Mitigating Adversarial Attacks on Data-Driven Invariant Checkers for Cyber-Physical Systems

Rajib Ranjan Maiti, Cheah Huei Yoong, Venkata Reddy Palleti, Arlindo Silva, and Christopher M. Poskitt

Abstract—The use of *invariants* in developing security mechanisms has become an attractive research area because of their potential to both prevent attacks and detect attacks in Cyber-Physical Systems (CPS). In general, an invariant is a property that is expressed using design parameters along with Boolean operators and which always holds in normal operation of a system, in particular, a CPS. Invariants can be derived by analysing operational data of various design parameters in a running CPS, or by analysing the system's requirements/design documents, with both of the approaches demonstrating significant potential to detect and prevent cyber-attacks on a CPS. While data-driven invariant generation can be fully automated, design-driven invariant generation has a substantial manual intervention. In this paper, we aim to highlight the shortcomings in data-driven invariants by demonstrating a set of adversarial attacks on such invariants. We propose a solution strategy to detect such attacks by complementing them with design-driven invariants. We perform all our experiments on a real water treatment testbed. We shall demonstrate that our approach can significantly reduce false positives and achieve high accuracy in attack detection on CPSs.

Index Terms—Cyber-physical systems, Data-driven invariants, Design-driven invariants, Axiomatic design, Adversarial attacks.

1 Introduction

Cyber-Physical Systems (CPS) consist of physical components (such as water storage tanks, pumps, and water flow sensors in a water treatment plant), software components (such as control programs running on Programmable Logic Controllers (PLCs) for coordinating among the physical components and generating appropriate control signals) and communication infrastructure (such as routers and switches for transferring data and control commands between the physical components and software components). Because CPSs increasingly utilize public networking infrastructure, such as the internet, the attack surface is becoming larger, and many successful attacks have been documented: see for example these summaries of incidents from 2010–17 on on critical infrastructure systems, such as power plants, water dams, and nuclear plants [1], [2], [3].

As a consequence, CPSs have drawn significant attention from security researchers who aim to develop different mechanisms that improve the defenses of the system under consideration. In this paper, we consider the problem of detecting adversarial attacks on a real CPS, namely the Secure Water Treatment (SWaT) testbed [4], [5], which forms our case study. We investigate the efficacy of two different types of invariant-based anomaly detectors, i) historian log-based and ii) system requirements and design-based, in the presence of adversarial attacks like the ones in [6]. In brief, an invariant is a property of a system represented in terms of a Boolean expression, e.g., "water flow rate is high" *and* "valve is open", that always holds true and hence the violation of which can indicate anomalies.

Motivation. A CPS, for instance, SWaT, is expected to operate in varying conditions (e.g., processing raw water of differing quality, or operating with backup pumps), keeping basic system parameters constant to produce a uniform quality of purified water from the plant. Any arbitrary sample of operational data logs from a historian may not capture a reasonable amount of these conditions and hence a data-driven invariant checker [7] can have inherent shortcomings for improving precision and coverage of attack/anomaly detection. Moreover, operational data-logs may be available to adversaries. For example, if the historian is compromised, the attacker can derive substantial knowledge about operating principles of the plant to be used in designing attacks (as shown in [6]). Thus, we ask if (1) adversarial attacks are feasible against data-driven invariant checkers, and (2) whether other types of invariant checkers can mitigate them?

We argue that any sample of historian logs can fail to capture sufficient variations of distinct operational conditions of a CPS and hence a data-driven invariant checker, such as [8], can be less effective at detecting certain process anomalies than those that rely on system requirements or design, such as [8], [9], [10]. These invariants can be based on design documents [9], [10] or system requirements [8]. Both of these techniques make use of design parameters (DPs), where a DP is a sensor or an actuator in the context of a CPS. Furthermore, we claim that an invariant crafted out of an arbitrary set of DPs, which is often the case in design-driven invariants [9], [10], may miss process anomalies that are not captured by these sets of DPs. In contrast, axiomatic design-driven invariants [8] are based on system

- *Rajib R. Maiti is with the Department of CSIS, BITS-Pilani, Hyderabad Campus, India, E-mail: rajib.maiti@gmail.com*
- *Venkata Reddy Palleti is with the Department of Chemical Engineering, Indian Institute of Petroleum and Energy, Vizag, India E-mail: venkat_palleti.che@iipe.ac.in*
- *Cheah Huei Yoong and Arlindo Silva are with the EPD Pillar, SUTD, Singapore E-mail: {cheahhuei_yoong, arlindo_silva}@sutd.edu.sg*
- *Christopher M. Poskitt is with the School of Computing and Information Systems, Singapore Management University, Singapore E-mail: cposkitt@smu.edu.sg*

requirements. An invariant in such a CPS corresponds to a low level requirement and hence has a lower probability of missing out any significant combination of the DPs used in addressing a requirement of a physical process in a CPS. Thus, we consider AD-based invariant checker in this paper.

Summary of Contributions. Our main contribution is in proposing a framework for creating adversarial attacks on a CPS targeting data-driven invariant checkers. Further contributions include:

- Using our proposed framework, we have designed five classes of adversarial attacks on data-driven invariant checkers and used them for investigating the efficacy of two types of invariant checkers.
- We show that the false positive rate of the data-driven invariant checker in [7] can be more than 60% with simple adversarial attacks, such as altering the state of a pair of actuators (e.g., pumps) in SWaT.
- We show that the false negative (i.e., actual attacks reported as normal behaviors) rate can be increased by up to 20% using relatively more complex adversarial attacks on the same invariant checker.
- We have developed nine new invariant checkers using axiomatic design principles [8], that are capable of detecting 4 out of 5 types of adversarial attacks that we have designed in this paper.

We show that the adversarial attacks targeting DPs across different stages, where the DPs are neither related by functional coupling nor information state coupling, cannot be detected using axiomatic invariant checkers and hence reveal the limitation of this approach.

The rest of the paper is organized as follows. Section 2 summarizes related works. Section 3 briefly describes the background of our work. Section 4 provides the problem statement. Section 5 and Section 6 discuss data-driven and axiomatic design-driven invariants. Section 7 and Section 8 respectively describe adversarial attacks and defenses. Finally, Section 9 concludes our work in this paper.

2 Related Works

Advanced attacks on CPSs such as water utilities and power systems are becoming a growing concern for security researchers [1], [2], [3]. Because of the availability of real world data sets, a wide variety of attacks have been demonstrated in water purification plants or water distribution systems [11], [12], [13] along with different defense mechanisms. The majority of the attacks aim to disturb the physical process so that the quality or the quantity of treated/distributed water can be compromised. We categorise the existing defense mechanisms of such CPSs into four broad types and briefly describe the works therein.

Anomaly detection. Based on historical data, a mathematical model is built up to characterize the normal behavior of a CPS. An anomaly detector computes the deviation between the model generated state and the observed state to decide any anomaly. The work in [14] uses the operation states in a power system and computes its deviation from that of a time-invariant model of the system. Similarly, the work in [15] considers simulated data sets and real systems like SWaT and WADI for anomaly detection. The works in [16], [17], [18], [19], [20], [21], [22] have used neural network based supervised or unsupervised machine learning models to find a match between the predicted and observed data logs for anomaly detection. Some works aim to construct formal models, for example: an event-aware finite state automata to model the event generation by the control programs of a CPS [23]; timed automata to learn the regular behavior of a plant exhibited by sensors; and Bayesian Networks to learn dependence behavior between sensor and actuators for anomaly detection [24], [25]. State estimation in a distributed setup has also been studied to detect unknown states [26]. The detection of outliers in operational log files using classical statistical methods [27], [28], [29], and the discovery of logical correlations or time-invariant *rules* among the physical properties have been proposed to detect attacks on CPSs [30], [31]. A survey [32] on anomaly detection systems highlights their limitations with respect to misclassification rates. According to the survey, the data-driven invariant checkers [7] we consider in this paper are state estimation based models, whereas our design-driven invariant checkers [8] are static linear state-space models. We assess the effectiveness of adversarial attackers in the presence of such models using data sets from the SWaT water treatment testbed [4].

Fingerprinting. Based on data/command processing, device identification is achieved. For instance, the data processing and/or response time is used for fingerprinting the sensors, actuators, or PLCs [33], [34]. The states of registers present in an ICS controller transferred in control packets can be used to fingerprint the controller [35], and the characteristics of buses carrying control packets can identify a control area network (CAN) in a CPS [36].

Fuzzing. Here, fuzzing aims to derive a set of test cases for assessing the security of CPSs. For instance, fuzzing along with machine learning is proposed for identifying vulnerabilities in CPSs [37], [38]. Manipulation of sensor readings or actuator states has been proposed for security testing of CPSs [39].

Invariant mining. These works involve discovering invariants from historian logs or system development artifacts. For instance, machine learning models can be used to automate the derivation of invariants from data logs [40], and data mining approaches, such as association rule mining and frequent item set generation, can be used for generating invariants from operational data logs [41], [42]. Furthermore, code mutation in controllers can be used to create anomalous system behaviors to investigate the effectiveness of such invariants [43]. Invariant based attack detection is also used in several types of CPSs beyond public infrastructure, such as in robotic vehicle control [44]. These techniques, however, risk missing out viable system behaviors if they are not represented in that data sufficiently.

Nevertheless, the need for manual inspection in cases of false alarms is unavoidable [45]. Also, design-based invariant checkers can be effective in the detection of stealthy and coordinated attacks [9], [10], [46]. However, both data-driven and design-driven invariants can suffer from either lack of sufficient data or lack of effective subsets of DPs [47]. The problem of lack of data is difficult to resolve and hence yields a possibility of arbitrary adversarial attacks on such anomaly detectors, e.g., using RNN deception techniques

[48]. In this paper, we use the knowledge of operational data leading to adversarial attacks on data-driven invariant checkers and that such attacks can be defended by axiomatic invariant checkers [8]. We follow a path similar to [6] to compare these types of anomaly detectors.

3 Background

3.1 Invariant

In general, an invariant is a relation, property, state, or quantity that does not change in any normal operational condition of a system. In the context of a CPS, an invariant can indicate that the readings of a sensor follow a certain distribution no mater how the overall behavior of the CPS is changing over time, or could indicate that a pair of actuators are only ever active mutually exclusively. An invariant can combine one or more DPs (a DP can be as atomic as a sensor or an actuator, or could be as broad as a subsystem) whose states remain unchanged irrespective of the operating conditions of the CPS. A violation of any invariant can potentially indicate an anomaly in the CPS. However, the violation itself may not directly identify a specific set of DPs under attack.

