This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2018.2875008, IEEE Transactions on Dependable and Secure Computing

Distributed Attack Detection in a Water Treatment Plant: Method and Case Study

Sridhar Adepu and Aditya Mathur
iTrust, Center for Research in Cyber Security
Singapore University of Technology and Design
Email: adepu_sridhar@mymail.sutd.edu.sg, aditya_mathur@sutd.edu.sg

***Abstract*—The rise in attempted and successful attacks on critical infrastructure, such as power grid and water treatment plants, has led to an urgent need for the creation and adoption of methods for detecting such attacks often launched either by insiders or state actors. This paper focuses on one such method that aims at the detection of attacks that compromise one or more actuators and sensors in a plant either through successful intrusion in the plant's communication network or directly through the plant computers. The method, labelled as Distributed Attack Detection (DAD), detects attacks in real-time by identifying anomalies in the behavior of the physical process in the plant. Anomalies are identified by using monitors that are implementations of invariants derived from the plant design. Each invariant must hold either throughout the plant operation, or when the plant is in a given state. The effectiveness of DAD was assessed experimentally on an operational water treatment plant named SWaT that is a near-replica of commercially available large treatment plants. The method used in DAD was found to be effective in detecting stealthy and coordinated attacks.**

***Index Terms*—Cyber Physical Systems, Cyber Security, Coordinated Attacks, Distributed Attack Detection, Industrial Control Systems, Invariants, SCADA, Water Treatment Plant.**

I. Introduction

A Cyber Physical System (CPS, also referred to here as a *plant*, consists of a physical process controlled by a computation and communications infrastructure. Public utilities, such as a power grid or a water treatment plant are CPS. The computation and communications infrastructure is also referred to as an Industrial Control System (ICS). CPS relevant to this work contain one or more Programmable Logic Controllers (PLCs) each with software for computing and effecting control actions. The control code in a PLC is executed in successive *scan* cycles. State information is input at the start of a scan cycle, control actions computed, and effected via appropriate commands sent to the actuators. Control actions are based on the current state of the system estimated using readings obtained through a network of sensors.

Motivation: CPS have been subject to successful cyber and physical attacks including those on nuclear enrichment plant [28], power grid [26], water treatment systems [2], and several other unsuccessful attempts [21]. Industrial Control System Cyber Emergency Response Team (ICS-CERT) reports [14], [21], tend to confirm that attacks on ICS are increasing rapidly as hackers and terrorist activists target a variety of critical infrastructures. Given the criticality of such systems, it is important to develop effective methods for detecting cyber and physical attacks. This paper focuses on one such method, namely Distributed Attack Detection, hereafter referred to as DAD.

This work was supported by research grant 9013102373 from the Ministry of Defense and NRF2014-NCR-NCR001-040 from the National Research Foundation, Singapore.

Problem context and statement: A CPS relevant to this work consists of one or more process *stages* where each stage is controlled by a PLC communicating over a network with other PLCs, a master controller workstation often referred to as Supervisory Control and data Acquisition System (SCADA), and one or more Human Machine Interfaces (HMI). Each stage contains sensors and actuators.

A sensor is considered compromised when an adversary modifies the sensor reading sent to a PLC. An actuator is considered compromised when an adversary takes over its control from the PLC. A PLC is compromised when its firmware or the control code has been modified by an adversary. A process stage is compromised when *all* sensors and actuators in that stage, and the host PLC, are compromised. When some, but not all, sensors, actuators, and PLCs are compromised, the stage is considered *partially* compromised. Depending on the level of access to the system, an adversary may manipulate controls or sensor readings either via the communications network, the Supervisory Control and Data Acquisition (SCADA) workstation, or by physically tampering the device. Such manipulations and tampering are referred to as, respectively, *cyber* and *physical* attacks. The term *attack* is used in the following to refer to both cyber and physical attacks or a combination thereof. A *coordinated attack* is one where an attacker, or a group of attackers, takes control of multiple components in a CPS to avoid, or limit the chances of, detection. A *process anomaly* is said to exist when a process moves to a state that does not conform to its design specifications.

