh matrix approach [38] is used, according to which, the number of sign changes between the slots refers to the requirements of the passes between the two layers. The sign changes are derived based on the occurrences of variables during a fixed slot. Thus, mathematically, number of passes can be expressed as:

$$K = \begin{cases} S_t^{(C)}\left(W\left(Z\left(2^{\vartheta_U}\right)\right)\right), t_1 \le t \le t_2, t_2 - t_1 \ne 0, t_1 > 0, S_t^{(C)}(.) \ne 0 \\ Q_t^{(C)}\left(W\left(Z\left(2^{|B_d|}\right)\right)\right), S_t^{(C)}(.) = 0, t_1 \le t \le t_2, t_2 - t_1 \ne 0, t_1 > 0, \\ \dfrac{common(\vartheta_U, B_d)}{2}, B_d = inconsistent, S_t^{(C)}(.) = 0, Q_t^{(C)}(.) = 0 \\ 1, otherwise \end{cases} \qquad (12)$$

where M is the Walsh matrix derived over the Hadamard matrix (Z) for the number of available variables, $S_t^{(C)}$ and $Q_t^{(C)}$are the functions tracking the change in signs for the recorded variables and behavior rules, respectively. The function $common(\vartheta_U, B_d)$ calculates the number of variables with a common interest for the behavior rules in the lower layer. It is to be considered that the total number of passes should not be allowed to go beyond the mesh structure ($\frac{K(K-1)}{2}$) and it should be consistent with the rule of passes and places followed by the original HCAPN model [33].

5. ***Number of tokens:*** The number of tokens is driven by the operational requirements of the 2-layer HCAPN model. For initial consideration, each behavior rule as well as each variable is provided with a single token, whose requirements depend on the number of occurrences in the transition-formulae and the security aspects, respectively. The periodicity of a behavior rule has a definite impact on the number of tokens to be set for evaluating the inputs for a device. Thus, for verification, the number of tokens is set as$B' \frac{\sum_{j=1}^{\vartheta_U} \alpha_j}{|B_d|}$.
6. ***Deciding the input and the outputs:*** The number of inputs is based on the data read for the embedded IoT device involved. The number of inputs initially is set to that needed by the first behavior rule. The choice after the initiation depends on the user, i.e., the 2-layer HCAPN model can be operated in a top-bottom or bottom-top approach. It can also be initiated in both directions to confirm the reachability of all the places as well as for checking the firing of all transitions. Note that reachability of all places and firing of all transitions also depends on the reliability of the system. For the outputs, the place formed at last during a given slot is taken as an output. Moreover, in any instance, 2-layer HCAPN can be halted, and, unlike traditional Petri nets, any place can be marked as an output.
7. ***Deciding the aspect and the context:*** The aspect refers to the feature of HCAPN, which is set as "verification" for the tracked behavior rules and the context refers to an event which causes a transition to fire. Multiple transitions can have the same context and each context depends on the number of behavior rules and the variables which form these behavior rules. The firing of the transition is dependent on the tokens with the variables in the upper layer and the tokens with the behavior rules in the lower layer. The firing of transition is based on the requirements of the context and the availability of variable information from the device under surveillance for behavior verification. The firing can also be predicted similar to the general Petri nets, provided that accurate transition matrices are formed for the tracked behavior rules. The context in the proposed set up is marked by C<index> and the aspect helps to understand the state of the HCAPN model, i.e., whether it is in the verification stage or the prediction stage. Moreover, aspect can also be used to identify if the system is evaluating the results through comparison or ignoring the available inputs.
8. ***Deciding Supervisory HCAPN***: The supervisory HCAPN is the experts' observations, which are based on a prediction as well as the flow of data available from the embedded device in the CPS. The decisive supervisory HCAPN helps to understand the deviation of the system in successfully verifying the behavior rules. Moreover, it is used as learning for the system, which helps to ignore pre-decided/pre-evaluated behavior rules, thereby saving computations as well as overheads of misbehavior detection.
9. ***Observing G=Fuzzy(H(.)):***Once all the above requirements are satisfied, the system is ready to verify that the behavior rules generated are correct and cover all the threats (or satisfy the security requirements) and that the resulting safe and unsafe states are complete and are generated correctly. For this, by definition of HCAPN, we have G=Fuzzy(H(A1, A2,A3, A4, A5, A6, A7, A8, A9)), where A1-A9 are the metrics of the HCAPN model, such that, A1 is the set of places ($\mathrm{N}_\mathrm{U}^\mathrm{P} + \mathrm{N}_\mathrm{L}^\mathrm{P}$), A2 is the set of transitions ($\mathrm{T}_\mathrm{U}^\mathrm{P} + \mathrm{T}_\mathrm{L}^\mathrm{P}$),A3 is the set of connections between A1 and A2, A4 is the set of passes (A4=$\{K|\ K > 0\}$), A5 is the set of type of passes, which is marked with the number of tokens for evaluation in BRIoT, A6 is the set of context conditions (A6=$\{C|\ C\ \in ABIs\ derived\ from\ the\ behavior\ rules\}$), A7 denotes the aspect, A8 is the number of layers, which is 2, and A9 is the set of output places, which is 1. The verification is done based on correctness properties, which are then fed into the fuzzy inference system for generating a Boolean output to check the correctness as well as the applicability of the behavior rules. The details of the properties used for verification [33] are as follows:
   a) ***Isolation:*** It refers to the places which are left alone and does not have any connectivity within the HCPAN based on the given behavior rules. The isolation is tested in the upper as well as the lower layer of HCAPN by accounting A1 without A2 and A6 associated with it. Mathematically, it can be written as:

$$S_I^{<Layer>} = 1 - \frac{1}{\left(\left(\frac{\left|\mathrm{N}_{<Layer>}^{\mathrm{P}}\right|'}{\left|\mathrm{N}_{<Layer>}^{\mathrm{P}}\right|}\right)^2 - 1\right)}, |\mathrm{N}_{<Layer>}^{\mathrm{P}}{}'| \neq |\mathrm{N}_{<Layer>}^{\mathrm{P}}| \quad (13)$$

where the prime operator($'$) denotes the non-functional places in the HCAPN model. $S_I^{<Layer>}$ is for an individual layer, subject to the identification of error only in the variables (upper layer) or behavior rules (lower layer). One can also compute isolation collectively based on A1 as:

$$S_I^{<collective>} = 1 - \frac{1}{\left(\left(\frac{|A1'|}{|A1|}\right)^2 - 1\right)}, |A1'| \neq |A1| \quad (14)$$

b) ***Non-Reachability:*** Non-reachability refers to the inaccessibility of places in the given HCAPN and can be expressed as a counter-value of reachability. In the given mode, the reachability can be determined by accounting the deflections in the number of transitions which are fired and the number of tokens retrieved at each place, such that:

$$R_B = \left(1 - \frac{1}{\left(\left(\frac{|A2'|}{|A2|}\right)^2 - 1\right)}\right)\left(1 - \frac{1}{\left(\left(\frac{|A5'|}{|A5|}\right)^2 - 1\right)}\right), \quad |A2'| \neq |A2|, |A5'| \neq |A5|. \tag{15}$$

The smaller value of reachability means higher non-reachability and vice versa. Similarly, reachability can account for the individual layer based on the location of the output. Moreover, non-reachability is also checked as part of the transition matrix by accounting the negatives for tokens, which refers to the congestion/cycle/conflict and is against the policies of a Petri Nets.

c) ***Dependability:*** It defines the relationship between the variables and decreases when more variables are in the behavior rules without the prior knowledge. It is difficult to predict any relation between the variables and the existing behavior rules without any library, which is not a case with real-time evaluations. Thus, dependability decreases with an increase in the variables with non-availability of relationships with any of the existing behavior rules. Based on this, the dependability of the 2-layer HCAPN model can be given by:

$$E_D^P = \left(\frac{|B_d| + |B_{d,x}|}{J1 + J2}\right) \tag{16}$$

where

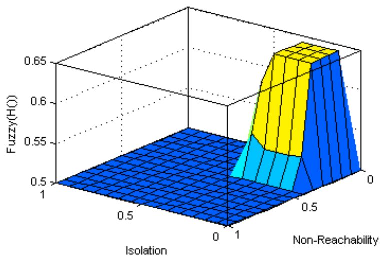
$$J1 = |B_d|\left(\left(\frac{1}{|B_d|}\sum_{i=1}^{|B_d|}(\vartheta_i - \bar{\vartheta})^2\right) + \left(\bar{\vartheta} - \left(\frac{\bar{\vartheta} + \vartheta_x}{2}\right)\right)^2\right) \tag{17}$$

and

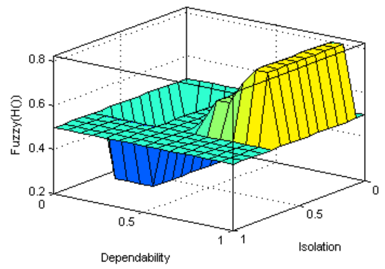
$$J2 = |B_{d,x}|\left(\left(\frac{1}{|B_{d,x}|}\sum_{i=1}^{|B_{d,x}|}(\vartheta_{x,i} - \bar{\vartheta_x})^2\right) + \left(\bar{\vartheta_x} - \left(\frac{\bar{\vartheta} + \bar{\vartheta_x}}{2}\right)\right)^2\right). \tag{18}$$

Here, $|B_{d,x}|$ is the number of behavior rules with new variables, $\bar{\vartheta}$ is the average number of variables in each behavior rule, ϑ_x is the number of new variables, and $\overline{\vartheta_x}$ is the average number of variables in the new behavior rules.

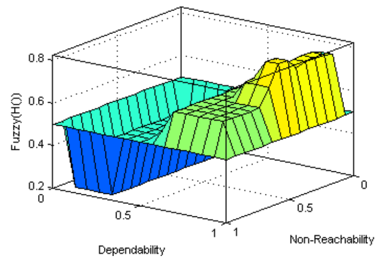
Now, the isolation, non-reachability and dependability are normalized by using similar formulations as used for *B'*. Considering this, the fuzzy inference for verification of behavior rules is formulated which gives verified or non-verifiable as an output. It can be expanded to check the correctness of variables as well as context used to relate variables and the behavior rules. The fuzzy inference rules and impact of properties on the decision are illustrated by Fig. 6.



A. Fuzzy(H(.)) vs. isolation and non-reachability.



B. Fuzzy(H(.)) vs. isolation and dependability.



C. Fuzzy(H(.)) vs. non-reachability and dependability.

Fig.6: A graphical illustration of fuzzy observations for determining the correctness of behavior rules based on the 2-layer statistical HCAPN model. The function Fuzzy (H (.)) is contrasting to Fig. 4 even with different variations in non-reachability, isolation and dependability. This depicts the role of the statistical model in the verification process. It also verifies that the identification of correctness is based on the expert's module as well as the accurate formation of a 2-layer HCAPN model.

In the fuzzy-based correctness evaluations, isolation and non-reachability (lower value on reachability means higher non-reachability and vice versa) are marked with low, medium, and high membership functions with values (0, 0.2, 0.4), (0.25, 0.5, 0.75), (0.4, 0.7, 1), and (0, 0.2, 0.4), (0.3, 0.45, 0.6), (0.5, 0.75, 1), respectively. Dependability is marked with very low, low, medium, high, and very high with values (0, 0.1, 0.2), (0.15, 0.25, 0.35), (0.30, 0.45, 0.60), (0.55, 0.7, 0.85), and (0.75, 0.875, 1), respectively. The outputs are marked as low, medium, sensitive, correct, strictly correct with values (0, 0.2, 0.4), (0.35, 0.5, 0.65), (0.54, 0.65, 0.75), (0.7, 0.825, 0.95) and (0.85, 1, 1), respectively. The decision on correctness can be attained based on the following conditions:

$$G = \begin{cases} 0, if\ Incorrect, Low \leq Fuzzy(H(.)) \leq medium, sensitive \\ 1, if\ Correct, sensitive \leq Fuzzy(H(.)) \leq strictlycorrect \end{cases} \tag{19}$$

In addition to preliminary observations for correctness, defuzzification can be used to evaluate the model on crisp values. Irrespective of that, the results of the correctness of behavior rules will be same as pointed out in (19). Once these verifications are done, the system can be operated towards the identification of misbehavior in a CPS. The details of these procedures for verification and correctness of behavior rules are presented in Algorithms 1 and 2.