Formally, let $\mathbf{D} = \{d_1, d_2, ..., d_n\}$ be the set of n DPs under consideration. Each DP can assume either a set of discrete values or a set of continuous values. For instance, a water pump (P101) is a DP having discrete values as, "On" or "Off", whereas a water flow sensor (FIT101) is a DP having a set of continuous values in a particular range. Let $\vec{v}^t = \{v_1^t, v_2^t, ..., v_n^t\}$, where v_i^t is the value of d_i at time t. $\vec{v}^t$ represents the state of a CPS at t. Two such vectors $\vec{v}^t$ and $\vec{v}^{t'}$ need not be equal and hence need not represent an invariant on their own. An invariant using two DPs MV101 and FIT101 can be written as:

$$\begin{aligned} ((MV101 = Open) \wedge (FIT101! = 0)) \quad \vee \\ (MV101 = Close \wedge (FIT101 = 0)) \end{aligned} \tag{1}$$

where MV101 and FIT101 are a valve and a flow sensor respectively. It indicates that either "$((MV101 = Open)$ and $(FIT101! = 0))$" or "$((MV101 = Close)$ and $(FIT101 = 0))$" at any timestep t holds true for any normal operation of a CPS. A violation of it at any t would indicate an anomaly in the CPS. One can see that creating an exhaustive list of all possible such Boolean expressions is a complex task.

Formally, let $D' \subseteq \mathbf{D}$ be the set of DPs for which an invariant, denoted by $\mathbf{I}(D')$, be defined. Then,

$$\mathbf{I}(D') = \{B_1(D') \vee B_2(D') \vee ... \vee B_k(D')\} \tag{2}$$

where each $B_k(D')$ is a boolean expression using all the DPs in D'. Several methodologies have been proposed to derive invariants of a CPS. In this paper, we discuss two popular methods, data-driven invariants that utilizes operational data-logs, and axiomatic design-driven invariants that utilizes functional specification of CPS.

3.2 SWaT Testbed

The Secure Water Treatment (SWaT) testbed [4], [5] is a scaled-down version of a modern water purification plant, intended to support research in cyber-security solutions for critical infrastructure. SWaT is able to produce up to five gallons of safe drinking water per minute. Figure 1 shows a schematic view of the six-stage (Stage-1 to Stage-6) testbed, involving chemical processes such as ultrafiltration (UF), de-chlorination (UV), and reverse osmosis (RO). In each stage, a PLC (Programmable Logic Controller) controls the sensors and actuators through a field-bus network. The PLCs across stages communicate over Ethernet. A SCADA workstation connects a human-machine interface to all of the PLCs, facilitating monitoring and control of the plant by human operators. Sensor readings and actuator states of SWaT are recorded by a historian server at a pre-specified time interval, and a dataset is publicly available [49], [50].

A DP prefixed with "AIT" is a sensor for one water quality parameter; "FIT" a water flow sensor; "LIT" and "LS" are water/chemical level sensors in water/chemical tanks; "MV" a motorized valve; "P" a pump for pumping out water/chemical from tanks. The number xyy in the suffix of each DP indicates a stage number (i.e., x) and an identifier (i.e., yy) of the component. Any DP prefixed with AIT, FIT or AIT is a sensor; any DP prefixed with MV or P is an actuator.

TABLE 1
Data set collected from SWaT.

Data Set	#DP	#Records	Duration (in Days)	Period
I	52	4554	4	December, 2015
II	53	1441719	12	June, 2017
III	61	28801	2	July, 2019
IV	78	29992	1	December, 2019
V	82	49982	2	June, 2020

3.3 SWaT Data Set

A total of 75 DPs are present in SWaT. Depending on the requirements, a subset of 68 of these DPs is chosen for data collection during an experiment and data sets of several such experiments conducted from December, 2015 to June, 2020 have been released for research works. Our work considers five such data sets: a summary is shown in Table 1. In brief, Set I, Set II, Set III, Set IV and Set V have 52, 53, 61, 78 and 82 DPs respectively; the data is available in Excel files and every record is associated with a time step. Some of these data sets consider a stage as a feature and hence contain a maximum of 75 + 6 + 1 = 82 features. However, we do not consider a stage as a feature in this paper. Feature or DP names are not uniform across these data sets. We have unified the names and store them in 5 tables a SQLite database, one for each data set.

Only Set II has labeled data: labels are either *Normal*, indicating that a row does not represent any attack scenario, or *Attack*, indicating that a row represents some kind of attack on one or more DPs in SWaT. We do not use any of these attacks hence they are not discussed here. A total of 1,536,265 records is present in five data sets.

4 Problem Statement

When invariants are derived by analysing operational data logs of raw sensor readings and actuator states, we refer to

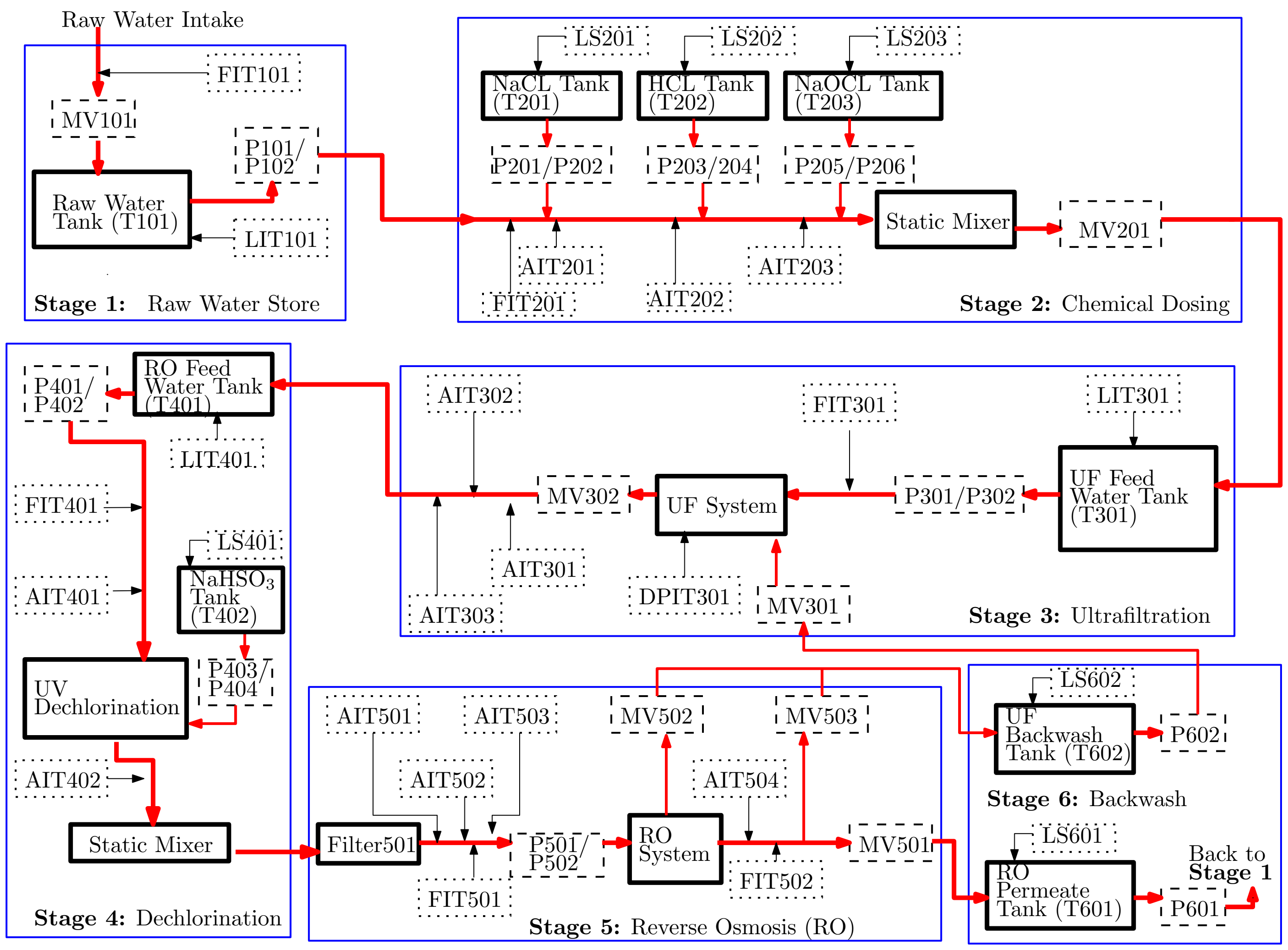

Fig. 1. Six-stage cyber-physical design of SWaT. The thick rectangles are the traditional physical components of the water treatment plant and all other rectangles are indicate its cyber infrastructure. Thick red arrows indicate water flow directions in the plant. Dotted and dashed rectangles indicate sensors and actuators respectively.

the outcomes as data-driven invariants. In order to derive effective data-driven invariants (and corresponding invariant checkers), the data logs should represent all possible non-attack behaviors of the CPS. Note that data-driven invariant checkers can only be developed *after* the CPS has been built and run for enough time to produce sufficient data logs.

In contrast, when invariants are derived by analysing the functional requirements or design documents of a CPS, we call as design-driven invariants. Those that are derived specifically by the method described in [8] are referred to as axiomatic design-driven invariants (detailed in Section 6). Unlike data-driven, design-driven invariants can be generated *before* a CPS is deployed and operational, although they involve a relatively higher amount of manual effort.

The problem in this paper is to identify the vulnerability of a data-driven invariant checker by utilizing the knowledge of operational data-logs, hence creating adversarial attacks. In particular, the aim is to craft a data sample by minimally altering a sample $\vec{v}^t$ of true class x to be predicted as a target class y ($\neq x$) with a high probability. Furthermore, we aim to explore how far this problem can be alleviated by complementing data-driven invariant checkers with design-driven invariant checkers.

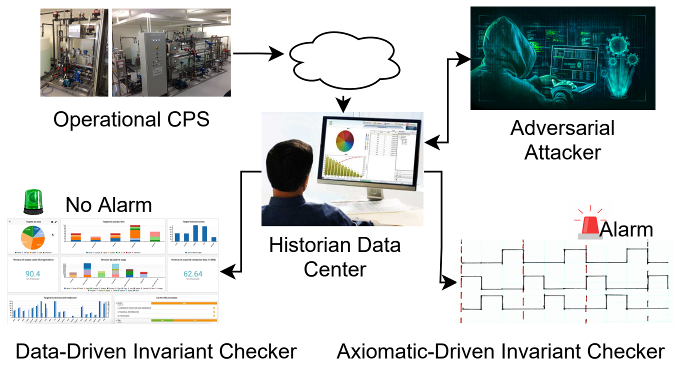

Fig. 2. System model for adversarial attack and anomaly detection systems in an operational CPS.

4.1 System Model

We envision a system (shown in Figure 2) where an operational CPS is exporting $\vec{v}^t$ at a fixed interval to a historian server over a network. Such records are then used by data- and design-driven invariant checkers for anomaly detection.

An attacker who compromises the historian system or an insider that exports data to the attacker is capable of launching adversarial attacks on the CPS.