Sources of anomalous behavior: Attacks as well as equipment faults, design and coding errors in control software, and human errors, are potential sources process anomalies. Control codes resident in PLCs do account for the possibility of component failure and generate appropriate alerts when components fail. However, there exists the possibility that failure of a CPS component, and not an attack, may lead an invariant to

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2018.2875008, IEEE Transactions on Dependable and Secure Computing

generate an alert. DAD does not aim at identifying the source of an anomaly. However, alerts issued by DAD could be useful in determining the location within a process stage where an anomaly has been detected.

Novelty of the proposed approach: DAD introduces the notion of *state entanglement* and uses it to derive invariants from the design of the CPS under consideration. Unlike in the methods proposed earlier that rely mostly on state estimation for detecting process anomalies [13], [30], [36], [53], the invariants use both discrete and continuous state variables. The invariants so derived do not require estimates of state variables; instead, the plant is observed and the relationships across state variables checked for consistency while the process runs. This approach is thus different from approaches such as CUSUM [8], [29] that detect changes in a process through state estimation and online manipulation of time series data generated by a sensor. In addition to using the invariants, DAD also uses a simpler version of the CUSUM method that does not require computation of the bias from an ongoing process, and is based exclusively on continuous state variables in the underlying process. While the novelty and generality of the methods used in DAD is a key advancement in the design of secure CPS, their experimental evaluation in an operational water treatment plant, in contrast to doing so in a simulation environment, adds to our claim of effectiveness in detecting multiple coordinated attacks. In summary, the novelty of DAD lies in a systematic procedure (a) for constructing process-based invariants from plant design, that integrate discrete and continuous state variables, for detecting stealthy and coordinated cyber attacks in a distributed manner, (b) whose effectiveness has been evaluated experimentally on a realistic plant using both *single–* and *multi-point* coordinated attack, and (c) that rarely raises false alarms.

Contributions: (a) A distributed attack detection (DAD) mechanism for detecting cyber and physical attacks. (b) Experimental assessment of DAD in a realistic testbed by the authors and independent teams from academia and industry.

Organization: The remainder of this work is organized as follows. Section II is an overview of DAD. Section III presents a case study to evaluate the effectiveness of DAD in detecting cyber attacks. Several key issues with DAD, that remain to be addressed before DAD can be widely deployed in operational and new commercial plants, are discussed in Section IV. Research related to the work presented in this paper is summarized in Section V. Conclusions derived from this work are in Section VI.

II. DAD: The Method

This section begins with a high level overview of the method followed by a description of how the invariants are generated and coded as monitors. Illustrative examples are derived from the design of an operational water treatment plant [47].

A. Overview

A "process invariant," or simply an *invariant*, is a mathematical relationship among "physical" and "chemical" properties of the process controlled by one or more PLCs. Invariants used in DAD are created executing the tasks listed in Fig. 1. Creation of invariants may begin soon after the Piping and Instrumentation Diagram (P&ID) [5] of the CPS is available, or later when the control algorithms have been designed. In practice, a P&ID is created by design engineers to meet CPS requirements using tools such as AutoCAD. A P&ID depicts physical components, flow paths, instruments, and controllers. Similar diagrams, known as *line-diagram*, or *single line-diagram*, are created for power grid. This paper uses examples derived from P&ID diagrams. A summary of the tasks in Fig. 1 follows.

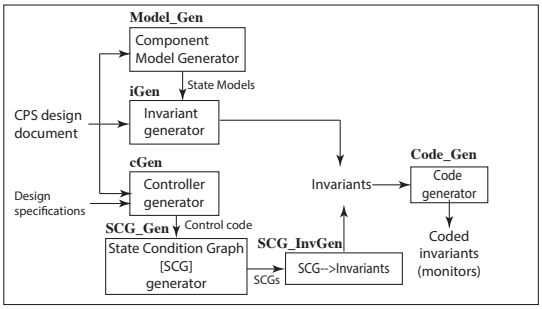

Fig. 1. Security by design: generation of invariants in DAD.