ALGORITHM 1: Verifiability and correctness of behavior rules

```
Input: B, W, I, $\gamma_U$, $\gamma_E$ [E.g. Table 5], fuzzy range and membership values
Output: V= True/False (0/1), G=True/False (0/1)
1.  While (I!=NULL)
2.      Set system and initiate operational profiler
3.      Obtain values for B ( as shown in Table 5) from experts
4.      While (Read(B)==True)
5.          Fetch ABI from experts and users
6.          Set Value for B
7.          Calculate D as in (4) using dependants from (1)~(3)
8.          If (W==unavailable)
9.              Calculate W as in (7) using dependants from (5)~(9)
10.         End If
11.         Invoke Fuzzy(B,D,W) with predefined rules
12.         Obtain V
13.         If ( V==1)
14.             Store $B_d$, $W_d$, and $D_d$
15.             G=Initiate HCAPN Tool → HCAPN($B_d$, $W_d$, and $D_d$)
16.             If(G==1)
17.                 ABI is verifiable and correct.
18.             Else
19.                 ABI is verifiable but incorrect
20.             End If
21.         Else
22.             Exit(-1) // return non-verifiable behavior rule
23.         End If
24.     End While
25. End While
For observations: Vary $\varepsilon$, W, $\gamma_U$, $\gamma_E$, $f(\alpha,\beta,\varepsilon)$
```

ALGORITHM 2: G=HCAPN (H (.))

```
Input: $B_d$, $W_d$, and $D_d$, fuzzy range and membership values
Output: Return G
1.  While ($B_d$!=NULL)
2.      Set number of layers = 2
3.      Lower layer places=behavior rules – follow $\mathbf{N}_L^P$ in (10)
4.      Upper layer places=variables– follow $\mathbf{N}_U^P$ in (10)
5.      Set transitions $\mathbf{T}_U^P$ and $\mathbf{T}_L^P$ – follow (11)
6.      Set passes between $B_d$ and $\boldsymbol{\vartheta}$
7.      Resolve loops, relationships, large traversals
8.      Set tokens and fix input and output places
9.      Build HCAPN
10.     While (Observation==True)
11.         Calculate Isolation as in (14)
12.         Calculate Non-reachability as in (15)
13.         Calculate Dependability as in (16)
14.         Normalize values of (14) ~ (16) and store H (.)
15.         Invoke Fuzzy (H (.))
16.         Obtain G and return
17.     End While
18. End While
```

B. AUTOMATIC TRANSFORMATION OF A BEHAVIOR RULE SET TO A STATE MACHINE FOR FEEDBACK-BASED MISBEHAVIOR DETECTION

We transform behavior rules to a C-language state machine labeled with safe and unsafe states, against which good (normal) and bad (malicious) behaviors of the IoT device can be statistically characterized. Suppose that there are n ABIs derived from the corresponding n behavior rules. Then all n ABIs (derived from the behavior rules) are combined in disjunctive normal form (DNF) into a Boolean expression for misbehavior detection. This means that a violation of any ABI Boolean variable (meaning taking a value of 1) indicates a violation of the corresponding behavior rule. The resulting state machine has a total of 2^nstates, out of which only one is a safe state (when all ABI Boolean variables take the value of 0).

However, environmental and operational conditions may change rapidly causing output variations even if an IoT device follows the behavior rules. Thus, it is necessary to model such variations for effective misbehavior detection. The reference point is the state machine generated (a DNF Boolean expression) as describe above which resembles an expert's observations. This helps track the feedback for each ABI (and hence each behavior rule) and understand the limits up to which the variation in the ABI can be treated as normal behavior. To model this, ε_X^F is used as an accumulated feedback variable, formulated as follows:

$$F\text{-}DNF = DNF \rightarrow \varepsilon_X^F = feedback(\text{Misbehavior Range (“ABI”)}) \quad (20)$$

where *F-DNF* is the feedback on DNF for an ABI, and the misbehavior range is marked as the feedback value. The feedback can be treated as a residual for determining new variables in the tracked behavior rule.

Let $Q(B_d, U, Y)$be the bipartite graph between the behavior rules and the set U containing all the readable variables (ϑ), such that $|B_d| \leq |U|$. The set Y contains the feedback variable (ε_X^F) and also forms the edge between the behavior rules and the variables. It is accounted for defining the *F-DNF* as well as for determining the misbehavior of an IoT device subject to its adjustment to fit into the network requirements. The graph operates for each connection between the rules and the variables and accumulates $\varepsilon_X^F$to check any malicious activity. To form an efficient feedback-based misbehavior detector, the reference points are required, which should not cause any excessive computation and must not keep on iterating for identifying changes in the same variable. A solution to such a problem can be sought from the amalgamation of bipartite graphs and the Barycentric coordinate theory for determining the center of mass. Both mechanisms are adopted in our proposed misbehavior detection method to help identify the misbehavior activity with feedback. A visualization of this process can be observed in Fig. 7.

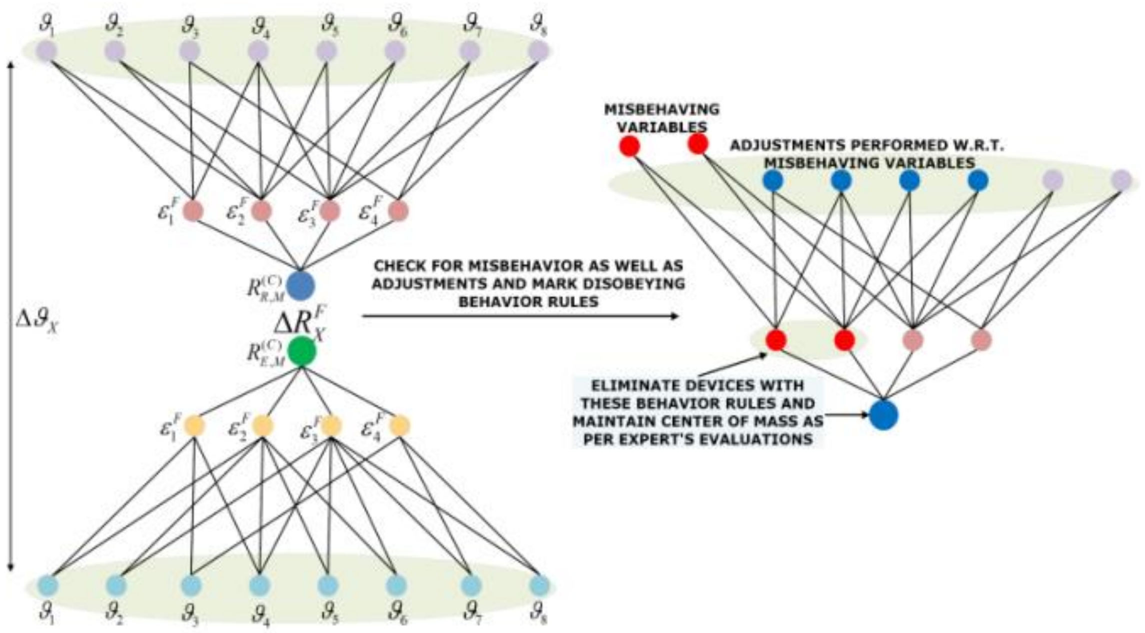

ϑ_1 ϑ_2 ϑ_3 ϑ_4 ϑ_5 ϑ_6 ϑ_7 ϑ_8
ε_1^F ε_2^F ε_3^F ε_4^F
$R_{R,M}^{(C)}$
ΔR_X^F
$R_{E,M}^{(C)}$
$\Delta\vartheta_X$
CHECK FOR MISBEHAVIOR AS WELL AS ADJUSTMENTS AND MARK DISOBEYING BEHAVIOR RULES
MISBEHAVING VARIABLES
ADJUSTMENTS PERFORMED W.R.T. MISBEHAVING VARIABLES
ELIMINATE DEVICES WITH THESE BEHAVIOR RULES AND MAINTAIN CENTER OF MASS AS PER EXPERT'S EVALUATIONS

Fig.7: Illustration of the Bipartite-based center of mass mechanism for misbehavior detection.

Based on this misbehavior detection method, the Barycentric coordinate for the center of mass for the misbehavior tracking for an IoT device in a CPS can be given as:

$$R_{R,M}^{(C)} = \frac{1}{\sum_{i=1}^{|B_d|} w_i} \sum_{j=1}^{|B_d|} w_j . p_j \text{ }, w \in W_d \tag{21}$$

where

$$p_j = \frac{\sum_{m=1}^{\vartheta_j} \gamma_m . Val(\vartheta)_m}{\sum_{k=1}^{\vartheta_j} \gamma_k} \tag{22}$$

Similar values are observed for expert's observations and marked as $R_{E,M}^{(C)}$.The feedback for observable behavior rules and the difference in the value of Barycentric coordinates for misbehavior detection can be calculated as:

$$\varepsilon_X^F = (w_X . p_X)_{expert} - (w_X . p_X)_{observed} \tag{23}$$

and

$$\Delta R_X^F = R_{E,M}^{(C)} - R_{R,M}^{(C)}, \Delta R_X^F \geq 0 \tag{24}$$

where p_X is derived from (22)for x. Evaluating these, the misbehavior can be marked as:

$$M_b = \begin{cases} 1, \Delta R_X^F \geq \Delta TH \\ 0, \text{ } Otherwise \end{cases} \tag{25}$$

where ΔTH marks the observational thresholds for all the behavior rules. It can be fixed by an expert or can be fixed as a value above which more than y% of behavior rules disobeys the principle of accuracy. Once M_b attains a value of 1, it is certain that there is a high probability of misbehavior, but the variables primarily causing this abnormality are still unclear and may affect the other behavior rules, which are dependent on it. To quantify, select a subset of behavior rules for which:

$$\varepsilon_X^F \geq \left| \sqrt{\frac{1}{|B_d|} \sum_{j=1}^{|B_d|} \left(w_j . p_j - \overline{w.p}\right)^2}_{observed} - \sqrt{\frac{1}{|B_d|} \sum_{j=1}^{|B_d|} \left(w_j . p_j - \overline{w.p}\right)^2}_{expert} \right| \tag{28}$$

and parse each behavior rule by following the importance of its variables (γ), such that, for each behavior rule, the alterations in the ith variable can be calculated trivially as $\Delta\vartheta_i = Val(\vartheta_i)_{expert} - Val(\vartheta_i)_{observed}$. For a decision on adjustments, a local Barycentric coordinate observed by an expert can be evaluated as:

$$L_{X,R,M}^{(C)} = \frac{1}{W_{d,X}} \sum_{j=1}^{|\vartheta_X|} W_{d,X} . Val(\vartheta_j)_{observed} \tag{29}$$

and

$$L_{X,E,M}^{(C)} = \frac{1}{W_{d,X}} \sum_{j=1}^{|\vartheta_X|} W_{d,X} . Val(\vartheta_j)_{expert} \tag{30}$$

Based on these, the adjustments can be evaluated as:

$$\Delta L_A^{(C)} = L_{X,R,M}^{(C)} - L_{X,E,M}^{(C)} = \begin{cases} (\approx)0, Adjustment; check\ other\ variables \\ No\ Adjustments \end{cases} . \tag{31}$$

The above formulation checks if a behavior rule's overall coordinates remain the same or not. If these are the same, the device is not misbehaving but merely performing certain adjustments to suit dynamically changing environmental or operational conditions; otherwise, it is treated as misbehaving which requires immediate actions. The detailed steps of Feedback-based mechanism can be followed in Algorithm 3.