4.2 Attacker Model

We assume an insider attack where the attacker is honest-but-curious. The attacker does not aim to exploit any real system but is interested in investigating the resilience of the system performance no matter which cyber defense mechanism is installed in the CPS. The attacker needs no access to any of the defence mechanisms, however, it has access to the historian's data logs.

The attacker has enough computing power to execute a set of statistical functions on a large set of numerical data and create synthetic data samples similar to that in the historian. The attacker does not attempt to break any cryptographic protocol that may be used to secure CPS communication.

5 Data-Driven Invariant Generation

We consider the data-driven invariant checker proposed in [42] as a case study in this paper. The set $\mathbf{D}$ of the design parameters (DPs) under consideration is classified into two groups based on the type of their values: the set $\mathbf{D}_s$ ($|\mathbf{D}_s| = n_s$) of sensors and the set $\mathbf{D}_a$ ($|\mathbf{D}_a| = n_a$) of actuators, s.t., $\mathbf{D} = \mathbf{D}_s \cup \mathbf{D}_a$. Every actuator d_i^a has a finite number of discrete states and exactly one of those state is seen at a particular time step. However, it is not guaranteed that a data set has to contain all possible states of d_i^a . Each sensor d_i^s has a range of real values. Data-driven invariants are generated in two broad steps: first, predicate generation, and second, correlated predicates mining.

5.1 Predicate Formation

Intuitively, a predicate in an invariant is the smallest sub-equation that gets satisfied at a time step. For instance, "an actuator d_i^a is ON" is a predicate which can be expressed as $d_i^a = ON$. Thus, if d_i^a exhibits a set $S_i^a = \{s_i^1, s_i^2, ..., s_i^k\}$ of k states in a data log then there are k predicates, as $d_i^a = s_i^1$, $d_i^a = s_i^2$, ..., $d_i^a = s_i^k$.

A relatively complex procedure is followed to form predicates using sensors. In brief, it is assumed that *updates* (an update is the difference in the sensor readings at times t and $t+1$) in the values of a sensor follow some certain distribution and the task is to estimate the parameters of these distributions. Specifically, a basic predicate using a sensor d_i^s can be stated as "the updates of the values of d_i^s is drawn from a normal distribution $\mathcal{N}(\mu, \sigma^2)$" and represented as $d_i^s = \mathcal{N}(\mu, \sigma^2)$. It is found that a Gaussian Mixture Model with c Gaussian distributions fits well with the updates of a sensor values. So, a predicate of a sensor d_i^s is expressed as $d_i^s = GMM(c; (\mu_1, \sigma_1^2), (\mu_2, \sigma_2^2), ..., (\mu_c, \sigma_c^2))$, where $GMM(...)$ is a Gaussian Mixture Model with c Gaussian distributions.

Another set of predicates is derived from the values of sensors that triggers a change in the state of an actuator. Let e be an event of changing state from $\mathbf{O}$pen to $\mathbf{C}$lose of a MV from time $t-1$ to t. The updates of all the sensors at t is assumed to be correlated. Hence, a linear regression model is fit for every single sensor update with the updates of all other sensors at t. Formally, let the reading of a sensor d_i^s at time step t be $x_{i,t}^s$ and $R_i^s(e)$ be the set of sensors that are *related* (defined mathematically in [42]) to an event e. Then, for every sensor reading at t, two predicates are defined: (i) $x_{i,t}^s < f_1(\epsilon, \alpha_0, \alpha_1, \alpha_2, ..., \alpha_r)$ and (ii) $x_{i,t}^s > f_2(\epsilon, \alpha_0, \alpha_1, \alpha_2, ..., \alpha_r)$, where $f_1(...)$ and $f_2(...)$ are both linear functions of coefficients $\alpha_1, \alpha_2, ..., \alpha_r$, threshold ϵ, and constant α_0; all these parameters are derived from the linear regression model. Note that the number of coefficients used in $f_1(...)$ and $f_2(...)$ need not be the same across different sensors even at a same time step and the coefficient count for a predicate on a sensor d_i^s depends on the number of sensors in $R_i^s(e)$, i.e., $|R_i^s(e)|$. Table 2 and Table 3 respectively show a small sample of actual data logs using a limited number of DPs and the corresponding predicates for a better comprehension of the output of the complete process.

TABLE 2
A sample of a data log.

Time	FIT101	LIT101	MV101	P101
3	2.653548	260.9131	1	2
4	3.668338	260.285	2	2
5	3.654815	259.8925	2	1
6	3.675456	258.0495	2	1

TABLE 3
A sample of predicates.

Time	FIT101	FIT101	LIT101	LIT101	MV101	MV101	P101	P101
	$\mathbf{N}_1$	$\mathbf{N}_2$	$\mathbf{N}_3$	$\mathbf{N}_4$	Open	Close	On	Off
1	0	1	1	0	1	0	0	1
2	0	1	1	0	1	0	0	1
3	0	1	1	0	1	0	0	1

5.2 Invariant Mining

Invariant generation is carried out using association rule mining techniques where the predicate table is taken as input. In brief, an invariant is a co-occurrence of two or more predicates at a certain time step. Two thresholds are derived for every meaningful invariant: minimum fraction threshold (γ) ranging in $(0,1)$ and minimum support threshold (θ) ranging in $(0, \gamma)$. Both γ and θ together define another threshold indicating the number of occurrences of an invariant. A set $P = \{p_1, p_2, ..., p_q\}$ of predicates can be considered as an invariant if:

$$\mathbf{F}(P) > max(\gamma min(F(p1), F(p_2), .., F(p_q)), \theta)$$

where $F(x)$ indicates the frequency of x in the predicate database. The thresholds γ and θ are chosen judiciously such that a set of predicates that occurs by chance can be avoided from being chosen as an invariant. Also, these thresholds significantly depend on the data set and the set of predicates under consideration.

Similar to the classical problem of market basket analysis, a subset of predicates P is considered to be an invariant if all the predicates in P have occurred together at least a minimum number of times in the predicate database such that no other $P' \subset P$ satisfies $F(P) = F(P')$, where $F(x)$ is frequency of x in the predicate database. Among others, Conditional Frequent Pattern-growth (CFPgrowth) and CFPgrowth++ algorithms fit well into the problem because of relatively smaller search space. The CFPGrowth++ algorithm has been selected because of the provision of applying several pruning steps to further reduce the search space. Once the set P of candidate invariants is found, it is divided into two non-empty sets P_1 and P_2 such that $P_1 \cup P_2 = P$ and $\frac{F(P_1)}{F(P_2)} = 1$. Hence, an invariant is generated as $P_1 \rightarrow P_2$ from P.

6 Axiomatic Design based Invarants

This section describes how design-driven invariants can be derived using the axiomatic design inspired approach of [8].

6.1 Axiomatic Design Principles

Axiomatic Design (AD) principles are used to transform customer requirements (called CAs) to design parameters (DPs) and process variables (PVs) through functional requirements (FRs) in order to control coupling and cohesion among the DPs. AD principles heavily rely on decomposition and mapping. Each of the high level FRs is systematically decomposed and mapped to a finer set of DPs by using a so-called design matrix $\mathbf{D}$. Typically, an entry $a_{i,k} \in \mathbf{D}$ represents a function or a method or a data structure in the software being developed and it indicates a binding between a FR and a DP. If there is no function that binds a FR, FR_i and a DP, DP_k then the entry $a_{i,k} = 0$ in the matrix $\mathbf{D}$. In an ideal design, the matrix $\mathbf{D}$ is a diagonal matrix which represents a fully uncoupled design: such a design matrix is used in the first level of mapping between FRs and DPs. This principle was used to systematically derive invariants in CPSs for the first time in [8].

In general, a FR in software engineering is an independent requirement derived from CAs, that is available in unstructured text and/or graph format and often incomplete. In SWaT, "supply water to water tanks" can be a FR in the first level and this FR can be satisfied using *digital* pumps which is a DP in the first level. The PV corresponding to this DP is the set of all possible states of the pumps. Though there are several water tanks in different stages of SWaT and each can have different operating logic, e.g., a constraint on the maximum or minimum amount of water required in the tanks, we consider that the supply of water to water tanks as a FR in the first level of design. A complete set of FRs and the corresponding DPs in SWaT are shown in Table 4.

Unlike traditional AD principles, the matrix $\mathbf{D}$ used in the derivation of invariants is not a diagonal matrix, as it incorporates "information state coupling" introduced in [8]. In brief, to address the requirement of controlling water flow a motorized valve is functionally coupled with a flow meter, whereas a pump pushing water through the flow meter has information state coupling with the FR. Such a design matrix at the first level is shown in Equation 3 (simplified from paper [8] for better comprehension) which is used for SWaT. The diagonal elements (shown as boldfaced $\mathbf{1}$s) indicate the first level direct mapping between the FRs and the DPs (same as in traditional AD approaches). The other entries are added at this level of design to indicate information state coupling with other DPs for a given FR.

$$\begin{bmatrix} FR1 \\ FR2 \\ FR3 \\ FR4 \\ FR5 \\ FR6 \\ FR7 \\ FR8 \end{bmatrix} = \begin{bmatrix} \mathbf{1} & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & \mathbf{1} & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & \mathbf{1} & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & \mathbf{1} & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \mathbf{1} & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & \mathbf{1} & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & \mathbf{1} & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & \mathbf{1} \end{bmatrix} \begin{bmatrix} DP1 \\ DP2 \\ DP3 \\ DP4 \\ DP5 \\ DP6 \\ DP7 \\ DP8 \end{bmatrix} \tag{3}$$

For instance, FR_1 is related to five other DPs (DP_2, DP_3, DP_4, DP_6 and DP_7) due to information state coupling, along with DP_1 that is related due to functional coupling.

6.2 Deriving Invariants using a Design Matrix

To derive an invariant, we need to decompose a high level FR to several lower level FRs; we call a FR as an *atomic* FR if it can be satisfied with a minimal subset of DPs along with their PVs. The matrix $\mathbf{D}$ can help in linking the DPs, d_k, with a FR, FR_i, using the equation

$$FR_i = \sum_{k=1}^{n} a_k * d_k, \text{ where } a_i \in \{0, 1\} \text{ is an entry in } \mathbf{D}$$

For instance, decomposing FR1 to a lower level, a singular FR can be "supply water to water tank T101 in stage-1 if its water level is low" which can be satisfied using MV101 and LIT101, along with their specific PVs. A precise mapping between FRs and DPs along with their PVs is obtained by repeated refinement of a high level design matrix by applying system building knowledge of a design engineer. Finally, by applying the knowledge of process logic, we can find two constraints $MV101 = Open \wedge LIT101 = Low$ and $MV101 = Close \wedge LIT101! = Low$ to feed water to T101. At any time step, the Boolean expression $(((MV101 = Open) \wedge (LIT101 = Low) \vee ((MV101 = Close) \wedge (LIT101! = Low)))$ is satisfied under normal operation and hence the expression is deemed as an invariant.