- `iGen`: The Invariant Generator takes the P&ID as input and generates a subset of the invariants.
- `cGen`: The P&ID is also input to a Controller Generator that outputs the control code needed to control CPS behavior for meeting design requirements.
- `SCG_Gen`: The control code is analyzed to generate State Condition Graphs (SCGs) [4]. Each SCG graphically depicts the conditions necessary for a CPS component in one of its states.
- `SCG_InvGen`: An SCG to Invariant converter generates another set of invariants.
- `Model_Gen`: A third set of invariants (not shown in Fig. 1) is generated using physical models of CPS components that exhibit continuous states such as water level in tanks, or pH of water entering or exiting a process stage.

Except for `cGen`, all tasks were executed manually by the authors in the case study in Section III; `cGen` was executed by the designers and builders of SWaT[47] used in the case study. Methods to generate invariants and tasks `iGen`, `SCG_Gen`, `SCG_InvGen`, and `Model_Gen` are described next.

B. Invariants

Let $C = \{c_1, c_2, \ldots, c_n\}$ be a set of n components in a CPS. Let $V = \{V_1, V_2, \ldots V_n\}$, where V_i denotes the state space of c_i, and $v_i(k) \in V_i; 1 \leq i \leq n$, is a state variable denoting the state of c_i at time instant k. A state variable

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2018.2875008, IEEE Transactions on Dependable and Secure Computing

may be discrete or continuous. Components in C include only those whose state is observable via at least one sensor and are controlled by a PLC. Given C and V for a CPS, an invariant is a boolean function on V that remain true during normal operation of the CPS under consideration.

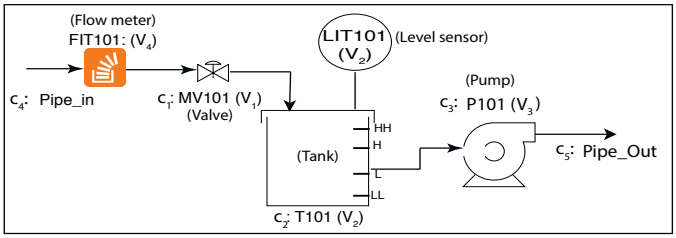

Fig. 2. Components in stage 1 of a water treatment plant [47]. The tank has four level markers labeled HH, H, L, and LL used by a PLC to control the inflow and outflow of water.

Example 1: Consider one stage of a water treatment plant as in Fig. 2. Here, $C = \{c_1 : \texttt{MV101}, c_2 : \texttt{T101}, c_3 = \texttt{P101}, c_4 : \texttt{Pipe\_In}, c_5 : \texttt{Pipe\_Out}\}$. Components c_1 and c_3 are actuators while the remaining are water storage or transfer components. One possible representation of the state space of C is as follows.

V_1={Open, Closed}, $V_2 = \{x \in R^+\}$,
V_3={On, Off}, V_4={No_Flow, Flow}, V_5={No_Flow, Flow}.

where LL (Very Low), L (Low), H (High), and HH (Very Hi) denote level markers on tank T101; Open and Closed are the states of the motorized valve, On and Off are the states of the pump, and No_Flow and Flow are the states of the inflow and outflow pipes indicating, respectively no flow or flow of water through the pipes. Note that the state of water flow in a pipe can also be treated as a continuous variable though for the invariants generated in the case study these are treated as discrete variables that indicate whether water is flowing through a pipe. Transient states of actuators, such as Open→Closed for valve MV101, are considered in Section II-D. In a formal sense, the state space of the motorized valve MV101 and that of pump P101 is finite while that of the tank is infinite. Each level marker corresponds to a specific state of the tank. For example, $800 \leq v_2 < 1200$ implies level is at `H`.

C. Invariant generation

Given the design of a CPS, DAD uses the following methods to generate invariants: *state entanglement*, *component model*, *State Condition Graphs (SCG)*, and *state bounds*. With reference to Fig. 1, generation of invariants using state entanglement and state bounds corresponds to task `iGen`, that using *component model* corresponds to `Model_Gen`, and that using SCGs corresponds to `SCG_Gen`. Methods for generating invariants are described next.