ALGORITHM 3: Feedback-based misbehavior detection

```
Input: $B_d$, $W_d$,, $D_d$, $\boldsymbol{Q(B_d, U, Y)}$, $\boldsymbol{\gamma}$, $\Delta \boldsymbol{TH}$
Output: $\boldsymbol{\varepsilon_X^F}$, $\Delta \boldsymbol{L_A^{(C)}}$, $\boldsymbol{M_b}$
1.  While ($B_d$!=NULL)
2.      Set experts inputs and check variables in traced $B$
3.      Define local weight w
4.      Define probability (p)  using (22)
5.      Perform steps 2, 3, and 4 for experts observations
6.      Calculate Barycentric coordinates ($\boldsymbol{R_{R,M}^{(C)}}$and $\boldsymbol{R_{E,M}^{(C)}}$) using (21)
7.      Calculate $\boldsymbol{\varepsilon_X^F}$ using (23) based on expert and observed values
8.      Calculate difference in Barycentric coordinates $\Delta \boldsymbol{R_X^F}$using (24)
9.      If ($\Delta \boldsymbol{R_X^F} \geq \Delta \boldsymbol{TH}$)
10.         $\boldsymbol{M_b}$=1
11.         Diff=$\left| \sqrt{\frac{1}{|B_d|}\sum_{j=1}^{|B_d|}(w_j.p_j - \overline{w.p})^2}_{observed} - \sqrt{\frac{1}{|B_d|}\sum_{j=1}^{|B_d|}(w_j.p_j - \overline{w.p})^2}_{expert} \right|$
12.         If ($\boldsymbol{\varepsilon_X^F} \geq \mathbf{Diff}$)
13.             Quantification= true
14.             $\Delta\boldsymbol{\vartheta_i} = \boldsymbol{Val(\vartheta_i)_{expert}} - \boldsymbol{Val(\vartheta_i)_{observed}}$
15.             Calculate $\boldsymbol{L_{X,R,M}^{(C)}}$ using (29)
16.             Calculate $\boldsymbol{L_{X,E,M}^{(C)}}$ using (30)
17.             If($\Delta\boldsymbol{L_A^{(C)}}(= \boldsymbol{L_{X,R,M}^{(C)}} - \boldsymbol{L_{X,E,M}^{(C)}}$)==0)
18.                 Device is adjusting, check other variables
19.             Else
20.                 No adjustments, mark misbehavior
21.             End If
22.         Else
23.             Quantifications = false
24.             Exit(-1)
25.         End If
26.     Else
27.         $\boldsymbol{M_b}$=0
28.         Exit(-1)
29.     End If
30. End While
```

For observations: **Vary** $\boldsymbol{R_{R,M}^{(C)}}$ and $\boldsymbol{R_{E,M}^{(C)}}$ as per the behavior rules, p, $\boldsymbol{\gamma}$ and $\Delta \boldsymbol{TH}$

ALGORITHM 4: Lightweight statistical analysis

```
Input: T, ρ, $B_d$, $W_d$,, $D_d$, steps $t_n$, $\boldsymbol{\gamma}$
Output: $\boldsymbol{\theta}$,  $\boldsymbol{\omega_g}, \boldsymbol{\xi(\omega_g)}, \boldsymbol{\omega_g^{(O)}}$, $\boldsymbol{\xi(\omega_g^{(O)})}, \boldsymbol{FP}, \boldsymbol{FN}$
1.  While (t<= T)
2.      Calculate compliance constant ρ for given instance
3.      Calculate $\Delta \boldsymbol{L_A^{(C)}}$ using (29)~(31)
4.      Calculate $\Delta \boldsymbol{R_X^F}$ using (24) and $\boldsymbol{R_{R,M}^{(C)}}$ using (21)
5.      Use $\boldsymbol{\varepsilon_X^F}$ from (28) and Algorithm 3
6.      Set $\boldsymbol{T(\theta)}$
7.      Calculate $\boldsymbol{\theta}$ using (32)
8.      Perform predictive evaluations for $\boldsymbol{\theta_p}$using (35)
9.      Calculate $\boldsymbol{\omega_g}$ using (37)
10.     Calculate $\boldsymbol{\xi(\omega_g)}$ using (38)
11.     Calculate $\boldsymbol{\omega_g^{(O)}}$ using (39)
12.     Calculate $\boldsymbol{\xi(\omega_g^{(O)})}$ using (40)
13.     Calculate adjustments and record $\boldsymbol{\psi}$
14.         If ($\boldsymbol{\psi}$== traceable)
15.             If($\boldsymbol{\xi(\omega_g^{(O)})} \geq \boldsymbol{\xi(\omega_g)} + \boldsymbol{\psi}$)
16.                 Record FP
17.             Else if ($\boldsymbol{\xi(\omega_g^{(O)})} \leq \boldsymbol{\xi(\omega_g)} - \boldsymbol{\psi}$)
18.                 Record FN
19.             End If
20.             Else
21.                 Mark as miss and continue
22.             End If
23.         t=t+100 // 10 steps in this case
24. End While
```

For observations: **Vary** ρ,$\boldsymbol{T(\theta)}$, Shuffle rules to change $\boldsymbol{R_{R,M}^{(C)}}$ and $\boldsymbol{R_{R,M}^{(C)}}$

C. LIGHTWEIGHT RUNTIME COLLECTION OF COMPLIANCE DEGREE DATA AND STATISTICAL ANALYSIS

Unlike anomaly detection which frequently requires heavy resources to profile/learn anomaly patterns, our behavior rule specification-based data collection process is lightweight. By using the transformed state machine, we only need to periodically monitor if a target IoT device is in safe or unsafe states. The periodic evaluations are similar to the recording of behavior rules with a periodicity *B'*. Now, considering that the overall evaluations are bounded by timestamps, t_1, t_2... t_n, the compliance degree data is denoted by θ, which has to be collected for each device for a given duration. The compliance degree of data is driven by the stiffness of the model and can be modeled using Wannier function of weight and the adjustment values of each device, such that:

$$\theta = \frac{1}{\rho\ tn}\sum\nolimits_{\mathrm{t1,t2...\,tn}}\left(\frac{1}{|B_d|}\sum\nolimits_{|B_d|}\frac{1}{f(W,\Delta R_X^F,\Delta L_A^{(C)})}\right)\rho_{\mathrm{t1,t2...\,tn}} \quad (32)$$

where ρ is the compliance constant, which is derived as a function of γ for all the behavior rules aggregated for a device under surveillance, such that $\rho = \frac{1}{|B_d|}\sum_{i=1}^{|B_d|}\gamma_i$. The function $f(W_d, \Delta R_X^F, \Delta L_A^{(C)})$ derives its value from the stiffness theory [39], such that:

$$f(W, \Delta R_X^F, \Delta L_A^{(C)}) = f(\Delta R_X^F, \Delta L_A^{(C)}) + \varphi\ .f(\alpha, \beta, \varphi_{UE}).f(\alpha, \beta, \varepsilon) \quad (33)$$

which traces feedback and availability of residual changes to:

$$f\left(W, \Delta R_X^F, \Delta L_A^{(C)}\right) = f\left(\Delta R_X^F, \Delta L_A^{(C)}\right) + \varphi\ .f(\alpha, \beta, \varepsilon_X^F)^2 \quad (34)$$

where $f(\alpha, \beta, \varepsilon_X^F)^2$ is derived from (9), and $f\left(\Delta R_X^F, \Delta L_A^{(C)}\right)$ can be set as a product of 0 and 1 based on their combinatorial outcome for a variation in Barycentric coordinate and adjustments. The compliance degree data can be predicted for a continuous interval as:

$$\theta_p = \frac{1}{\rho\ tn}\int \frac{1}{|B_d|}\sum\nolimits_{|B_d|}\frac{1}{f\left(\Delta R_X^F,\Delta L_A^{(C)}\right)+\varphi\ .f\left(\alpha,\beta,\varepsilon_X^F\right)_p{}^2}\rho_p(\rho, t)\ d\rho \quad (35)$$

where

$$f(\alpha, \beta, \varepsilon_X^F)_p = \frac{1}{\sqrt{|B|}}\sum\nolimits_{j=1,q=1}^{n_1,n_2} e^{-j||\alpha_{i,j}-\beta_{i,q}||}\left(e^{j||\alpha_{i,j}-\beta_{i,q}||}.\int \varepsilon_{P,X}^F(\varepsilon_X^F, t)d\varepsilon_X^F\right)(36)$$

The observed system model and compliance degree data are evaluated using Weibull Distribution [41] as it is difficult to predict the type of distribution of data from a set of IoT devices in a CPS. Moreover, Wannier formulations used in the weight calculations are true for pseudo-periodic behavior rules, as a device may not behave in a similar pattern throughout its operations. Furthermore, with predictive evaluations, Weibull distribution can be more specific and can take dimensions of any well-suited statistical model. To keep the entire process light-weighted, Weibull reliability is determined which operates over the Weibull formation of the Wannier function-based compliance degree data and also accounts for the false positives and false negatives focusing on the misbehavior detection of embedded IoT devices in a CPS. To model this, $W_d, \Delta R_X^F, \Delta L_A^{(C)}$as the instance-based value of θ are used for evaluating the cumulative reliability of the model and to specify its capacity in identifying the misbehavior of a device [41], such that:

$$\omega_g = \frac{\Delta R_X^F}{T(\theta)}\left(\frac{\mathrm{t1+t2+\cdots+tn}}{T(\theta)}\right)^{\Delta R_X^F - 1} e^{-\left(\frac{\mathrm{t1+t2+\cdots+tn}}{T(\theta)}\right)^{\Delta R_X^F}}, T(\theta) \neq 0, \Delta R_X^F \neq 0 \tag{37}$$

where $T(\theta)$ is the instance evaluating function which records the steps for which all the metric values are available based on the compliance degree of the data collected for a device. Here, ω_g is the Weibull PDF, based on which the reliability of the system can be modeled as [41]:

$$\xi(\omega_g) = e^{-\left(\frac{\mathrm{t1+t2+\cdots+tn}}{T(\theta)}\right)^{\Delta R_X^F}} \tag{38}$$

For actual observations, (37) and (38) are modeled for Wannier function, such that:

$$\omega_g^{(O)} = \frac{R_{R,M}^{(C)}}{T(W_d)}\left(\frac{\mathrm{t1+t2+\cdots+tn}}{T(W_d)}\right)^{R_{R,M}^{(C)} - 1} e^{-\left(\frac{\mathrm{t1+t2+\cdots+tn}}{T(W_d)}\right)^{R_{R,M}^{(C)}}}, R_{R,M}^{(C)} \neq 0, T(W_d) \neq 0 \tag{39}$$

and

$$\xi\left(\omega_g^{(O)}\right) = e^{-\left(\frac{\mathrm{t1+t2+\cdots+tn}}{T(W_d)}\right)^{R_{R,M}^{(C)}}} \tag{40}$$

Formulations in (37) to (40) are only used when the system shows a non-approximated value for ΔR_X^F.Such a situation leads to some false positives or negatives in misbehavior detection of an IoT device. To understand this, a limiting constant ($\pm\psi$) is derived, such that the false positives and negatives are identified as:

$$Output = \begin{cases} False\ Positive\ (FP), \xi\left(\omega_g^{(O)}\right) \geq \xi(\omega_g) + \psi \\ False\ Negative\ (FN), \xi\left(\omega_g^{(O)}\right) \leq \xi(\omega_g) - \psi \end{cases}. \tag{41}$$

The steps for lightweight statistical analysis are provided in Algorithm 4.

V. APPLYING BRIOT TO UAV CPS

In this section, the proposed BRIoT is applied to a UAV embedded in a UAV-CPS as in BRUIDS [18], which is used as a baseline model for performance comparison. Step-by-step descriptions are given to explain the application, including deriving the security requirements of a UAV device given its operational profile as input, deriving the threats that can violate the security requirements, generate the behavior rules, verifying the behavior rules are complete and cover all threats (with respect to the security requirements), performing the transformation from the behavior rules to a state machine for misbehavior monitoring, collecting runtime compliance degree data, conduct statistical analysis for misbehavior detection, and assessing detection accuracy in comparison with BRUIDS [18].