7 Formation of Adversarial Attacks

This section describes a methodology to launch adversarial attacks on data-driven invariant checkers. In a targeted adversarial attack, the attacker feeds adversarial examples so that a true class X of the examples gets predicted as a target class Y and hence defeat a trained model M. In a non-targeted adversarial attack, the aim is to cause M to classify the data as a class other than its true class X. We prepare both targeted and non-targeted adversarial attacks.

7.1 Framework for Adversarial Attack

We introduce a simple yet effective framework, in Figure 3, describing the process for generating adversarial data logs. Note that adversarial attacks on data-driven invariant checkers highly depend on the knowledge that can be derived from the historian data logs of a CPS. First, an attacker estimates the statistical properties of the DPs from the log samples. This step can help to categorise the DPs, e.g., chemical sensors, or motorised valves.

TABLE 4
Top-level FRs and corresponding DPs with their respective PVs in SWaT.

Functional Requirements (FRs)	Design Parameters (DPs)	Process Variables (PVs)
FR1: Supply water to water tanks/systems	DP1: DOL/VSD Pumps	Switch On/Off and Speed
FR2: Measure amount of water in water tanks	DP2: Water level sensors	Value range
FR3: Track flow rate of water	DP3: EMF sensors	Value range
FR4: Monitor chemical properties of water	DP4: Chemical property sensors	Value range
FR5: Inject chemicals to water	DP5: Dosing Pumps	Switch (On/Off)
FR6: Track water pressure	DP6: Pressure sensors	Value range
FR7: Direct flow of water	DP7: Motorised valves	Switch (On/Off)
FR8: Measure amount of chemicals in chemical tanks	DP8: Level switch	Value range

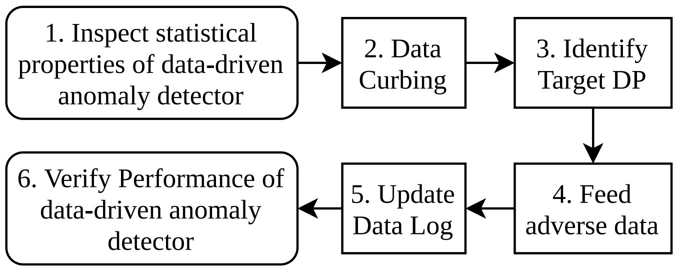
1. Inspect statistical properties of data-driven anomaly detector
2. Data Curbing
3. Identify Target DP
4. Feed adverse data
5. Update Data Log
6. Verify Performance of data-driven anomaly detector

Fig. 3. Process to create adversarial attacks on data-driven invariants.

Second, the attacker investigates and discovers the basis of data-driven invariants, e.g., the rate of change of sensor readings or possible missing states of each actuator. Third, attacker identifies a small set of n_{ad} potential DPs (s.t., $n_{ad} << |\mathbf{D}|$) that has some missing combinations of values compared to that of other similar set of DPs . The output of this step is a limited number of (DP,value(s)) pairs to form adversarial data, e.g., ($P101 = On$) and ($P102 = Off$). Fourth, the attacker seeks an appropriate time window in the actual data log to inject the crafted values identified in the third step. We consider this step as horizontal crafting of adversarial data as it can affect only a limited number of rows, instead of an entire column. For instance, a time window when the UF System in Stage-3 is back-washed using UF Backwash system in Stage-6, and during this time window, both P301 and P302 will be switched Off. Fifth, the attacker then plugs the complete rows along with the other rows in the original data log. This step is important because the attacker has to avoid sudden change in the values of $|\mathbf{D} - n_{ad}|$ DPs. Also, if a range of values, e.g., in case of a chemical sensor, needs to be injected into the data log then an appropriate distribution of the data across several timesteps needs to be generated and hence we call this as vertical crafting of adversarial records. Finally, the attacker tests the impact of the adversarial attack by selecting a set of metrics before attacking the actual plant. We use the miss-classification rate and false alarm rate for such tests.

7.2 Evaluation Metrics

We consider two well known metrics of false alarms: false negatives and false positives. Let N_a^t and N_n^t be the number of true attack and true non-attack samples respectively. Assume that a data-driven invariant checker has resulted in a confusion matrix N_{11}, N_{22} of *correctly* predicted attacks and non-attacks, and N_{21}, N_{12} *incorrectly* predicted non-attacks and attack samples. Thus, the false negative rate (FNR) = $\frac{N_{21}}{N_{21}+N_{11}}$, and false positive rate (FPR) = $\frac{N_{12}}{N_{12}+N_{22}}$. False negative and false positive rates together indicate the misclassification rate (MCR) = $\frac{N_{21}+N_{12}}{N_a^t+N_n^t}$. Finally, false alarms rate (FAR) = $\frac{N_{12}}{N_{12}+N_{11}}$ indicates the fraction of alarms due to normal operations of CPS.

An attacker is interested in the change of false alarm rate and misclassification rate due to adversarial attacks. In particular, the change in the actual attacks due to adversarial data and the corresponding change in the false alarms rate, i.e., Impact = $\frac{FAR^{ddi}-FAR^{advr}}{N_a^t-N_a^a}$, where FAR^{ddi} and FAR^{advr} indicate the false alarms rate of the data-driven invariant checker and the false alarm rate of the *adversarially* attacked data-driven invariant checker. Finally, N_a^a is the actual number of attacks after adversarial data injection.

7.3 Identify Target Actuators

Because each state of an actuator forms a predicate (refer to Section 5), we show in Figure 4 the distribution of states of each of the actuators in SWaT, aggregated from all data sets denoting non-attack operating conditions.

In Stage-1, datasets indicate that MV101 has two states (Open ($S3$) and Close ($S2$); the state $S1$ is an intermediate state that arises due to state change) and both the Pumps P101 and P102 have two states (On ($S2$) and Off ($S1$)), i.e., each actuator exhibits all possible states. In Stage-2, MV201 shows all two states, whereas the Pumps do not exhibit all possible states. Only three (P201, P203 and P205) out of 8 pumps in this stage have state $S2$, i.e., On, and all other remain Off all through out and can become a simple choice of target for adversarial attack. In Stage-3, each of four MVs and two pumps have all the possible states. In Stage-4, one (P404) out of 4 pumps has never been ON. All possible states of UV401 can be observed in the data set. In Stage-5, all MVs have changed their states, but one (P502) out of 2 pumps has never been switched ON. Finally, in Stage-6, one (P603) out of 3 pumps has never been ON. Every actuator that does not changing state can be a choice of target for attack.

Inspecting deeper in each data set separately, we have noticed that some actuators are not activated. For instance, pump P102 has not been used in 4 out of 5 data sets. Similarly, pumps P202, P204, P206, P208, P302, P402, P404, P502 and P603 have not been activated in more than one data set. In fact, some of data sets do not consider these pumps for data collections. Similarly, both MV502 and MV503 are not switched On in data set IV and, these valves are not present in three data sets, possibly because these are insignificant in certain operations of the plant.

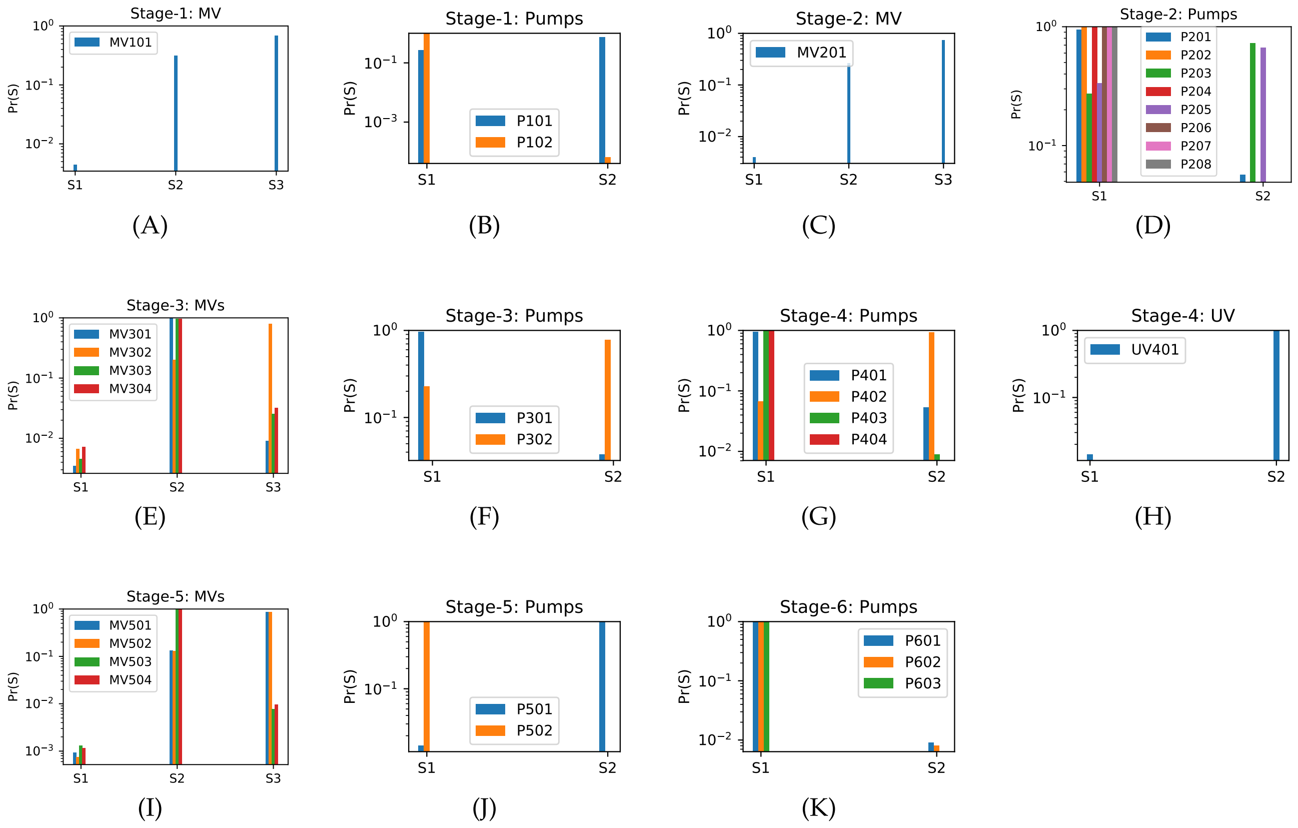

Fig. 4. Distribution of the states in each of the actuators. $Pr(x)$ indicates the probability of observing a state x, i.e, $Pr(x) = \frac{n_x}{n}$, where n and n_x are the total number of data points and the number of data points having state x of an actuator respectively.

In summary, we have found enough evidence to identify a number of actuators that can be utilized to generate adversarial attacks. Note that an arbitrary change of state of any of these DPs need not invade an anomaly detection system built-up based on data-driven invariants. Only an appropriate combination of these DPs along with other DPs can lead to an attack, which will completely depend on the functionalities of each of these DPs in the CPS.