Invariants using state entanglement: Let $c_i, c_j \in C$ be two components in a CPS. The *joint state space* of these two components is given as $V_{i,j} = V_i \times V_j$. However, during plant operation, constraints imposed by laws of nature may result in dependence due to state entanglement. Such dependence leads to an *expected* joint state space of the two components and is denoted as $V^E_{i,j} \subseteq V_{i,j}$. Components c_i and c_j are considered *entangled* if $V^E_{i,j} \subset V_{i,j}$; $V^E_{i,j} = V_{i,j}$ implies no state entanglement in which case the components behave independent of each other. Note that an arbitrary subset of C may be entangled.

Let $V^P_{i,j} = V_{i,j} \setminus V^E_{i,j}$ be the *prohibited* subset of the joint state space of components c_i and c_j. Set $V^P_{i,j}$ leads to a set of $k = |V^P_{i,j}|$ invariants defined as follows. The following invariant is generated for each pair of states $(s_i, s_j) \in V^E_{i,j}$:

if $(v_i == s_i)$ **then**
  $(v_j == s_j)$
end if

However, to generate alerts during plant operation, the above invariant is implemented using state pair $(s'_i, s'_j) \in V^P_{i,j}$.

if $(v_i == s'_i$ **and** $v_j == s'_j)$ **then**
  `genAlert(message)`
end if

Function `genAlert()` sends an alert, with a suitable message, to a CPS process monitor. Note that $V^P_{i,j}$ simplifies the implementation of invariants; doing so only with the state pairs in $V^E_{i,j}$ would be inefficient and require checking the entire set during plant operation.

Example 2: This example illustrates state entanglement among two components in a CPS. Consider $c_1 = \texttt{MV101}$ and $c_4 = \texttt{Pipe\_In}$ in Fig. 2. The state of c_1 can be obtained by sending an appropriate request to it while that of c_4 can be obtained from the flow meter `FIT101`. The joint, expected, and prohibited state spaces of these two components are as follows.

$V_{1,4}$={(Closed, No_Flow), (Closed, Flow)
(Open, No_Flow), (Open, Flow)}

$V^E_{1,4}$={(Closed, No_Flow), (Open, Flow)}

$V^P_{1,4}$=$V_{1,4} \setminus V^E_{1,4}$={ (Closed, Flow), (Open, No_Flow)}

$V^P_{1,4}$ leads to the following two invariants.

if $(v_1 ==$ Closed **and** $v_4 ==$ Flow) **then**
  `genAlert(``State mismatch: MV101 and FIT101")`
end if

if $(v_1 ==$ Open **and** $v_4 ==$ No_Flow) **then**
  `genAlert(``State mismatch: MV101 and FIT101")`
end if ■

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2018.2875008, IEEE Transactions on Dependable and Secure Computing

Invariants using state model: In this approach invariants are generated for components that exhibit continuous states. A water tank and electric battery are examples of such components. The states of interest for a water tank include water level in the tank and its chemical properties such as pH and conductivity. The states of interest in a battery include voltage across terminals and its age. Ideally, a state space model is constructed for each such component in the CPS. However, in practice doing so may be challenging for components that exhibit non-linear behavior such as ultra-filtration or reverse osmosis units. Simplified models are used in such cases. Note that state variables used in the models are those that can be measured directly via sensors or computed through one or more sensor measurements.