A. EXPERIMENTAL SETUP

We first describe the mandatory steps required to setup the system to be driven by the proposed BRIoT model.

1) Generation of Behavior Rules and Attack Behavior Indicators with Formal Verification

The first step is to specify the operational profile (or the mission assignment) of a UAV in a UAV-CPS. It specifies mission events according to the probabilities with which they are expected to occur during the operational phase of the UAV. Without loss of generality, a special type of UAV, a military UAV [11], is considered with the following combat mission operational profile during its lifetime:

Navigate to specified locations following specified routes, perform correct data routing and IDS functions, return correct and timely sensing data to the designated ground station only, conserve energy, and upon confirmation from an authority, launch a missile at a specified battlefield location target and return to the home airbase.

Given this operational profile as input, the security requirements of this UAV can be automatically derived as listed in Table 2 (please refer to Fig. 1 for the physical components inside this UAV device).

Table 2: UAV Security Requirements.

ID	Security Requirement
SR 1	The UAV must follow a specified route to reach a specified location
SR 2	The UAV must perform correct data routing functions
SR3	The UAV must perform correct IDS functions when serving as a monitor node, i.e., providing true recommendations
SR 4	The UAV must send correct and timely sensing data to a specified ground station only
SR 5	The UAV must ready a missile when it is at the specified battlefield location and upon an authorized command to fire, must fire the missile accurately
SR 6	The UAV must not be captured
SR 7	The UAV must consume energy only as needed so as not to jeopardize the mission

With the system requirements defined, it is relatively straightforward to identify the threats that will keep this UAV from accomplishing its mission, as listed in Table 3.

Table 3: UAV Threats.

ID	Threat
THREAT1	The UAV is not able to follow a specified route
THREAT2	The UAV is not able to perform correct data routing functions
THREAT3	The UAV is not able to perform correct IDS functions, i.e., not able to provide true IDS recommendations
THREAT 4	The UAV is not able to return correct sensing data
THREAT 5	The UAV is not able to return timely sensing data
THREAT 6	The UAV is not able to follow authorized commands
THREAT 7	The UAV is not able to read a missile when it is at the specified battlefield location
THREAT 8	The UAV is not able to fire a missile accurately
THREAT 9	The UAV takes off/lands from/to an enemy airbase
THREAT 10	The UAV sends data to sources other than the specified ground station
THREAT 11	The UAV unnecessarily consumes energy, making it unavailable for mission execution

Here it is noticeable that the threats do not make any assumption of the attack types (known or unknown). Threats 1 and 3-9 threaten integrity; threat 10 threatens the confidentiality, and threats 2 and 11threaten availability. One can assign a priority to a threat, thereby making one threat more critical than another. For this UAV, one may want to consider integrity > confidentiality > availability as the priority order. Correspondingly one can assign a behavior rule (to be described later) with a priority, thus making a behavior rule more critical than another. This can change the criticality associated with behavior rules and affect the standard by which a node is considered malicious.

Next, the behavior rules can be automatically derived for this UAV. Table 4lists the behavior set without priority order for simplicity. It also lists the security aspect (integrity, confidentiality, or availability) associated with each behavior rule. A behavior rule is typically derived from a threat because a threat specifying a negative event that can lead to an undesired outcome is just opposite to a behavior rule specifying a good behavior or a good event that can lead to the desired outcome. Consequently, it is straightforward to map a threat to a behavior rule(for example THREAT 1 in Table 3 leads to BR 1 in Table 4) for a negative event that has a single cause or source. However, a threat that is too generally specified (e.g., THREAT 11 in Table 3 about energy consumption) can have more than one cause or source for the negative event and can require several behavior rules to specify where good behaviors are to be monitored. Out of the 11 threats in Table 3, only THREAT 11 has more than one source or cause for the negative event, so THREAT 11 maps to BR 11 – BR 13 in Table 4 specifying several sources where excessive energy consumption occurs.

Table 4: UAV Behavior Rules

ID	Behavior Rule	Security Aspect
BR 1	Fly a specified route	integrity
BR 2	forward data packets	availability
BR 3	provide true recommendations	integrity
BR 4	produce accurate sensing data	integrity
BR 5	produce timely sensing data	integrity
BR 6	accept only authorized commands	integrity
BR 7	ready missile if at target	integrity
BR 8	fire missile accurately	integrity
BR 9	do not deploy landing gear if outside home airbase	integrity
BR 10	send data only to designated ground station	confidentiality
BR 11	do not send an exceptionally higher number of packets than necessary	availability
BR 12	use minimum thrust when loitering	availability
BR 13	do not emit exceptionally higher signal strength than necessary	availability

Table 5: UAV Attack Behavior Indicators in Conjunctive Normal Form.

ID	Attack Behavior Indicator	Context
ABI 1	$\lvert$Location–Planned Location $\rvert >\delta_{distance}$	C1
ABI 2	$\lvert$Trusted Node NPR–Trusted Node NPS $\rvert >\delta_{NPR-NPS}$	C2
ABI 3	Trusted Node Audit ≠ Monitor Node Audit	C3
ABI 4	$\lvert$(Trusted Node Data – Monitor Node Data)/Monitor Node Data $\rvert >\delta_{data}$	C4
ABI 5	$\lvert$Time Received Trusted Node Data –Time Received Monitor Node Data $\rvert >\delta_{time}$	C5
ABI 6	(Action≠ FIRE) ∧ (Command = AUTHORIZED)	C6
ABI 7	(Missile≠ READY) ∧ (Location = TARGET LOCATION)	C7
ABI 8	(Action= FIRE) ∧ (Outcome ≠ SUCCESS)	C8
ABI 9	(Gear = DEPLOYED) ∧ (Location ≠HOME AIRBASE)	C9
ABI 10	Report Site ≠HOME GROUND STATION	C10
ABI 11	$\lvert$(Trusted Node NPS – Monitor Node NPS)/Monitor Node NPS $\rvert >\delta_{NPS}$	C11
ABI 12	(Thrust >MINIMUM THRUST) ∧ (Status = LOITER)	C12
ABI 13	$\lvert$Trusted Node RSSI– Monitor Node RSSI $\rvert >\delta_{RSSI}$	C13

Table 5 lists 13 one-to-one “attack behavior indicators” (ABI 1 – ABI 13) in Conjunctive Normal Form (CNF), each being expressed as a Boolean expression to be evaluated true (1) or false (0), indicating whether the corresponding behavior rule is violated or not. When a Boolean expression is evaluated to true, the UAV is detected as misbehaving against the corresponding behavior rule. Here we note that each attack behavior indicator may have several (internal) state variables. For example, ABI 1 in Table 5 has two state variables, namely, Location and Planned Location.

The 1st attack behavior indicator (ABI 1 in Table 5) is that this UAV deviates too much from its specified route at any point in time. The CNF of the Boolean expression is |Location–Planned Location | $>\delta_{distance.}$ Here $\delta_{distance}$ stands for the maximum distance separation between the UAV's location and its planned location at the monitoring instant.

The 2nd attack behavior indicator (ABI 2) is that the trusted node is not forwarding packets to neighbor nodes whenever it should. The CNF is |Trusted Node NPR–Trusted Node NPS | $>\delta_{NPR\text{-}NPS.}$ Here NPR stands for the number of packets received per time unit, NPS stands for the number of packets sent per time unit, and $\delta_{NPR\text{-}NPS}$ stands for the maximum difference between this UAV's packet receiving rate and packet sending rate.

The 3rd attack behavior indicator (ABI 3) is that a monitor UAV provides bad recommendations toward a behaving trusted UAV (called bad-mouthing attacks), or/and good recommendations toward a misbehaving trusted UAV (called ballot-stuffing attacks). This is detected by comparing recommendations provided by multiple monitor UAVs and detecting discrepancies. The CNF is Trusted Node Audit ≠ Monitor Node Audit.

The 4th attack behavior indicator (ABI 4) is that a trusted node's embedded sensor reading differs from the monitor node's embedded sensor reading. The monitor node is in the neighborhood of the trusted node, measuring the same physical phenomenon. The CNF is|(Trusted Node Data – Monitor Node Data)/Monitor Node Data | $>\delta_{data}$ where δ_{data} is the maximum percentage difference between the trusted node sensor reading and the monitor node sensor reading.

The 5th attack behavior indicator (ABI 5) is that a trusted node is not reporting its sensor reading timely, therefore making the delayed sensing outcome practically useless. The CNF is |Time Received Trusted Node Data – Time Received Monitor Node Data | $>\delta_{time}$ where δ_{time} is the maximum time difference between the trusted node sensor reading time and the monitor node sensor reading time.

The 6th attack behavior indicator (ABI 6) is that a UAV does not accept authorized commands to fire the missile. The CNF is (Action≠ FIRE) $\wedge$ (Command=AUTHORIZED).

The 7th attack behavior indicator (ABI 7) is that a UAV could not ready its missile when it is at the specified target location. The CNF is (Missile≠ READY) $\wedge$ (Location = TARGET LOCATION).

The 8th attack behavior indicator (ABI 8) is that a UAV fires the missile in accurately. The CNF is (Action= FIRE) $\wedge$ (Outcome≠ SUCCESS).

The 9th attack behavior indicator (ABI 9) is that a UAV deploys landing gear when outside its home airbase. The CNF is (Gear = DEPLOYED)$\wedge$ (Location ≠HOME AIRBASE). This indicates that the UAV is likely to be captured by the enemy.

The 10th attack behavior indicator (ABI 10) is that a UAV sends sensing results to unauthorized parties (not to the specified ground station). This indicator catches attackers that intend to exfiltrate sending data. The CNF is Report Site ≠HOME GROUND STATION.

The 11th attack behavior indicator (ABI 11) is that a UAV transmits an exceptionally high number of packets sent (NPS) per second to consume energy. This UAV is also suspicious of performing DoS or jamming attacks when the NPS is too high [27]. The CNF is |(Trusted Node NPS – Monitor Node NPS)/Monitor Node NPS |$>\delta_{NPS}$ where δ_{NPS} is the maximum percentage difference between the trusted node's NPS and the monitor node's NPS.

The 12th attack behavior indicator (ABI 12) is that a loitering UAV uses more than the minimum thrust required to maintain altitude. This indicator catches attackers that intend to decrease a UAV's endurance by wasting its energy; these attackers attach to the UAV thrust module. The CNF is (Thrust >MINIMUM THRUST) $\wedge$ (Status = LOITER).The 13th attack behavior indicator (ABI 13) is that a UAV emits exceptionally high received signal strength intensity (RSSI) to consume energy. This UAV is also a suspect in spoofing GPS to seize control of another neighbor UAV [27]. The CNF is |Trusted Node RSSI–Monitor Node RSSI | $>\delta_{RSSI}$ where δ_{RSSI} is the maximum RSSI difference between the trusted node's RSSI and the monitor node's RSSI.

2) Formal Verification of Behavior Rules

We conduct automatic model checking and formal verification of the behavior rules generated (and the corresponding ABIs generated) by verifying that the behavior rules are correct and can cover all the threats (or satisfy the security requirements). We follow the description in Section IV to express the 13 ABIs (in Table 5) and the security requirements (in Table 2) in a 2-layer Fuzzy-HCAPN model, such that "any violation of the security requirements implies violations of the behavior rules" is expressed as a non-conclusive expression proven to be true in HCAPN.

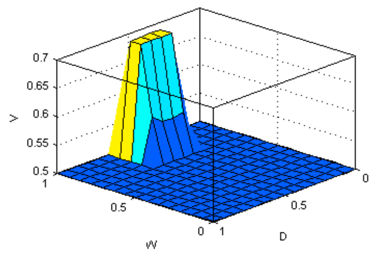


Fig. 8: Fuzzy rule observations for ABI1.

More specifically, using the 13 ABIs (in Table 5) and the security requirements (in Table 2) a monitor UAV initiates the verification by creating *V=Fuzzy(B, D, W)* based on the variables defined in each behavior rule by using conditions (1) ~ (9). We take ABI 1 as our running example to illustrate the process. ABI1 has "location" as the main variable which also appears in ABI 7 and ABI 9 for both the user and the expert. Now, since $R_{UEk} = 1$based on (2) ~ (3) as both the incoming UAV and the monitor UAV share the same set of behavior rules in a mission, *D* will attain a value of 0.33 based on (4). With *W* being defined as a function of available residual, *W* is varied between 0 and 1, for which the fuzzy rule observations for ABI 1 can be seen in Fig. 8.