7.4 Identify Target Sensors

In the case of sensors, we inspect the distribution of values. Our aim is not to identify an accurate set of parameters of a specific distribution fitting the sensor values, rather we aim to identify a set of sensors as target of adversarial attacks. We aggregate the values from all the data sets for each sensor and bucketize them suitably. For instance, we create buckets of size 25 value ranges from 0 to higher than 500, and a bucket size of 0.5 where the range is from 0 to less than 10. Then, we find the probability of each of such buckets for a given sensor. Mostly, the number of sensor readings is 1,481,646 and the number of buckets varies from 1 to 45 depending on the range of sensor readings and the bucket size. Figure 5 shows the results of sensors, i.e., FITs, LITs, AITs and PITs, in all the stages.

In Stage-1, FIT101 varies in [0.0,4.7] indicating water flow to tank T101 is sometimes 0 and the rate can reach to 4.7. LIT101 varies in [120, 1000], which potentially indicates that the capacity of the tank T101 is at least 1000 units. Logically, if the water level in T101 is below 120 units and FIT101 = 0, or the level above 1000 units and FIT101 = 4.7, then an anomaly should be triggered by an anomaly detector. LIT301 in Stage-3 and LIT401 in Stage-4 indicate different ranges of values; LIT301 in [132, 1201] and LIT401 in [130, 1007]. An attacker can make use of differences in the ranges to prepare attack data in combination with other DPs. The buckets of 100 in both LIT301 and LIT401 are empty even though the ranges seem valid, when compared with LIT101. Similarly, the buckets of 1000 and higher in both LIT101 and LIT401 can be used for attacks.

Unlike LITs, the ranges of AIT301 or AIT201 are not directly comparable as they measure different properties of water at different stages. *The knowledge of basic properties of water can help to decide normal ranges of values*. For instance, pH in natural water varies in [6.5, 8.5] and hence one can say that AIT202, AIT301 and AIT501 are the pH sensors; AIT201 in [6.0, 9.7], AIT301 in [6.7, 8.9] and AIT501 in [6.9, 8.3]. Therefore, the buckets of 6.0, 8.5 and 9.0 in pH sensors can be used to prepare attack data. Similarly, the sensors for other properties like dissolved oxygen and oxidation reduction potential can be used in attack formation.

7.5 Generate Adversarial Data Logs

Adversarial data represents both anomalous or non-anomalous states of a CPS and a data-driven invariant checker should detect them appropriately. We aim to craft adversarial data by manipulating existing data logs as we target only a small set of DPs.

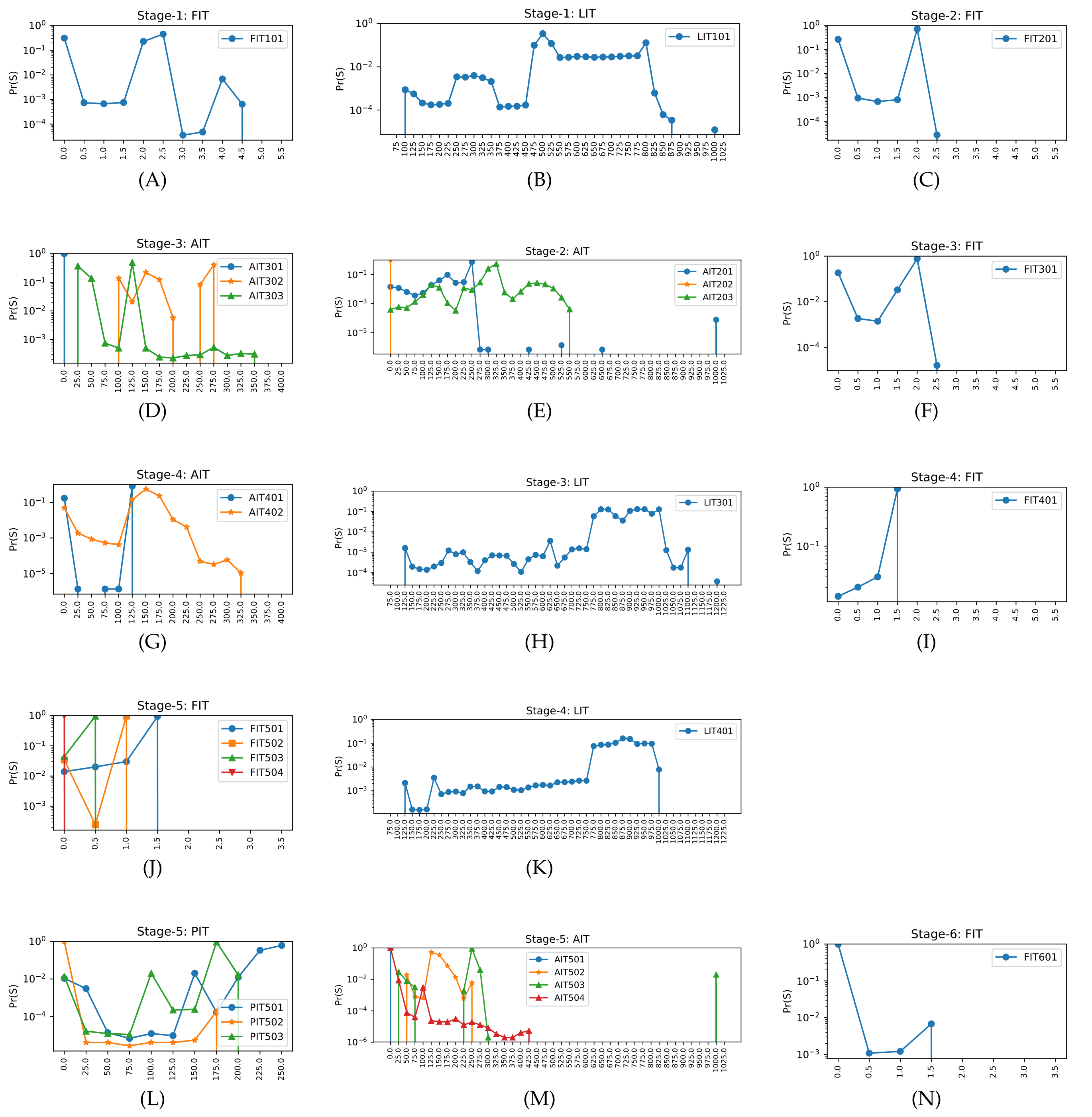

Fig. 5. Distribution of the values of each of the sensors. $Pr(R)$ is the probability of observing a range R of values, i.e., $Pr(R) = \frac{n_R}{n}$, where n and n_R are the total number of data points and the number of data points with a value in range R of a sensor respectively.

7.5.1 *Adversarial Attack on Actuator in One Stage*

We consider any pair of pumps that coordinates closely, possibly one being backup for the other, to perform a same function, e.g., {P101, P102}, {P201,P202}, and {P403,P404}. For instance, P101 and P102 can be ON alternately, but not at the same time, to represent a valid non-attack state of SWaT.

7.5.2 *Adversarial Attack on Actuator in Multiple Stages*

Our analysis shows that P602 in Stage-6 has strong coupling with MV301 in Stage-3 to perform the task of washing UF system whenever required. Working principles indicate that these two DPs can be used to create adversarial data logs for both non-attack and attack scenarios. A non-attack state is generated by putting $P602 = On$ & $MV301 = Open$, whereas an attack state can be generated with $P602 = On$ & $MV301 = Close$. However, it is important to maintain these state combinations for appropriate time periods in order to be stealthy. An anomaly state can be created if such a period is about 10 time steps.

7.5.3 *Adversarial Attack on Sensor in One Stage*

Our inspection on the range of values of LIT101 and LIT301 reveal that these two LITs are used for monitoring water level in water tanks in two stages. Our knowledge about normal operating principles indicates that the tanks at different stages are of the same capacity and water level in any of the tanks can show the same range of waters levels. Figure 5B and Figure 5H show that the lowest limit of LIT101 and LIT301 are 100 and 125 respectively and both show maximum limits of 875 and 1100 respectively (except for some outlier values). We make use of this observation to create adversarial data log wherein we create a set of synthetic values to equalize the ranges of both the DPs.

Let $\delta_{min} = |min(LIT101) - min(LIT301)|$ and $min(LIT101) < min(LIT301)$, where $min(x)$ indicate the lowest limit of x. To create additional values of LIT301, we follow a normal distribution with $\mu = \mu(LIT301)$ and $\sigma = \sigma(LIT301)$, where $\mu(LIT301)$ and $\sigma(LIT301)$ are the mean and standard deviation of $\delta_t(x) = \{(x_t - x_{t+1})\}$, x is a DP, and $(x_t - x_{t+1})$ indicates the change of values in x from time t to next time step $t + 1$ observed in the data log of LIT301. Two ranges of values are created, each of these is equal to δ_{min}; one half used to bring down the minimum of LIT301 to equalize with LIT101 and the other half to bring it up to the minimum of LIT301. Note that these extra values of LIT301 represent non-attack cases and we validated this argument with the plant engineer.

7.5.4 *Adversarial Attack on Multiple Sensors in One Stage*

We identify a correlation between LIT301 and FIT301 in Stage-3, where a reduction of water level in T301 indicates a non-zero flow rate. We create a mismatch in the range of change in the water level in T301 and the flow rate. Following a similar process as in Section 7.5.3, we create a series of 1000 values in [992, 692] for LIT301 and a series of values in [1.44, 2.44] for FIT301. All these records are labeled as "Attacks" because i) the correlation of rate of change of values in LIT301 and FIT301 is different compared to that observed in the real plant, and ii) impact of water flow is not adjusted in Stage-4 or in the remaining parts of Stage-3.

7.5.5 *Adversarial Attack on Sensor and Actuator in Multiple Stages*

Our deep inspection shows that AIT402 at Stage-4 measures one of the water quality parameters and based on this sensor certain chemical dosing is done to maintain the quality of water at this stage. The range of the values of AIT402 is seen to be {0, 325} (refer to Figure 5G). Normal data log shows that AIT402 can report a fixed value when both water flows are constant. We correlate this observation with that of the status of pumps P301 and P302 that push water from Stage-3 to T404 tank in stage 4 (refer to Figure 1). The process logic indicates that once sufficient water is pushed to T401 from Stage-3, the variation in AIT402 is limited compared to that when fresh water is fed again to the tank from Stage-3. We have utilized this logic to create the attack.

We create a sequence of about 600 readings for AIT402 that starts from 149 to 301 and back to 149 again following a procedure similar as in Section 7.5.3. This sequence of values are injected into the records after both pumps P301 and P302 become Off from a state where either one of the two are On and labeled them as "attack".

TABLE 5
Impact of adversarial attack on the checker in [42]. TN, FP, FN and TP indicate true negatives, false positives, false negatives and true positives respectively. Acc., Prec., FS indicate accuracy, precision and F-score resp.