A state model is used during plant operation to predict the state of the corresponding component. In addition, the state of this component is sampled from the corresponding sensor(s) and compared with the predicted state. A discrepancy between the predicted and the sampled states leads to an alert. To illustrate, let x denote a state variable to model some component property and y its measurement. Assuming linear relationship, x and y can be related as follows.

$$x(k+1) = ax(k) + bu(k) \tag{1}$$

$$y(k) = cx(k) + du(k) + \eta(k), \tag{2}$$

where $y(k)$ denotes the sensor measurement of state $x(k)$ at time instant t_k, a, b, c and d are constants, $u(k)$ is the control input, and $\eta(k)$ is sensor noise. Let $\hat{x}(k)$ be an estimate of $x(k)$ obtained from a sequence of sensor readings $y(k)$. Known techniques [31] can be used to derive $\hat{x}(k)$ assuming appropriate model of noise $\eta(k)$. In the absence of sensor errors and no attack, $\hat{x}(k) = x(k) = y(k)$. As illustrated next, when a sensor is compromised, $y(k)$ may not retain the expected relationship to the actual state of the component implying that $\hat{x}(k)$ is being controlled by an adversary.

Example 3: Let $x(k)$ denote the water level in tank T101 (Fig. 2) at sampling instant t_k. $x(k)$ is measured by sensor LIT101. Sensors FIT101 and FIT201 (not shown in Fig. 2) measure, respectively, water flow into and out of T101. These flow rates are denoted as $u_i(k)$ for inflow and $u_o(k)$ for outflow. At sampling instant t_{k+1}, the water level in T101 depends on the level, inflow, and outflow at time t_k. This relationship is captured in the following idealized model of the tank assuming perfect sensors,

$$x(k+1) - x(k) = \alpha(u_i(k) - u_o(k)), \tag{3}$$

where α is a constant computed using the physical dimensions of the tank and the flow rates. ■

The invariants are derived using both open and closed loop methods for state estimation. The following equations describe the open loop method.

$$\hat{x}(0) = y(0) \tag{4}$$

$$\hat{x}(k+1) = \hat{x}(k) + \alpha(u_i(k) - u_o(k)) \tag{5}$$

The following equations are used in closed loop estimation.

$$\hat{x}(0) = y(0) \tag{6}$$

$$\hat{x}(k+1) = y(k) + \alpha(u_i(k) - u_o(k)) \tag{7}$$

The key difference in the two methods is in the iteration to estimate $\hat{x}(k)$. In Eq.5, the sensor reading is used only once to initialize $\hat{x}(k)$ whereas in Eq.7 the sensor reading is used at every iteration. In either case, the outcome is the following invariant that must hold,

$$\frac{\sum_{i=1}^{n} |\hat{x}(i) - y(i)|}{n} < \epsilon \tag{8}$$

where n is the number of deviations between the estimated and the observed state to obtain the average difference, and ϵ is the maximum tolerable deviation. Estimates of n and ϵ are obtained either from plant design or, when the plant is operational, by collecting and analyzing multiple observations. For example, the plant process, for which Eq.8 is derived, can be run several times in normal mode (without any attacks) to estimate the mean μ_d and the standard deviation σ_d, where $d = (\hat{x}(k) - y(k))$, i.e. the mean and variance of the difference between the estimated tank level $\hat{x}(k)$ and its measured value $y(k)$. Note that predicting process state and comparing it against measured state to discover process anomaly is also done when using the CUSUM method [8], [49]. Note that Eq. 8 uses the moving average of the difference across successive measurements, while CUSUM uses a statistic that sums up the differences when it is more than a bias parameters computed using the mean of the predicted difference. Details of the CUSUM approach, as applied for attack detection, are in [8]. While Eq. 8 is used in the case study reported, there exist other methods such as, for example, using the squared of the difference between the predicted and the measures values.

Chemical properties: Equations such as Eq. 8 can also be derived for state variables that relate to the chemical properties of water such as pH and conductivity. However, doing so requires estimation of such parameters. The challenge here is due to the nonlinear nature of the UF and RO units that filter the water using various types of membranes. Hence, only the invariants that relate the lower and upper limits of the chemical properties, were used in DAD.

Selection of n and ϵ: Optimal values of n and ϵ (Eq.8) are estimated experimentally. The objective is to reduce the occurrence of false alarms. Stage 1 of SWaT was selected to fine tune n and ϵ. Stage 1 of SWaT contains flow meter FIT101, output flow meter FIT201, level indicator LIT101, and two actuators– valve MV101 and pump P101. Both open and closed loop methods (Eqs. 5 and 7) were used to estimate n and ϵ for LIT101 based on FIT101 and FIT201 though the results presented here are from the open loop method.