This figure helps to understand the impact of each behavior rule and tells whether each behavior rule is verifiable or not based on the values set for *V*. Furthermore, *V=Fuzzy (B, D, W)* can be tracked for individual behavior rules considering the occurrences of each variable in it as well as a collective model by considering variables in all the available behavior rules. However, in such a case, micro-management is not possible and additional overheads are accumulated for massive computations.

Note that the model will be operated in localized, non-localized, or non-available forms depending on the presence of an expert's information. Once fixed, a HCAPN model based on (10) ~ (18), as shown in Fig. 9, can be built using all 13 behavior rules available from the expert. The upper layer is formed by using the variables from the individual behavior rules. To keep it simple, we only show the variables of ABI 1. For ABI 1, there are two variables, namely, location and planned locations. Thus, two places, V1 and V2, are shown in the upper layer. The relations between the places in the lower layer are governed by the principle of having common variables with each other. The passes are marked by checking the dependency of behavior rules. The transitions are fired if the actual data for the variables are available.

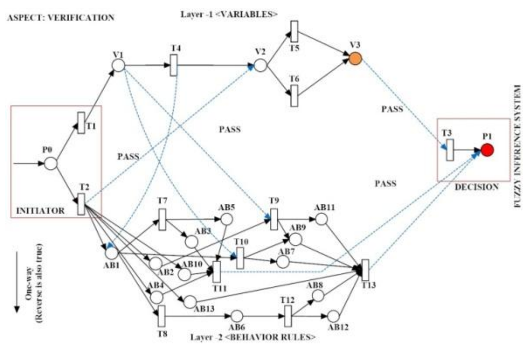
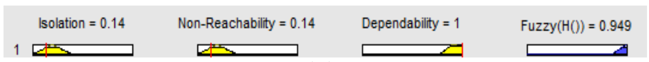
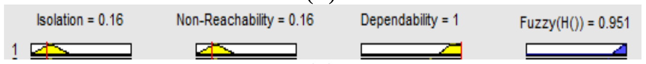


Fig. 9: An illustration of the HCAPN model using 13 behavior rules and ABI 1 as the verifiable content.

Next, by using fuzzy observations (Fig. 6) and (19), a decision is taken to determine if ABI 1 with a defined set of variables is correct or not. To understand this, put the fuzzy output *V* for ABI 1 to *Fuzzy(H(.))* along with the range of isolation, dependability, and non-reachability (as discussed in (13)~(18)) of ABI 1 by using Fig 9. For ABI 1, isolation and non-reachability have the same value (i.e., $R_B = \left(1 - \frac{1}{\left(\left(\frac{|A2'|}{|A2|}\right)^2 - 1\right)}\right)$) as the second term of non-usability passes is 1 since the no-extra pass is used and no-pass remains unused for ABI 1. Now, by putting values of the total number of places, i.e., 13, and the total number of unused places, i.e., 4 based on (13) and (14), a value of 2 for isolation and non-reachability is obtained. Upon normalizing on the maximum values, i.e., 7 and 8 attained by using the maximum value of unused places=13-1=12 and the minimum value, i.e., 1, isolation and non-reachability are given the values of 0.16 or 0.14. For dependability, (16) and (17) yield a dependability value of 1 since no additional variables are included and the average number of variables in ABI 1 for the user's model and the expert's model is the same, we set $\overline{\vartheta}$ at 2 as two variables are required for ABI 1 (thresholds can be ignored).



(A)



(B)

Fig. 10: Observed values of *G* alongside *Fuzzy(H())* for ABI 1 at two different values of isolation and non-reachability (A: isolation and non-reachability=0.14; and B: isolation and non-reachability=0.16).

Now, placing these values in Fig. 6, it can be determined that ABI 1 is greater than the range defined for correctness. As illustrated in Fig. 10, the observed output from *Fuzzy(H())* is 0.949~0.951 based on fuzzy inference rules. By (19) we obtain *G*= 1, meaning that ABI 1and its two variables are verifiably correct.

3) State Machine Generation and Feedback-based DNF for UAV Misbehavior Detection

For the UAV state machine, there are 13 Boolean variables (each taking the value of either 1 or 0) in the state representation, resulting in the total number of states being 2^{13}= 8192, out of which only one is a safe state (when all 13 Boolean variables are false or take the value of 0) and all other 8191 states are unsafe states. This acts as the expert's model in the identification of UAV misbehavior. Here, it is to be noted that there are many variables in these 13 attack behavior indicator expressions, including Location, Planned Location, Trusted Node NPR, Trusted Node NPS, Trusted Node Audit, Monitor Node Audit, Trusted Node Data, Monitor Data, Time Received Trusted Node Data, Time Received Monitor Node Data, Action, Command, Missile, Outcome, Gear, Report Site, Monitor Node NPS, Thrust, Status, Trusted Node RSSI, and Monitor Node RSSI. However, these variables are internal variables maintained by a monitor UAV who updates these internal variable values at monitoring intervals to determine the true/false (or 1/0) of the 13 Boolean variables for a trusted UAV that is being monitored on.

The state machine generated is equivalent to connecting the 13ABIs in Table 5 in disjunctive normal form (DNF). In other words, it is a Boolean expression of the 13 ABIs connected in DNF, representing the expert's opinion on UAV misbehavior. To account for output variations which may be caused by rapid environmental and operational changes and UAV dynamic adjustments in response to environmental changes, we apply feedback-based DNF as discussed in Section IV.A.3, such that:

$$F\text{-}DNF = DNF \rightarrow \varepsilon_X^F = feedback(\text{Misbehavior Range ("ABI: X")}),$$

where DNF is observable as: (|Location–Planned Location | $>\delta_{\text{distance}}$) $\vee$(|Trusted Node NPR–Trusted Node NPS | $>\delta_{\text{NPR-NPS}}$) $\vee$ (Trusted Node Audit $\neq$ Monitor Audit) $\vee$(|(Trusted Node Data - Monitor Node Data)/Monitor Node Data| $>\delta_{\text{data}}$) $\vee$ (|Time Received Trusted Node Data –Time Received Monitor Node Data| $>\delta_{\text{time}}$) $\vee$ ((Action$\neq$ FIRE) $\wedge$ (Command = AUTHORIZED)) $\vee$ ((Missile$\neq$ READY) $\wedge$ (Location = TARGET LOCATION)) $\vee$((Action= FIRE) $\wedge$ (Outcome$\neq$ SUCCESS)) $\vee$ ((Gear = DEPLOYED) $\wedge$ (Location $\neq$ HOME AIRBASE)) $\vee$ (Report Site $\neq$ HOME GROUND STATION) $\vee$(|(Trusted Node NPS – Monitor Node NPS)/Monitor Node NPS | $>\delta_{\text{NPS}}$) $\vee$ ((Thrust >MINIMUM THRUST) $\wedge$ (Status = LOITER)) $\vee$ (|Trusted Node RSSI– Monitor Node RSSI | $>\delta_{\text{RSSI}}$), and ε_X^F associates the correction values for the *X*th ABI, which helps the system detect misbehavior that accounts for output variations due to environmental and operational changes and adjustments in the activity of a UAV during its mission.

The system performs misbehavior detection as discussed in Section IV.B by following conditions in (20) ~ (41). The parameter values used in (20) ~ (41) are listed and explained in Table 6 in the next section.

Table 6: UAV-CPS Observations from the given 13 behavior rules (Tables4 – 5).

Parameter	Meaning	Value / derived using	Type
B	Behavior set	13 rules (Table 5)	Input
V	Verification Function	Fuzzy(B, D, W)	Output
D	Degree of dependence	(4)	Output
W	Statistical Weightage	0 ~ 1 (normalized)	Input
r_{UE}	Correlation coefficient between experts and user behavior rules	(1)	Output
ϑ	Total number of variables from B	32 (Table 5)	Input
α	Number of occurrences of variables in the user's behavior rules	2~5±5 (Table 5)	Input
β	Number of occurrences of variables in the expert's behavior rules	1~3 (Table 5)	Input
$D_{U,k}$	Dependence for kth behavior rule in the user's file	(2)	Output
$D_{E,k}$	Dependence of kth behavior rule in the expert's file	(3)	Output
$R_{UE,k}$	Ratio of dependence for a given behavior rule in user and expert file	(4)	Output
γ_U	Coefficient indicating importance of a behavior rule for the user	0.1~0.5	Input
γ_E	Coefficient indicating importance of a behavior rule for the Expert	0.1~0.5	Input
$\overline{\alpha_U}$	mean of occurrences for ϑ_U variables in all behavior rules	Table 5	Input
φ	Memory coefficient	(5)	Output
$\varphi_{UE,i}$	Relative memory coefficient	(6)	Output
ε	Residuals	0.1~1 [induced]	Input
B_d, W_d, D_d	Derived sets of B, W, and D for the evaluated behavior rules	B, with V=True (Fig. 4)	Output
N_U^P	Set of places in the upper layer of HCAPN	ϑ	Input
N_L^P	Set of places in the lower layer of HCAPN	B_d	Input
K	Number of sign changes between the slots	(12)	Output
$S_t^{(C)} and\ Q_t^{(C)}$	Functions tracking the change in signs for the recorded variables and behavior rules, respectively	(12)	Output
$S_I^{<Layer>}$	Layer-wise isolation in HCAPN	(13)	Output
$S_I^{<collective>}$	Collective isolation in HCAPN	(14)	Output
R_B	Non-reachability in HCAPN	(15)	Output
E_D^P	Dependability in HCAPN	(16)	Output
$\lvert B_{d,x} \rvert$	Number of behavior rules with new variables	Table 5	Input
$\bar{\vartheta}$	Average number of variables in each behavior rule	Table 5	Input
ϑ_x	the number of new variables	1~5 [induced]	Input
$\overline{\vartheta_x}$	Average number of variables in the new behavior rules	5 [induced]	Input
Fuzzy (H (.))	Value of G	Fig. 6	Output
ε_X^F	Accumulated feedback variable	(20)	Output
$Q(B_d, U, Y)$	Bipartite graph between the behavior rules	(13, count(ϑ), count(ε_X^F))	Input
$R_{E,M}^{(C)}$	Barycentric coordinate for center of mass for expert's observations	(21)	Output

$R_{R,M}^{(C)}$	Barycentric coordinate for center of mass for a given device	(21)	Output
ΔR_X^F	Difference in the Barycentric Coordinates	(24)	Output
M_b	Misbehavior	(25)	Output
ΔTH	Observational thresholds for all the behavior rules.	10 % ~ 50 %	Input
$L_{X,R,M}^{(C)}$, $L_{X,E,M}^{(C)}$	Local Barycentric coordinates	(29) and (30)	Output
$\Delta L_A^{(C)}$	Adjustments in the local Barycentric Coordinates	(31)	Output
t_1, t_2… t_n	Timestamps	100 s each	Input
θ	Compliance degree	(32)	Output
ρ	Compliance constant	$\frac{1}{\lvert B_d \rvert}\sum_{i=1}^{\lvert B_d \rvert}\gamma_i$, γ_U or γ_E	Input
θ_p	Predicted compliance degree	(35)	Output
ω_g	Weibull PDF	(37)	Output
$\xi(\omega_g)$	System's reliability	(38)	Output
$T(\theta)$	Instance evaluating function	= T Step interval 10	Input
$\omega_g^{(O)}$	Observed Weibull PDF	(39)	Output
$\xi(\omega_g^{(O)})$	Observed System's reliability	(40)	Output
$\pm\psi$	Limiting constant	(41) [based on adjustments]	Output
T	Total time with step 10	1000s	Input

B. EVALUATION

The proposed approach is evaluated against a simulated UAV-CPS by MATLAB^{TM} operating under randomized scenarios created by different sets of model parameter values as listed in Table 6.The system comprises good and bad UAVs in accordance with the true input from an expert. An expert has a true account of all the 13 behavior rules in the behavior rule set B listed in Table 4 (correspondingly the 13 ABIs in Table 5) and each incoming UAV is evaluated against it. The details of the model parameters and their values used in evaluation are listed in Table 6. The values in this table are obtained by following the formulations of the 13 ABIs in Table 5 as discussed in Section IV.A. For example, ABI 1 is marked with "location" and "planned location," meaning that in this formulation, ABI 1 operates with these two variables. Furthermore, the variable "location" appears three times in ABI 1, ABI 7 and ABI 9, so its β value is 3. On the other hand, the variable "planned location" appears only once in ABI 1, so its β value is 1. Similarly, other variables in other behavior rules are obtained to generate the experiment setup in Table 6.