Attack# (impact)	Base Case (–)	Sec. 7.5.1 (all)	Sec. 7.5.2 (306)	Sec. 7.5.3 (1000)	Sec. 7.5.4 (1001)	Sec 7.5.5 (601)
Target DPs	–	P101, P102	MV301, P602	LIT101, LIT301	LIT301, FIT301	AIT402, P301/2
TN	55829	321	55525	54842	53841	53240
FP	58	55566	56	58	58	58
FN	1918	0	2221	2905	3893	4490
TP	32178	34096	32181	32178	32191	32195
Acc.	0.978	0.382	0.975	0.967	0.956	0.949
FPR	0.001	0.994	0.001	0.001	0.001	0.001
FNR	0.056	0	0.065	0.083	0.108	0.122
Prec,.	0.998	0.380	0.998	0.998	0.998	0.998
Recall	0.944	1.0	0.935	0.917	0.892	0.878
FS	0.970	0.551	0.966	0.956	0.942	0.934

7.6 Impact on Data Driven Invariants

We consider a base data log file of SWaT that has a total of about 89k records including 55,887 and 34,096 non-attack and attack records respectively. The second column in Table 5 shows the performance of the data-driven invariant checker in [42] using this log file. The following rows in the table shows the impact of adversarial attacks in Section 7.5.

The attack in Section 7.5.1 has adverse impact on accuracy where it reduced to 38% from 97% in the base case. Both of the attacks in Section 7.5.1 and Section 7.5.2 have most impact on recall, indicating that the invariant checker fails to cover all the attacks. Both the attacks in Section 7.5.4 and Section 7.5.5 have most impact on both FNR and recall. In summary, the data-driven invariant checker suffers from 1 FN or FP every 100 data points which takes about 9 minutes if data points are generated every 5 seconds, i.e., 12 records per minute, in the least possible case of the attacks. Essentially, the checker can suffer from 7 FN or FP every hour in a operational plant, like SWaT. Moreover, simple alteration of pumps in SWaT can create a substantial amount of FPs (>94%) if data logs do represent the simple operational logic of this types of DPs. Alternately, the amount operational data can be irrelevant to ensure expected performance of a data-driven invariant checker.

8 Defending Against Adversarial Attacks

In this section, we consider the axiomatic design-driven invariant checkers of [8] as a defense against adversarial attacks. We claim that almost all the adversarial attacks can be detected by this type of invariant checker. In this paper for the first time, we consider a reverse engineering approach for developing AD-based invariant checkers which will aim to detect anomalies in a specific set of DPs.

8.1 AD Invariant Checker

8.1.1 *Detection of Attack on Actuators in One Stage*

This attack targets two pumps P101 and P102 in Stage-1. We consider the FRs that functionally couple these two

DPs, rather than those FRs that considers them as related due to *information state coupling*. FR1.1, i.e., "pump raw water from stage-1 to UF feed tank in stage-3", considers DP2.1 (LIT101), DP2.2 (LIT301) and DP7.1 (MV201) due to information state coupling. The notation "xxp.q" indicates that "xx" is a design element (either FR or DP) and "p.q" is the level and sub-level numbers of decomposition. Thus, DP1.2 indicates a set of DPs in the second level decomposition of DP1; similarly, DP1.2.3 is a set of DPs in a third level decomposition of DP1 or a next level decomposition of DP1.2. Using the knowledge of a plant engineer about the process logic of FR1.1, we derive the following normal operating conditions that helps in deriving the invariant:

- If LIT301 indicates low water level in tank T301, then PLC301 in Stage-3 signals to PLC201 in Stage-2, which then *O*pens MV201 and informs PLC101 to start the pumps. PLC101 then switches *O*n either P101 or P102 if water level indicated by LIT101 in T101 is not low; otherwise no pump is switched on and PLC201 *C*loses MV201.

The row indicated by FR1.1 in Table 6 shows the invariant. The rows in the the second column in Table 6 shows all the Boolean expressions that are connected with a logical *OR* operation to form the complete invariant for FR1.1.

8.1.2 Detection of Attack on Actuators in a Multiple Stages

This attack targets MV301 and P602. Either FR1.6 ("pump water to UF backwash system") and FR7.1 ("control water flow direction to UF backwash system") can be used to detect this attack.

The process logic of FR1.6 is that water can be pumped out of tank T602 using P602 when water level indicated by LS602 is not low and valve MV301 is opening; otherwise water is not pumped. The process logic of FR7.1 is similar to that of FR1.6. The invariant based on FR1.6 is shown in row FR1.6 in Table 6.

8.1.3 Detection of Attack on Single Sensor in One Stage

This attack targets LIT101 and LIT301. Unfortunately, there is no one FR that functionally couples both the DPs, nor through information state coupling. Therefore, this attack is an example where a single invariant is not sufficient to detect it. Among others, two FRs, FR2.1 ("determine water level in raw water tank in stage-1") and FR2.2 ("determine water level in UF feed tank in stage-3") can be used to detect this attack. The process logics are:

- FR2.1 is directly dependent on LIT101 and has information state coupling with P101, P102, P601, LS601 and MV101. FR2.1.1, indicates that LIT101 can be used for feeding-in water to tank T101. The process logic of FR2.1.1 is that water can be fed into T101 by using pump P601 and opening the valve MV101 if water level indicated by LIT101 is low and water level indicated by LS601 is not low.
- Similar to FR2.1.1, FR2.1.2 indicates that LIT101 can be used for pumping out water from T101. The process logic of FR2.1.2 is same as FR1.1.
- FR2.2 directly relates to LIT301 and has information state coupling with P101, P102, P301, P302 and MV201. Decomposing FR2.2, FR2.2.1 states that if LIT301 indicates a water level as low then water can be fed into the tank T301 by opening valve MV201 and switching on either pump P101 or P102. This requirement is the same as FR1.1.
- Another requirement, FR2.2.2, states that if LIT301 is not low then water can be pumped out of T301 by switching on either P302 or P301 and opening the valve MV302.

The invariants are shown in Table 6 indexed by the FRs.

8.1.4 Detection of Attack on Multiple Sensors in One Stage

There is no single FR that relates LIT301 and FIT301 (targets of this attack) together. We find FR2.2.2 (discussed before) that uses LIT301 along with other DPs, and the invariant from FR2.2.2 is already derived. FR3.3 uses FIT301 along with four other DPs, P301, P302, LIT401 and MV302, and states "measure water flow rate using FIT301 in Stage-3 when either P301 or P302 is switched On provided water level indicated by LIT401 is Low and MV302 is Open; otherwise water flow rate is zero as no pump can be switched On". The invariant is shown in row FR3.3 in Table 6.

8.1.5 Detection of Attack on Sensors and Actuators in Multiple Stages

This attack targets the sensors (AIT402, P301 and P302) of chemical properties of water and the detection of this attack is relatively more challenging. There is no single FR that relates all theses DPs together. Hence, we consider the four most relevant FRs, where each involves a subset of these DPs.

- FR4.8 ("calculate chemical properties of water in stage-4"") is directly related to AIT402 and has coupling with P401 and P402. The process logic is that AIT402 should detect low levels of chemicals if none of the pumps are on; otherwise, it should show a non-low level of chemicals as fresh water is pumped using either P401 or P402.
- FR5.1 ("pump chemicals into water from Tank $T402$ in stage-4") is directly related to P403 and P404 and has coupling with AIT402 and LS401. The process logic is that if chemical level indicated by AIT402 is low then pump chemicals into water using either P403 or P404 provided the amount of chemical indicated by LS401 in the chemical tank is not low.
- FR1.2 ("pump water from Stage-3 to RO feed tank in Stage-4") is directly related to P301 and P302 and has coupling with LIT301, MV302 and LIT401. The Process logic is that if water-level shown by LIT401 is low then water is pumped using either P301 or P304 through the valve MV302 (=On) provided the water level indicated by LIT301 is not low.
- FR7.4 ("control water flow direction from stage-3 to stage-4") is directly related to MV302 and has coupling with P301, P302 and LIT401. The process logic is that MV302 is open whenever either P301 or P302 is switched on provided LIT401 is not high.

The coupling in each of these FRs is information state coupling and the invariants are shown in Table 6.

TABLE 6
Invariant derived from process logic of various FRs.