A distinct set of parameter values as listed in Table 6 defines a distinct scenario to the model and the system is tracked for misbehavior amongst UAVs. To accurately trace false positives, false negatives, and true positives, a single UAV is selected in the simulation with a run time of 1000s with 10 evaluation instances of 100s each. It means that the UAV evaluation is conducted afresh in every 100s. In each evaluation, a fresh scenario is being tested with the value of αbeing varied by ±5 to check if the system can track its activity and mark the suspicious activity as misbehavior. Furthermore, in each evaluation, other parameters also change their values (can be observed in graphs) to test the sensitivity of the performance with respect to these changes. The variation in these values helps us simulate certain bad behavior UAVs (with incorrect behavior rules), which allows accounting for false negatives, false positives and true positives using standard formulations [18]. The range of compliance degree is modeled around compliance constantρ, which attains its values from γ as presented in (32). Note that this paper does not consider the UAVs' communication aspect and the issues related to latency, overheads and real-scenario noise are to be addressed in future work.

The initial observations help to understand the dependence of the behavior rules (D), which is verified by deriving the correlation coefficient in (1). D considers the importance of behavior rules as the key in deciding whether it is relevant to decide the misbehavior of a device based on a particular behavior rule. This also helps to track the behavior profiling of the entire model as well as its applicability to a particular scenario. To understand the impact of D on the behavior rules in Table 4, Fig.11shows that the variation in the importance of a variable in the observed rule causes a variation in the dependence value and it increases with it. Predominately, this graph shows that it is the expert's profiling which matters and the user's derived rules (whether generated manually or automatically) are affected accordingly. This value of D is crucial in forming the fuzzy set which also helps in the formal verification of the behavior rules.

Next, the evaluations are conducted to understand the impact of weight on the derived system as shown in Figs. 12, 13 and 14. These results help to understand the impact of W through Wannier derivations in (5)-(9). These results are affected by the residuals and follow different trends on the identification of errors with a given set of rules. These results suggest that if the residuals are localized and the system is aware of them, the possibilities of identification of errors (misbehavior) increases with an increase in the residual. Furthermore, with more localized residuals, the system is able to take a far more appropriate decision in marking a particular rule form is behavior detection (Fig. 13). In case the residuals are non-localized, the system's performance degrades as there are high possibilities of the system being unable to mark certain rules form is behavior detection (Fig. 14). However, this scenario can return in favor if the residuals follow a particular trend, as shown in Fig. 14, because the system can perform better prediction as expressed in the statistical modeling in (35) and (36).

All these observations impact the proposed center of mass based misbehavior detection as shown in Fig. 15. This figure shows the comparison of $R_{R,M}^{(C)}$vs. probability derived over behavior rules with a variation in the types of residuals through Wannier function for the accurate identification of misbehavior based on the given behavior rules. These results comply with the residuals' state and follow the trend as per the system observation in localization. Furthermore, this result can be used to check the deviation of the system from the present state and generate feedback. Such a solution can be helpful in making devices learn about the accurate state of operation when deployed in a particular scenario.

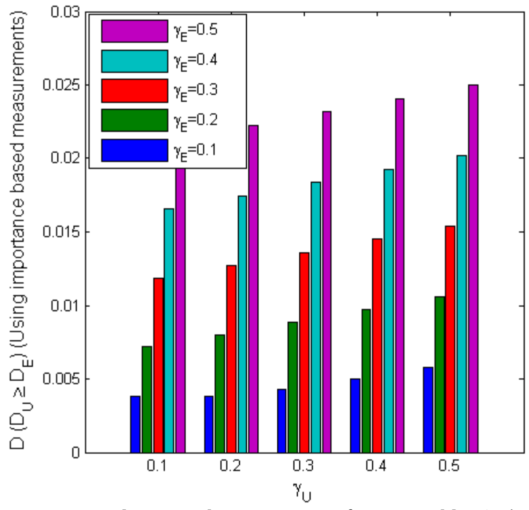


Fig. 11: Dependence vs. the importance of user variables (γ_U) with a variation in the importance of experts variable (γ_E) for the given behavior rules in Tables 4 and 5.

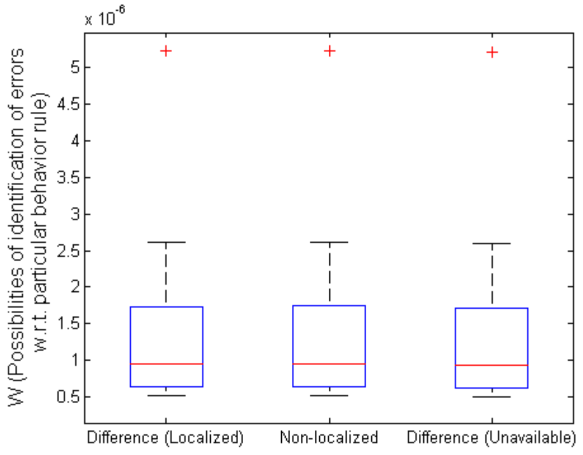


Fig. 12: Possibilities of identification of errors with a difference in residual localization through Wannier functions.

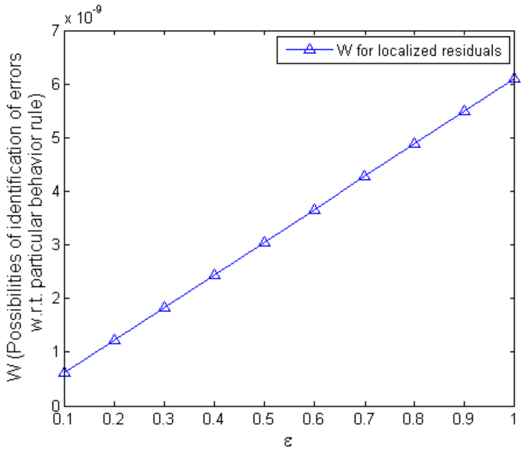


Fig. 13: Possibilities of identification of errors with a difference in localized residuals through Wannier functions.

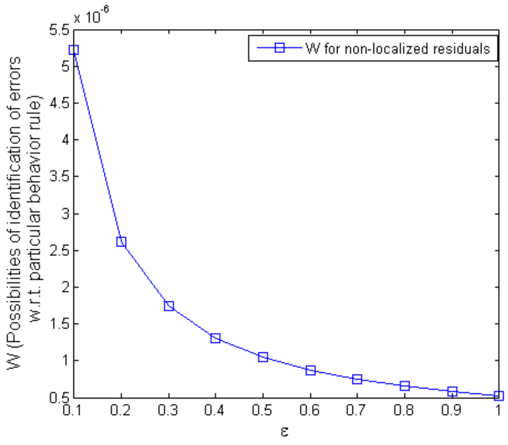


Fig. 14: Possibilities of identification of errors with difference in non-localized residuals through Wannier functions.

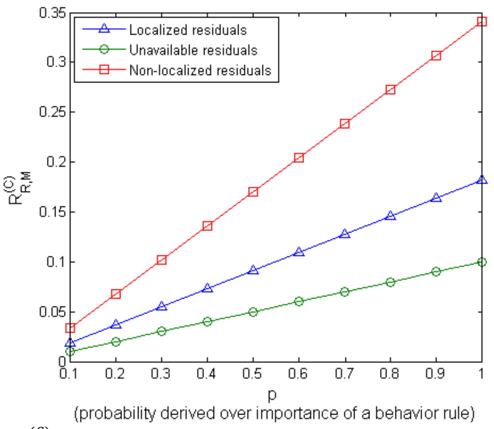


Fig. 15:$R_{R,M}^{(C)}$ vs probability derived over behavior rules with a variation in the types of residuals through Wannier function for the accurate identification of misbehavior based on the given behavior rules.

The unavailable residuals are generally operated for the systems with no feedback from the experts and show a less deviation for Barycentric values as shown in Fig. 15. However, for localized residuals, accurate readings are available and the deviation is in control showing the true nature of the system. For the non-localized scenario, the system becomes too pessimist and it takes a large value and marks major of the rules as a possible non-follower with the expert's policies. Thus, it becomes extremely important to evaluate the system for all the three conditions to accurately identify misbehavior amongst the devices. Further, these results suggest that with an increase in the value of probabilistic weights, the systems deviation increases with it; and at a high value, large feedback is generated which marks the system as misbehavior under all circumstances. Thus, control overweight is also additionally required to accurately implant the proposed model. Numerically, the misidentification of residuals can lead up to 75% error in the readings, whereas accurate generations can immediately lower the error readings by 50%, as shown in Fig.15.

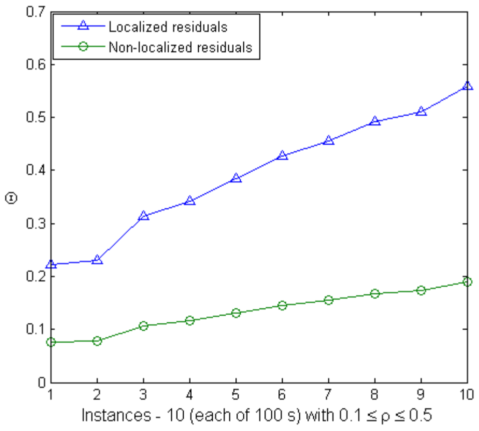


Fig. 16: Compliance degree data variation for the observed instances for given localized and non-localized residuals with a variation of compliance constant observed for the derived importance of user variables and the expert variables.

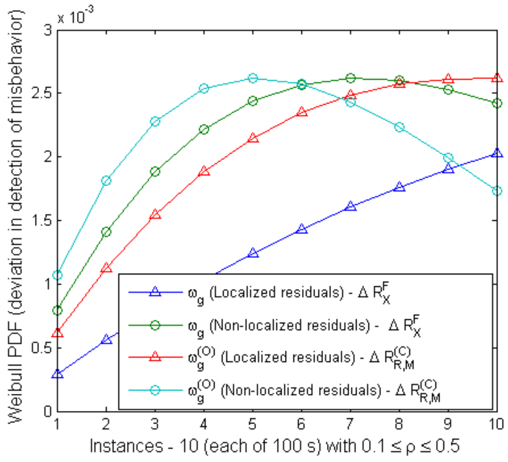


Fig. 17: Misbehavior detection distribution vs. variation in the number of instances for the derived compliance degree data for localized, non-localized and unavailable.

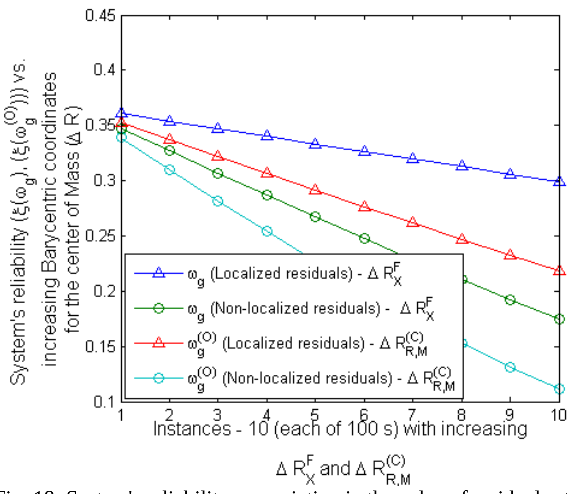


Fig. 18: System's reliability vs. variation in the value of residuals at the observed Barycentric coordinates for the center of mass for the misbehavior tracking in the defined UAV model.

The compliance degree data collection is a key step behind the evaluation of the system towards accuracy and reliability. This model is generated to overcome the issues of the linear approach used by most existing work, which limits their applicability to complex systems. The compliance degree data (θ) varies as per the instance and better readings for each instance generate better compliance data as shown in Fig. 16. These results suggest that the scenario with better feedback and better residuals provide sufficiently detailed data, which helps to provide better evaluations for the given system. The localized scenario offers 11.2% better compliance value than the non-localized scenario, thus offering a better understanding of the system and more accurately tracking the behavior. Irrespective of these, the proposed BRIoT model can be applied to any scenario with adjustments to additional metrics, such as compliance constants and importance value. These observations can be seen in Figs. 17and 18, which complement each other and show that in the case of non-localized residuals, the system may show a variable distribution over the compliance data and then gradually decreases, thus lowering the performance of the entire model. In contrast to this, the localized model operates with a much accurate reading and keeps on increasing with the detailed availability of the compliance data. Additionally, the exact observation can be marked as reliability in tracking the misbehavior, which is shown in Fig.18. Accordingly, the localized and completely available system shows improved performance compared with a system with partial observations. This variation lies between 8.10 % and 43.75 %, which decreases as the deviation of the system from the given (expert's value) Barycentric coordinate increases. The reliability of the model can be controlled with better compliance degree and accurate identification of compliance constant which is affected by the importance value of each variable in the observed as well as available behavior rule. All these evaluations help to understand the range and limit up to which the proposed BRIoT can be successfully applied to check whether the generated behavior rules are correct or not. Furthermore, these results demonstrate that the proposed BRIoT can accurately mark misbehavior amongst embedded IoT devices based on the derived rules.

C.COMPARATIVE ANALYSIS

The proposed BRIoT approach is compared with BRUIDS [18], which is a well-versed and a competitive solution in the detection of UAV misbehavior. The statistical model in BRUIDS considers the compliance degree as a random variable following Beta distribution such that a value of zero indicates zero compliance and a value of one indicates total compliance. It collects a device's compliance degree periodically based on the proportion of time the device stays in a safe state, but it does not track which state the device is in over time. Once it parameterizes the Beta distribution using the compliance degree data collected, it sets a minimum compliance threshold below which the node is identified as malicious; otherwise the node is considered good. BRUIDS could fail when the number of states is large. Also, there is no support for feedback to allow for output variations which may cause misdetection of misbehavior. Further, BRUIDS is only theoretically verified with pre-generated state data. We compare the performance of BRIoT against BRUIDS using the exact statistical model used by BRUIDS. The major difference between the two is the model of statistical evaluation. The proposed approach with the use of compliance constant (derived over the importance of each variable (γ)) offers better observations through Weibull-evaluations. Additionally, the details of variables in the given rules are kept the same for both the models for a fair comparison.

Table 7: Performance comparison of BRIoT with BRUIDS for misbehavior detection of UAVs.

Parameters	BRUIDS	BRIoT (Localized residuals)	BRIoT (Non-localized residuals)
False Negative Rate	AVG: 0.229	AVG: 0.137	AVG: 0.022
False Positive Rate	AVG: 0.059	AVG: 0.045	AVG: 0.040
True Positive Rate	AVG: 0.771	AVG: 0.863	AVG: 0.978
Compliance Degree (In-depth)	RANGE: 0.1~0.9	RANGE: 0.008~0.080	RANGE: 0.008~0.063

Table 7 compares BRIoT (with localized residuals or non-localized residuals) with BRUIDS in false negative rate, false positive rate, true position rate, and range of compliance degree. The core observation of BRIoT is in its high accuracy in dealing with compliance degree data, driven by the compliance coefficients and the importance of the variables in the behavior rules. In contrast, BRUIDS focuses on collecting instances of compliance degree, driven by a time model without tracking output values and may cause high false positives and high false negatives. For BRUIDS, the compliance data is observed for 10instances ranging between 0.1 and 0.9, whereas for BRIoT, the range is much narrower and varies slightly for localized residual-based and non-localized residual-based evaluations. The results show that our proposed BRIoT model with localized residuals (with non-localized residuals) improves the false negative rate or true positive rate by an average of 9.2% (20.7% respectively) and false positive rate by an average of 1.4% (1.9% respectively) in comparison with BRUIDS over 10 instances.

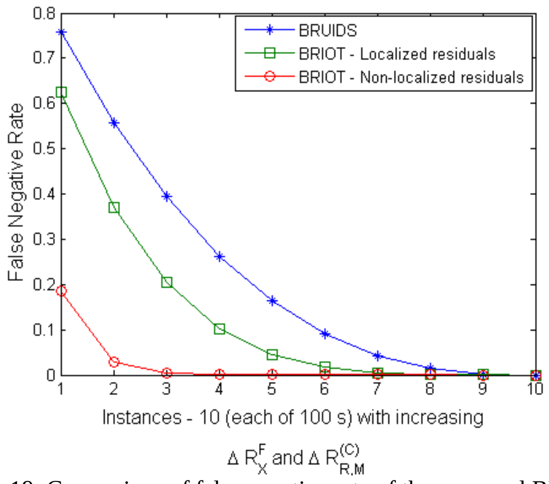


Fig. 19: Comparison of false negative rate of the proposed BRIoT with the existing BRUIDS at a variation on the instances and the available values for the center of masses for the given model.

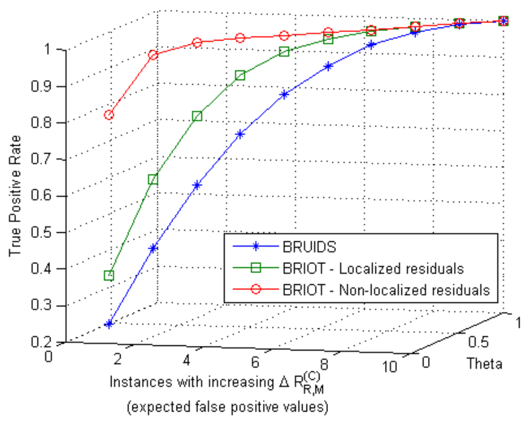


Fig. 20: Comparison of true positive rate of the proposed BRIoT with the existing BRUIDS at a variation on the instances and the available values for the center of masses for the given model along with the compliance degree data.

Figs. 19 and 20 compare our proposed BRIoT model with the existing BRUIDS model for the UAV-CPS in false negative probability and the true positive probability, respectively. These results demonstrate the effectiveness of our proposed BRIoT model in verifying the correctness of the behavior rules and achieving better convergence than BRUIDS for identifying true misbehavior amongst UAV devices in the UAV-CPS. As more instances are observed, both BRIoT and BRUIDS perform comparably. This refers to the situation when ΔR_X^F(see (24)) becomes too large showing a major difference from the values of Barycentric coordinates as per the center of mass defined in (21)-(24). It means that at a huge gap between the observational and attained feedback, the performance of the system degrades yielding almost similar false negatives for all the approaches. The initial verification of behavior rules helps control the feedback. in Fig. 19 and Fig. 20, the performance of both the models is comparable only after the 9th iteration where feedback has a minimum role to play. Considering this, BRIoT offers better convergence for misbehavior detection of UAVs than BRUIDS. Hence, BRIoT is especially applicable to devices whose initial states of operations are unknown. It is observed that the scale parameters show a variation of 41.31% and 83.63% with the BRUIDS based on localized and non-localized residuals, respectively. Although the scale parameters show 72.12% variation for localized and non-localized residuals, the compliance degree operates the entire results and better convergence of localized approach offers accurate misbehavior detection of UAVs. The non-localized residual approach adjusts based on the given compliance values, thus generating lower false negatives and false positives.

We attribute the superiority of BRIoT over BRUIDS for its ability to account for runtime output variations using the feedback mechanism and its effective misbehavior detection to avoid false alarms through a Barycentric-coordinated based center of mass calculation method. Another reason of BRIoT performing better than BRUIDS is that BRIoT collects compliance data during state transitions so it can track the current state a target UAV is in at any time, while BRUIDS only collects the compliance degree (a value between 0 and 1 representing whether or not a target UAV complies with the behavior rules) at time instants without the ability to track state transition history. This state-tracking ability helps BRIoT achieve higher false negative rate and false positive rate especially when evidence is not easily observable until many instances have been seen.

VI. CONCLUSION

In this paper, a behavior rule specification-based misbehavior detection method called BRIoT has been designed and built that can be generally applicable IoT-embedded CPSs. BRIoT is capable of formally verifying the correctness of behavior rules for any embedded IoT device and collecting/analyzing compliance data for misbehavior detection. BRIoT is especially applicable to mission-critical CPSs with specified security requirements regardless if the attacks are known or unknown because it detects an IoT device's misbehavior manifested as a result of attacks.

We have developed BRIoT as a tool allowing a user (or a domain expert) to specify the operational profile of an embedded IoT device as input. The tool can then automatically generate a set of security requirements and a set of behavior rules, verify the correctness of the behavior rules generated, and convert the behavior rules into a state machine for runtime misbehavior detection. The overall operational cost is very low and it can be operated in both on-devices as well as off-device mode in three possible situations, i.e., localized residuals, non-localized residuals, and unavailable residuals. Through a comparative analysis, we demonstrated that BRIoT outperforms BRUIDS, a contemporary specification-based misbehavior detection method, for misbehavior detection of UAVs in a UAV-CPS in reliability, false-positives, false-negatives, and true positives.

In the future, we plan to further analyze the tradeoff between effectiveness (measured by false negative rate, false positive rate, and true position rate) vs. efficiency (measured by memory, run time, communication, and computation overhead) for BRIoT to apply to practical IoT-embedded CPSs.

REFERENCES

[1] R. Berthier and W.H. Sanders, "Specification-based Intrusion Detection for Advanced Metering Infrastructures," *17th IEEE Pacific Rim Int.Symp. Dependable Computing*, pp. 184-193, 2011.
[2] A. Bezemskij, G. Loukas, R.J. Anthony, and D. Gan, "Behaviour-based anomaly detection of cyber-physical attacks on a robotic vehicle," *IEEE Symposium on Cyberspace and Security*, pp. 1-8, Dec. 2016.
[3] I.R. Chen, F. Bao, and J. Guo, "Trust-based Service Management for Social Internet of Things Systems," *IEEE Transactions on Dependable and Secure Computing*, vol. 13, no. 6, Nov-Dec 2016, pp. 684-696.
[4] A. DaSilva et al., "Decentralized intrusion detection in wireless sensor networks," *1st ACM inter. workshop on quality of service & security in wireless and mobile networks*, pp. 16–23, 2005.
[5] S. Hanna, "A path to securing billions of insecure devices,"https://www.trustedcomputinggroup.org/wp-content/uploads/Trusted-Computing-for-IoT-ESC-2015_final.pdf, 2015.
[6] J. Hong, C.C. Liu, and M. Govindarasu, "Integrated Anomaly Detection for Cyber Security of the Substations," *IEEE Trans. Smart Grid*, vol. 5, no. 4, 2014, pp. 1643-1653.
[7] S. Huda, et al., "Defending unknown attacks on cyber-physical systems by semi-supervised approach and available unlabeled data," *Information Sciences*, vol. 379, 2017, pp. 211-228.
[8] K. Ioannis, T. Dimitriou, and F. Freiling, "Towards intrusion detection in wireless sensor networks," *13th European Wireless Conference*, 2007.
[9] P. Jokar, H. Nicanfar, and V.C.M. Leung, "Specification-based Intrusion Detection for Home Area Networks in Smart Grids," *IEEE Int. Conf. on Smart Grid Communications*, 2011.
[10] A.M. Kosek, "Contextual anomaly detection for cyber-physical security in smart grids based on an artificial neural network model," *IEEE Workshop on Cyber- Physical Security and Resilience in Smart Grids*, 2016.