FR	Boolean Expression	Label
FR1.1	$(LIT101! = Low) \wedge (P101 = On) \wedge (P102 = Off) \wedge (MV201 = Open) \wedge (LIT301 = Low)$	Non-anomaly
	$(LIT101! = Low) \wedge (P101 = Off) \wedge (P102 = On) \wedge (MV201 = Open) \wedge (LIT301 = Low)$	Non-anomaly
	$(LIT101! = Low) \wedge (P101 = On) \wedge (P102 = Off) \wedge (MV201 = Open) \wedge (LIT301! = High)$	Non-anomaly
	$(LIT101! = Low) \wedge (P101 = Off) \wedge (P102 = On) \wedge (MV201 = Open) \wedge (LIT301! = High)$	Non-anomaly
	$(LIT101! = Low) \wedge (P101 = Off) \wedge (P102 = Off) \wedge (MV201 = Close) \wedge (LIT301! = Low)$	Non-anomaly
	$(LIT101 = Low) \wedge (P101 = Off) \wedge (P102 = Off) \wedge (MV201 = Close) \wedge (LIT301! = Low)$	Non-anomaly
	$(LIT101 = Low) \wedge (P101 = Off) \wedge (P102 = Off) \wedge (MV201 = Close) \wedge (LIT301 = Low)$	Non-anomaly
FR1.6	$(LS602! = Low) \wedge (MV301 = Open) \wedge (P602 = On)$	Non-anomaly
	$(LS602 = Low) \wedge (MV301 = Close) \wedge (P602 = Off)$	Non-anomaly
	$(LS602! = Low) \wedge (MV301 = Close) \wedge (P602 = Off)$	Non-anomaly
FR2.1.1	$(LIT101 = Low) \wedge (MV101 = Open) \wedge (LS601! = Low) \wedge (P601 = On)$	Non-anomaly
	$(LIT101! = High) \wedge (MV101 = Open) \wedge (P601 = On) \wedge (LS601! = Low)$	Non-anomaly
	$(LIT101! = Low) \wedge (MV101 = Close) \wedge (LS601! = Low) \wedge (P601 = Off)$	Non-anomaly
	$(LIT101! = Low) \wedge (MV101 = Close) \wedge (LS601 = Low) \wedge (P601 = Off)$	Non-anomaly
FR2.2.2	$(LIT301! = Low) \wedge (P301 = On) \wedge (P302 = Off) \wedge (MV302 = Open)$	Non-anomaly
	$(LIT301! = Low) \wedge (P301 = Off) \wedge (P302 = On) \wedge (MV302 = Open)$	Non-anomaly
	$(LIT301! = Low) \wedge (P301 = Off) \wedge (P302 = Off) \wedge (MV302 = Close)$	Non-anomaly
	$(LIT301 = Low) \wedge (P301 = Off) \wedge (P302 = Off) \wedge (MV302 = Close)$	Non-anomaly
FR3.3	$(LIT401 = Low) \wedge (P301 = On) \wedge (P302 = Off) \wedge (MV302 = Open) \wedge (FIT301! = 0)$	Non-anomaly
	$(LIT401 = Low) \wedge (P301 = Off) \wedge (P302 = On) \wedge (MV302 = Open) \wedge (FIT301! = 0)$	Non-anomaly
	$(LIT401! = Low) \wedge (P301 = Off) \wedge (P302 = Off) \wedge (MV302 = Close) \wedge (FIT301 = 0)$	Non-anomaly
	$(LIT401 = Low) \wedge (P301 = Off) \wedge (P302 = Off) \wedge (MV302 = Close) \wedge (FIT301 = 0)$	Non-anomaly
FR4.8	$(AIT402! = Low) \wedge (P401 = On) \wedge (P402 = Off)$	Non-anomaly
	$(AIT402! = Low) \wedge (P401 = Off) \wedge (P402 = On)$	Non-anomaly
	$(AIT402 = Low) \wedge (P401 = Off) \vee (P402 = Off)$	Non-anomaly
FR5.1	$(P403 = On) \wedge (P404 = Off) \wedge (LS401! = Low) \wedge (AIT402 = Low)$	Non-anomaly
	$(P403 = Off) \wedge (P404 = On) \wedge (LS401! = Low) \wedge (AIT402 = Low)$	Non-anomaly
	$(P403 = Off) \wedge (P402 = Off)) \wedge (LS401! = Low) \wedge (AIT402 = High)$	Non-anomaly
	$(P403 = Off) \wedge (P402 = Off)) \wedge (LS401 = Low) \wedge (AIT402 = High)$	Non-anomaly
FR1.2	$(LIT301! = Low) \wedge (P301 = On) \wedge (P302 = Off) \wedge (MV302 = Open) \wedge (LIT401 = Low)$	Non-anomaly
	$(LIT301! = Low) \wedge (P301 = Off) \wedge (P302 = On) \wedge (MV302 = Open) \wedge (LIT401 = Low)$	Non-anomaly
	$(LIT301! = Low) \wedge (P301 = Off) \wedge (P302 = Off) \wedge (MV302 = Close) \wedge (LIT401! = Low)$	Non-anomaly
	$(LIT301 = Low) \wedge (P301 = Off) \wedge (P302 = Off) \wedge (MV302 = Close) \wedge (LIT401! = Low)$	Non-anomaly
FR1.2	$(MV302 = Open) \wedge (P301 = On) \wedge (P302 = Off) \wedge (LIT401! = High)$	Non-anomaly
	$(MV302 = Open) \wedge (P301 = Off) \wedge (P302 = On) \wedge (LIT401! = High)$	Non-anomaly
	$(MV302 = Close) \wedge (P301 = Off) \wedge (P302 = Off) \wedge (LIT401! = Low)$	Non-anomaly

8.2 Performance of Axiomatic Detector

An AD-based invariant checker is built based on one invariant, i.e., such a checker is limited to detect attacks on the DPs present in the invariant. Work in [8] has reported four such invariant checkers. In this paper, we derive nine new such invariant checkers, shown in Table 6, and an anomaly is reported if at least one of them reports anomaly.

Each of these checkers is built using decision tree based supervised machine learning as in [8]. In brief, for each invariant, a decision tree model is built with a training set consisting of only those records that are sufficient to detect anomalies. For instance, in FR1.1, we have 7 rows of non-anomalies and the remaining 25 rows as anomalies, and hence training is done only with 32 records, but testing is done on 1000 records created randomly that are labeled based on the invariant in Table 6 to generate ground truth. We have observed that, for every attack, one of the nine checkers raises alarm for entire duration of the attack. For instance, the duration of one instance of our second attack is 10 timesteps, and 10 out of 10 timesteps of our invariant checkers indicates an attack (note that the first three timesteps should not be considered as an attack as they represent sensor settlement delay). In our first attack, none of the AD-based invariant checkers raised anomaly. In general, we observe that these checkers have not missed out any attack in Table 6. However, these checkers are limited to detect attacks that target the DPs corresponding to a particular FR and to detect state transition effects in CPSs.

8.3 Limitations of Axiomatic Detectors

Our proposed methodology is constrained by two factors. First, the construction of FRs that lead to the creation of invariants can differ significantly if the designer's knowledge of the CPS requirements is limited. Second, the generation of adversarial data logs to defeat data-driven invariant checkers is dependent on the capability of the attacker's understanding of the operational data logs. However, our approach is extensible to other similar CPSs that disclose a significant amount of historian data logs and some information about the data-driven invariant checker.

9 Conclusion

In this paper, we have proposed a framework for preparing adversarial attacks targeting data-driven invariant checkers. We categorise the attacks into five groups based on the type and stage of the targeted DPs. Considering a real testbed, namely SWaT, we show that the false alarms can be as high as about 80% with our adversarial attacks. We have designed nine new axiomatic design-driven invariant checkers to detect adversarial attacks and our results show that the these invariant checkers do not raise any false alarms. However, while our design-driven invariant checkers are robust against adversarial attacks (i.e., they are not caused to report false positives), the checkers are not able to detect that an attack is being attempted. Our study in this paper shows interesting research directions in blending the advantages of different kinds of invariant checkers, including (axiomatic) design-driven and data-driven, as well as the identification of DPs that are targets of adversarial attacks.

ACKNOWLEDGMENTS

This research/project is supported by the NRF, Singapore, under its National Satellite of Excellence Programme "Design Science and Technology for Secure Critical Infrastructure" (Award Number: NSoE_DeST-SCI2019-0004). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of NRF, Singapore.

REFERENCES

[1] M. Al-Mhiqani, R. Ahmad, W. Mohamed, A. Hassan, Z. Z. Abidin, N. Ali, and K. Abdulkareem, "Cyber-security incidents: A review cases in cyber-physical systems," *International Journal of Advanced Computer Science and Applications*, vol. 9, pp. 499–508, 2018.

[2] L. Maglaras, M. A. Ferrag, A. Derhab, M. Mukherjee, H. Janicke, and S. Rallis, "Threats, countermeasures and attribution of cyber attacks on critical infrastructures," *Security and Safety*, vol. 5, pp. 1–9, 12 2018.

[3] R. Clark, S. Panguluri, T. Nelson, and R. Wyman, "Protecting drinking water utilities from cyber threats," *Journal - American Water Works Association*, vol. 109, pp. 50–58, 02 2017.

[4] "Secure Water Treatment (SWaT)," https://itrust.sutd.edu.sg/itrust-labs-home/itrust-labs_swat/, 2020, accessed: May 2022.

[5] A. P. Mathur and N. O. Tippenhauer, "Swat: a water treatment testbed for research and training on ICS security," in *Proc. International Workshop on Cyber-physical Systems for Smart Water Networks (CySWater@CPSWeek 2016)*. IEEE Computer Society, 2016, pp. 31–36.

[6] A. Erba and N. O. Tippenhauer, "No need to know physics: Resilience of process-based model-free anomaly detection for industrial control systems," 2020.

[7] V. R. Palleti, J. V. Joseph, and A. Silva, "A contribution of axiomatic design principles to the analysis and impact of attacks on critical infrastructures," *International Journal of Critical Infrastructure Protection*, vol. 23, pp. 21–32, 2018.

[8] C. H. Yoong, V. R. Palleti, R. R. Maiti, A. Silva, and C. M. Poskitt, "Deriving invariant checkers for critical infrastructure using axiomatic design principles," *Springer Cybersecurity*, vol. 4, 2021.

[9] S. Adepu and A. Mathur, "Distributed detection of single-stage multipoint cyber attacks in a water treatment plant," in *Proc. of ACM Asia Conference on Computer and Communications Security (AsiaCCS)*. ACM, 2016, pp. 449–460.

[10] ——, "Distributed attack detection in a water treatment plant: Method and case study," *IEEE Transactions on Dependable and Secure Computing*, 2018.

[11] A. Hassanzadeh, A. Rasekh, S. Galelli, M. Aghashahi, R. Taormina, A. Ostfeld, and M. K. Banks, "A review of cybersecurity incidents in the water sector," *Journal of Environmental Engineering*, 09 2019.

[12] J. Leyden, "Water treatment plant hacked, chemical mix changed for tap supplies," *The Register*, 2016, accessed: July, 2020. [Online]. Available: https://www.theregister.com/2016/03/24/water_utility_hacked/

[13] ICS-CERT Alert, "Cyber-attack against Ukrainian critical infrastructure," https://ics-cert.us-cert.gov/alerts/IR-ALERT-H-16-056-01, 2016, document number: IR-ALERT-H-16-056-01.

[14] F. Pasqualetti, F. Dorfler, and F. Bullo, "Cyber-physical attacks in power networks: Models, fundamental limitations and monitor design," in *Proc. IEEE Conference on Decision and Control and European Control Conference (CDC-ECC 2011)*. IEEE, 2011, pp. 2195–2201.

[15] W. Aoudi, M. Iturbe, and M. Almgren, "Truth will out: Departure-based process-level detection of stealthy attacks on control systems," in *Proc. ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2018, pp. 817–831.

[16] J. Goh, S. Adepu, M. Tan, and Z. S. Lee, "Anomaly detection in cyber physical systems using recurrent neural networks," in *Proc. of IEEE International Symposium on High Assurance Systems Engineering*, 2017, pp. 140–145.

[17] J. Kim, J. Yun, and H. C. Kim, "Anomaly detection for industrial control systems using sequence-to-sequence neural networks," in *Proc. International Workshop on the Security of Industrial Control Systems and Cyber-Physical Systems (CyberICPS 2019)*, ser. LNCS, vol. 11980. Springer, 2019, pp. 3–18.

[18] J. Inoue, Y. Yamagata, Y. Chen, C. M. Poskitt, and J. Sun, "Anomaly detection for a water treatment system using unsupervised machine learning," in *Proc. of IEEE International Conference on Data Mining Workshops: Data Mining for Cyberphysical and Industrial Systems*, 2017, pp. 1058–1065.

[19] Z. He, A. Raghavan, G. Hu, S. M. Chai, and R. B. Lee, "Power-grid controller anomaly detection with enhanced temporal deep learning," in *Proc. IEEE International Conference On Trust, Security And Privacy In Computing And Communications (TrustCom)*. IEEE, 2019, pp. 160–167.

[20] M. Kravchik and A. Shabtai, "Detecting cyber attacks in industrial control systems using convolutional neural networks," in *Proc. Workshop on Cyber-Physical Systems Security and PrivaCy (CPS-SPC 2018)*. ACM, 2018, pp. 72–83.

[21] P. Schneider and K. Böttinger, "High-performance unsupervised anomaly detection for cyber-physical system networks," in *Proc. of Workshop on Cyber-Physical Systems Security and PrivaCy (CPS-SPC)*. ACM, 2018, pp. 1–12.

[22] M. A. M. Carrasco and C. Wu, "An unsupervised framework for anomaly detection in a water treatment system," in *Proc. of IEEE International Conference On Machine Learning And Applications*, 2019, pp. 1298–1305.

[23] L. Cheng, K. Tian, and D. D. Yao, "Orpheus: Enforcing cyber-physical execution semantics to defend against data-oriented attacks," in *Proc. Annual Computer Security Applications Conference (ACSAC)*. ACM, 2017, pp. 315–326.

[24] Q. Lin, S. Adepu, S. Verwer, and A. Mathur, "TABOR: A graphical model-based approach for anomaly detection in industrial control systems," in *Proc. Asia Conference on Computer and Communications Security (AsiaCCS)*. ACM, 2018, pp. 525–536.

[25] V. Narayanan and R. B. Bobba, "Learning based anomaly detection for industrial arm applications," in *Proc. Workshop on Cyber-Physical Systems Security and PrivaCy (CPS-SPC 2018)*. ACM, 2018, pp. 13–23.

[26] S. Adepu, F. Brasser, L. Garcia, M. Rodler, L. Davi, A. Sadeghi, and S. A. Zonouz, "Control behavior integrity for distributed cyber-physical systems," in *Proc. of ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*, 2020, pp. 30–40.

[27] Y. Harada, Y. Yamagata, O. Mizuno, and E. Choi, "Log-based anomaly detection of CPS using a statistical method," in *Proc. of IEEE International Workshop on Empirical Software Engineering in Practice*, 2017, pp. 1–6.

[28] C. M. Ahmed, M. Ochoa, J. Zhou, A. P. Mathur, R. Qadeer, C. Murguia, and J. Ruths, "*NoisePrint*: Attack detection using sensor and process noise fingerprint in cyber physical systems," in *Proc. of Asia Conference on Computer and Communications Security (AsiaCCS)*. ACM, 2018, pp. 483–497.

[29] C. M. Ahmed, J. Zhou, and A. P. Mathur, "Noise matters: Using sensor and process noise fingerprint to detect stealthy cyber attacks and authenticate sensors in CPS," in *Proc. of Annual Computer Security Applications Conference (ACSAC)*. ACM, 2018, pp. 566–581.

[30] E. Aggarwal, M. Karimibiuki, K. Pattabiraman, and A. Ivanov, "CORGIDS: A correlation-based generic intrusion detection system," in *Proc. of ACM Workshop on Cyber-Physical Systems Security and Privacy*, 2018, pp. 24–35.

[31] T. K. Das, S. Adepu, and J. Zhou, "Anomaly detection in industrial control systems using logical analysis of data," *Computers & Security*, vol. 96, 2020.

[32] J. Giraldo, D. I. Urbina, A. Cardenas, J. Valente, M. A. Faisal, J. Ruths, N. O. Tippenhauer, H. Sandberg, and R. Candell, "A survey of physics-based attack detection in cyber-physical systems," *ACM Computing Surveys*, vol. 51, no. 4, pp. 76:1–76:36, 2018.

[33] D. Formby, P. Srinivasan, A. M. Leonard, J. D. Rogers, and R. A. Beyah, "Who's in control of your control system? device fin-

gerprinting for cyber-physical systems," in *Proc. Annual Network and Distributed System Security Symposium (NDSS)*. The Internet Society, 2016.

[34] Q. Gu, D. Formby, S. Ji, H. Cam, and R. A. Beyah, "Fingerprinting for cyber-physical system security: Device physics matters too," *IEEE Security & Privacy*, vol. 16, no. 5, pp. 49–59, 2018.

[35] K. Yang, Q. Li, X. Lin, X. Chen, and L. Sun, "ifinger: Intrusion detection in industrial control systems via register-based fingerprinting," *IEEE Journal of Selected Areas in Communications*, vol. 38, no. 5, pp. 955–967, 2020.

[36] M. Kneib and C. Huth, "Scission: Signal characteristic-based sender identification and intrusion detection in automotive networks," in *Proc. of ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 787–800.

[37] Y. Chen, C. M. Poskitt, J. Sun, S. Adepu, and F. Zhang, "Learning-guided network fuzzing for testing cyber-physical system defences," in *Proc. IEEE/ACM International Conference on Automated Software Engineering (ASE 2019)*. IEEE Computer Society, 2019, pp. 962–973.

[38] Y. Chen, B. Xuan, C. M. Poskitt, J. Sun, and F. Zhang, "Active fuzzing for testing and securing cyber-physical systems," in *Proc. ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA)*. ACM, 2020.

[39] H. Wijaya, M. Aniche, and A. Mathur, "Domain-based fuzzing for supervised learning of anomaly detection in cyber-physical systems," in *Proc. International Workshop on Engineering and Cybersecurity of Critical Systems (EnCyCriS 2020)*. ACM, 2020.

[40] Y. Chen, C. M. Poskitt, and J. Sun, "Towards learning and verifying invariants of cyber-physical systems by code mutation," in *Proc. of International Symposium on Formal Methods (FM)*, vol. 9995. Springer, 2016, pp. 155–163.

[41] K. Pal, S. Adepu, and J. Goh, "Effectiveness of association rules mining for invariants generation in cyber-physical systems," in *IEEE International Symposium on High Assurance Systems Engineering*, 2017, pp. 124–127.

[42] C. Feng, V. R. Palleti, A. Mathur, and D. Chana, "A systematic framework to generate invariants for anomaly detection in industrial control systems," in *Proc. Annual Network and Distributed System Security Symposium (NDSS 2019)*. The Internet Society, 2019.

[43] Y. Chen, C. M. Poskitt, and J. Sun, "Learning from mutants: Using code mutation to learn and monitor invariants of a cyber-physical system," in *Proc. of IEEE Symposium on Security and Privacy*, 2018, pp. 648–660.

[44] H. Choi, W. Lee, Y. Aafer, F. Fei, Z. Tu, X. Zhang, D. Xu, and X. Xinyan, "Detecting attacks against robotic vehicles: A control invariant approach," in *Proc. ACM SIGSAC Conference on Computer and Communications Security (CCS 2018)*. ACM, 2018, pp. 801–816.

[45] A. A. Cárdenas, S. Amin, Z. Lin, Y. Huang, C. Huang, and S. Sastry, "Attacks against process control systems: risk assessment, detection, and response," in *Proc. of ACM Symposium on Information, Computer and Communications Security (AsiaCCS)*. ACM, 2011, pp. 355–366.

[46] S. Adepu and A. Mathur, "Using process invariants to detect cyber attacks on a water treatment system," in *Proc. of International Conference on ICT Systems Security and Privacy Protection (SEC)*, vol. 471. Springer, 2016, pp. 91–104.

[47] M. A. Umer, A. Mathur, K. N. Junejo, and S. Adepu, "Integrating design and data centric approaches to generate invariants for distributed attack detection," in *Proceedings of the 2017 Workshop on Cyber-Physical Systems Security and PrivaCy*. Association for Computing Machinery, 2017, p. 131–136.

[48] Y. Jia, J. Wang, C. M. Poskitt, S. Chattopadhyay, J. Sun, and Y. Chen, "Adversarial attacks and mitigation for anomaly detectors of cyber-physical systems," *CoRR*, vol. abs/2105.10707, 2021. [Online]. Available: https://arxiv.org/abs/2105.10707

[49] "iTrust Labs: Datasets," https://itrust.sutd.edu.sg/itrust-labs$_{d}atasets/$, 2020, $accessed : May$2022.

[50] J. Goh, S. Adepu, K. N. Junejo, and A. Mathur, "A dataset to support research in the design of secure water treatment systems," in *Proc. of International Conference on Critical Information Infrastructures Security*, 2016.

Rajib Ranjan Maiti is currently an Assistant Professor in CSIS, BITS Pilani, Hyderabad campusn India. He has done his PhD in CSE at IIT Kharagpur, India. His research interest lies in the area of Cyber Security in IoT and CPS. He has published his research works in journals like Transaction on Mobile Computing, Computer Networks and Cybersecurity, and conferences like Esorics, WiSec and AsiaCCS. He is currently executing two sponsored projects related to cyber security, one funded by SERB, DST, India and the other funded by Axiado, India.

Andrew Yoong Cheah Huei has received a PhD from National University of Singapore (NUS), Singapore. He studied at Iowa State University, USA for both MSc and BSc. He has over 20 years of experience as a trainer and a teacher. He has taught many areas including cybersecurity, machine learning, IOT, networking, operating system, mobile applications, and Python programming. He has experienced in research and teaching in industrial control systems (ICS). His current research interests include cybersecurity, machine learning, ICS, IOT, education computing, and data mining. He has many publications in peer-reviewed journals and conferences. He has served as a reviewer for many conferences and journals.

Venkata Reddy Palleti is a assistant professor at Indian institute of petroleum and energy-Visakhapatnam. Before joining this, he worked as a research fellow at iTrust, SUTD, Singapore. during his Postdoctoral Researcher he worked on the design of secured cyber-physical systems .He has published several research articles in the major conferences like NDSS, ICC and journals like ACM TCPS, Computers and Chemical Engineering and Springer Cyber Security. His research interests are in cyber physical systems, water distribution systems and data analytics.

Arlindo Silva has a PhD in Mechanical Engineering. His current research interests rest on engineering design, product development, creativity, materials selection methodologies, additive manufacturing in composite structures, cost modelling and management of uncertainty in design. He published over a hundred and fifty papers in journals, conferences and book chapters, more than 50 patents and authored/co-edited five books in engineering related topics. He received the MIT-Portugal Education Innovation Award in 2009 and was a Professor of Excellence at the University of Lisbon in 2009, 2013, 2014 and 2015, before joining the Singapore University of Technology and Design as an Associate Professor with the Engineering Product Development Pillar. He is the current NAMIC (National Additive Manufacturing Innovation Cluster) Hub Director at SUTD, liaising Singaporean companies with SUTD's expertise in Additive Manufacturing and Composites Technologies through the DManD (Digital Manufacturing and Design) Center.

Christopher M. Poskitt is an Assistant Professor of Computer Science (Education) at Singapore Management University (SMU), where he is a member of the Software Analysis and Verification Group. Prior to SMU, he held research and teaching positions at ETH Zürich, Switzerland, and the Singapore University of Technology and Design. His research broadly addresses the problem of engineering correct and secure software/systems, towards which he has co-developed techniques for testing/defending cyber-physical systems, tools for analysing execution models of concurrency APIs, and logics for reasoning about the correctness of graph-rewriting programs. His research interests span software engineering, formal methods, cybersecurity, and computer science education.