[11] C. Kwon, S. Yantek, and I. Hwang, “Real-Time Safety Assessment of Unmanned Aircraft Systems Against Stealthy Cyber Attacks,” *Journal of Aerospace Information Systems*, vol. 13, no. 1, 2016, pp. 27-46.

[12] R. Mitchell, and I.R. Chen, “A Survey of Intrusion Detection Techniques in Cyber Physical Systems,” *ACM Computing Survey*, vol. 46, no. 4, article 55, 2014.

[13] R. Mitchell and I.R. Chen, “Modeling and Analysis of Attacks and Counter Defense Mechanisms for Cyber Physical Systems,” *IEEE Transactions on Reliability*, vol. 65, no. 1, March 2016, pp. 350-358.

[14] S. Ntalampiras, “Automatic identification of integrity attacks in cyber-physical systems,” *Expert Systems with Applications*, vol. 58, 2016, pp. 164-173.

[15] Y. Zhang, L. Wang, W. Sun, R. Green, and M. Alam, “Distributed intrusion detection system in a multi-layer network architecture of smart grids,” *IEEE Trans. Smart Grid*, vol. 2, no. 4, pp. 796–808, Dec. 2011.

[16] J. Musa, “Operational profiles in software reliability engineering,” *IEEE Software*, pp. 14–32, Mar. 1993.

[17] R. Mitchell and I.R. Chen, “Behavior Rule Specification-based Intrusion Detection for Safety Critical Medical Cyber Physical Systems,” *IEEE Transactions on Dependable and Secure Computing*, vol. 12, no. 1, 2015, pp. 16-30.

[18] R. Mitchell and I.R. Chen, "Adaptive Intrusion Detection of Malicious Unmanned Air Vehicles Using Behavior Rule Specifications,” *IEEE Transactions on Systems, Man and Cybernetics*, vol. 44, no. 5, 2014, pp. 593-604.

[19] I.R. Chen, J. Guo, and F. Bao,"Trust Management for SOA-based IoT and Its Application to Service Composition," *IEEE Transactions on Services Computing,* vol. 9, no. 3, 2016, pp. 482-495.

[20] P. Xun et al., "Command Disaggregation Attack and Mitigation in Industrial Internet of Things," *Sensors*, vol. 17, 2017, article 2408.

[21] T. Song, et al., "Formal Reasoning about a Specification-based Intrusion Detection for Dynamic Auto-configuration Protocols in Ad Hoc Networks," *Formal Aspects in Security and Trust,* pp. 16-33, 2006.

[22] C.-H. Tsang and S. Kwong. Multi-agent intrusion detection system in industrial network using ant colony clustering approach and unsupervised feature extraction. In *International Conference on Industrial Technology*, pages 51–56, Hong Kong, December 2005.

[23] D.-T. Ho and S. Shimamoto. Highly reliable communication protocol for WSN-UAV system employing TDMA and PFS scheme. In *Global Communications Conference Workshops*, pages 1320–1324, Houston, TX, USA, December 2011.

[24] C. E. Palazzi, C. Roseti, M. Luglio, M. Gerla, M. Y. Sanadidi, and J. Stepanek. Enhancing Transport Layer Capability in HAPS-Satellite Integrated Architecture. *Wireless Personal Communications*, 32:339–356, 2005.

[25] R. Mitchell and I. R. Chen. Effect of Intrusion Detection and Response on Reliability of Cyber Physical Systems. *IEEE Transactions on Reliability*, 62(1):199–210, March 2013.

[26] M. Aldebert, M. Ivaldi and C. Roucolle,"Telecommunications Demand and Pricing Structure: An Econometric Analysis,"*Telecommunication Systems*, 25:89–115, 2004.

[27] H. Sedjelmaci, S.M. Senouci, and N. Ansari, "A Hierarchical Detection and Response System to Enhance Security against Lethal Cyber-Attacks in UAV Networks," *IEEE Transactions on Systems, Man, and Cybernetics: Systems,* 2017.

[28] B.B. Zarpelao, R.S. Miani, C.T. Kawakani, S.C. de Alvarenga, "A Survey of Intrusion Detection in Internet of Things," *Journal of Network and Computer Architecture,* vol. 84, 2017, pp. 25-37.

[29] A. Saeed, A. Ahmadinia, A. Javed, and H. Larikani, "Intelligent Intrusion Detection in Low-Power IoTs," *ACM Trans. Internet Technology,* vol. 16, no. 4, article 27, 2016.

[30] M.T. Khan, D. Serpanos, and H. Shrobe, "ARMET: Behavior-based Secure and Resilient Industrial Control Systems," *Proceedings of The IEEE*, 2017.

[31] D. He, S. Chan, and M. Guizani, "Drone-assisted Public Safety Networks: The Security Aspect," *IEEE Communications Magazine*, 2017, pp. 2-8.

[32] M. Kaufmann and J.S. Moore, A Computational Logic for Applicative Common Lisp, http://www.cs.utexas.edu/users/moore/acl2/, 2017.

[33] V. Sharma, G. Choudhary, Y. Ko, I. You, "Behavior and Vulnerability Assessment of Drones-Enabled Industrial Internet of Things (IIoT),"*IEEE Access*, vol. 6, 2018, pp. 43368-83.

[34] D.P. Francis, A.J. Coats, and D.G. Gibson, "How high can a correlation coefficient be? Effects of limited reproducibility of common cardiological measures,"*International Journal of Cardiology*, vol. 69, no. 2, 1999, pp. 185-9.

[35] J. Creedy and V. Martin,*Chaos and non-linear models in economics*, Edward Elgar Publishing, 1994.

[36] A. Bjorck, *Numerical methods for least squares problems*, Siam; ISBN 0-89871-360-9, 1996.

[37] N. Marzari, I. Souza, and D.Vanderbilt,"*An introduction to maximally-localized Wannier functions*", Psi-K newsletter. 2003, pp. 57:129.

[38] B.J. Fino and V.R. Algazi, "Unified matrix treatment of the fast Walsh-Hadamard transform,"*IEEE Transactions on Computers*, vol. 1, no. 11, 1976, pp. 1142-6.

[39] R.L. Burden and J.D. Faires,*Numerical analysis*, Cengage Learning, 2010.

[40] N. Zhang, K. Sun, W. Lou, and Y.T. Hou, “CaSE: Cache-Assisted Secure Execution on ARM Processors,” *IEEE Symposium on Security and Privacy*, 2016.

[41] R. Jiang and D.N.P. Murthy, "A study of Weibull shape parameter: Properties and significance," Reliability Engineering & System Safety, vol. 96, no. 12, pp.1619-1626.

[42] V. Novák, I. Perfilieva, and J. Mockor, "Mathematical principles of fuzzy logic," Vol. 517. Springer Science & Business Media, 2012.

VISHAL SHARMA (S'13, M'17) received the Ph.D. and B.Tech. degrees in computer science and engineering from Thapar University (2016) and Punjab Technical University (2012), respectively. He worked at Thapar University as a Lecturer from Apr'16-Oct'16. From Nov. 2016 to Sept. 2017, he was a joint post-doctoral researcher in MobiSec Lab. at Department of Information Security Engineering, Soonchunhyang University, and Soongsil University, Republic of Korea. Dr. Sharma is now a Research Assistant Professor in the Department of Information Security Engineering, Soonchunhyang University, The Republic of Korea. Dr. Sharma received three best paper awards from the IEEE International Conference on Communication, Management and Information Technology (ICCMIT), Warsaw, Poland in April 2017; from CISC-S'17 South Korea in June 2017; and from IoTaas Taiwan in September 2017. He is the member of IEEE, a professional member of ACM and past Chair for ACM Student Chapter-TIET Patiala. He has authored/coauthored more than 90 journal/conference articles and book chapters, and co-edited two books with Springer. He serves as the program committee member for the Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA). He was the track chair of MobiSec'16 and AIMS-FSS'16, and PC member and reviewer of MIST'16 and MIST'17, respectively. He has been the TPC member of ETIC- 2019, WiMO-2019, ITNAC-IEEE TCBD'17, ICCMIT'18, CoCoNet'18 and ITNAC-IEEE TCBD'18. Also, he serves as a reviewer for various IEEE Transactions and other journals. His areas of research and interests are 5G networks, UAVs, estimation theory, and artificial intelligence.

ILSUN YOU (SM'13) received the M.S. and Ph.D. degrees in computer science from Dankook University, Seoul, Korea, in 1997 and 2002, respectively. He received the second Ph.D. degree from Kyushu University, Japan, in 2012. From 1997 to 2004, he was at the THIN multimedia Inc., Internet Security Co., Ltd. and Hanjo Engineering Co., Ltd. as a research engineer. Now, he is an associate professor at the Department of Information Security Engineering, Soonchunhyang University. He has served or is currently serving as the main organizer of international conferences and workshops such as MobiWorld, MIST, SeCIHD, AsiaARES, and so forth. Dr. You is the EiC of Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA). He is in the Editorial Board for Information Sciences (INS), Journal of Network and Computer Applications (JNCA), International Journal of Ad Hoc and Ubiquitous Computing (IJAHUC), Computing and Informatics (CAI), Journal of High Speed Networks (JHSN), Intelligent Automation & Soft Computing (AutoSoft), and Security and Communication Networks (SCN). His main research interests include Internet security, authentication, access control, and formal security analysis. He is a Fellow of the IET and a senior member of the IEEE.

KANGBIN YIM received the B.S., M.S., and Ph.D. degrees from the Department of Electronics Engineering, Ajou University, Suwon, South Korea, in 1992, 1994, and 2001, respectively. He is currently a Professor with the Department of Information Security Engineering, Soonchunhyang University. His research interests include vulnerability assessment, code obfuscation, malware analysis, leakage prevention; secure platform architecture, and mobile security. Related to these topics, he has involved in over sixty research projects and published over a hundred research papers. Prof. Yim has served as an Executive Board Member of the Korea Institute of Information Security and Cryptology, Korean Society for Internet Information, and The Institute of Electronics Engineers of Korea. He also has served as a committee chair of the international conferences and workshops and the Guest Editor of the journals, such as JIT, MIS, JCPS, JISIS, and JoWUA.

ING-RAY CHEN (M'90) received the BS degree from the National Taiwan University, and the MS and PhD degrees in computer science from the University of Houston. He is a professor in the Department of Computer Science at Virginia Tech. His research interests are primarily in service and trust management as well as reliability and performance analysis of mobile systems and wireless networks, including Internet of Things, wireless sensor networks, service-oriented peer-to-peer networks, ad hoc networks, mobile social networks, mobile web services, mobile cloud services, and cyber physical systems. Dr. Chen currently serves as an editor for IEEE Transactions on Services Computing, IEEE Transactions on Network and Service Management, and The Computer Journal. He is a recipient of the IEEE Communications Society William R. Bennett Prize in the field of Communications Networking and a recipient of the U.S. Army Research Laboratory (ARL) Publication Award.

JIN-HEE CHO (SM'14) is currently an associate professor in the Department of Computer Science at Virginia Tech since Aug. 2018. Prior to joining the Virginia Tech, she worked as a computer scientist at the U.S. Army Research Laboratory (USARL), Adelphi, Maryland, since 2009. Dr. Cho has published over 100 peer-reviewed technical papers in leading journals and conferences in the areas of trust management, cybersecurity, metrics and measurements, network performance analysis, resource allocation, agent-based modeling, uncertainty reasoning and analysis, information fusion / credibility, and social network analysis. She received the best paper awards in IEEE TrustCom'2009, BRIMS'2013, IEEE GLOBECOM'2017, 2017 ARL's publication award, and IEEE CogSima 2018. She is a winner of the 2015 IEEE Communications Society William R. Bennett Prize in the Field of Communications Networking. In 2016, Dr. Cho was selected for the 2013 Presidential Early Career Award for Scientists and Engineers (PECASE), which is the highest honor bestowed by the US government on outstanding scientists and engineers in the early stages of their independent research careers. Dr. Cho earned MS and PhD degrees in computer science from the Virginia Tech in 2004 and 2008, respectively. She is a senior member of the IEEE and a member of the ACM.