Invariants using state condition graphs: Control software in PLCs uses sensor data to compute control actions. Thus, when control software is available, or during its design and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2018.2875008, IEEE Transactions on Dependable and Secure Computing

implementation, one could extract the conditions that lead to a control action to change the state of an actuator. Such conditions are represented in the form of a State Condition Graph (SCG) [4]. Formally, an SCG can be viewed as a tuple (N, E) where N and E are, respectively, sets of labeled nodes and labelled or unlabeled directional edges. A node could be one of the following types: Sensor (S-node), Actuator (A-node), Actuator Condition (AC-node), and boolean connector (B-node). An S-node is labeled with the name of a sensor, an A-node is labeled with the name of an actuator, and a B-node node is labeled with a boolean operator such as AND, OR, XOR, etc. An AC-node is the target of a B-node or an S-node, and is labeled by the name of an actuator and its state. An edge from an S-node is labeled with the state of the component measured by the sensor. An edge from an A-node is labeled with the state of the corresponding actuator. An edge from a B-node is unlabeled and must end at an AC-node. There is one SCG for each state of each actuator. However, for compactness, one SCG may be used to specify conditions for multiple actuators.

Example 4: Consider an SCG for the motorized valve MV101 and pump P101. An SCG for the Closed state of MV101 and Off state of P101 is shown in Fig. 3. In this SCG, $N = S \cup A \cup AC \cup B$, where $S = \{S1, S2\}$, $A = \{A1, A2\}$, $AC = \{AC1, AC2\}$, and $B = \{B1\}$. An SCG for a component in a stage could include conditions on the states of components from another stage of the plant. Thus, for example, the SCG for pump P101 contains conditions on valve MV201 and flow rate sensor FIT201 from stage 2 and tank T301 from stage 3. ■

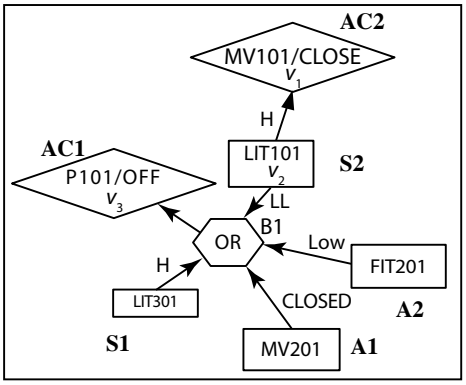

Fig. 3. State Condition Graph for pump P101 Off and valve MV101 Closed.

A condition in an SCG is checked by the control software to decide when and to which actuator should a command be sent. When this condition is true, an appropriate command is sent, and its effect on the actuator validated. This sequence of actions: check condition, send command if condition true, and validate state of the actuator, is part of the control software and activated in each scan cycle of the PLC. Given the possibility of an attack, it is also important to check in each scan cycle, the state of the actuator and the corresponding conditions. It is this latter check that leads to one invariant for each state of each actuator. The next example illustrates such an invariant.

Example 5: In Fig. 3, consider the conditions that must be true for pump P101 to be in the Off state. A PLC responsible to control P101 may use following algorithm to change the state of P101.

if (LIT301$\geq$H **or** MV201==Closed **or** FIT201==Low **or** $v_2 \geq$LL) **then**
  v_3= Off
end if

The above code leads to the following invariant to be checked in each scan cycle of the PLC.

if v_3==On **then**
  if (**not** ($v_2 \geq$LL)) **or** (**not** (MV201==Closed)) **or** (**not** (LIT301$\geq$ H)) **or** (**not** (FIT201==Low)) **then**
    genAlert("Pump P101 in invalid state.")
  end if
end if ■

Other invariants: There are several other approaches to generate invariants In addition to the ones introduced above. These include generation of safety invariants (SI) from plant safety conditions, command control invariants (CCI) , and mass balance invariants (MBI). SI enable checks against lower and upper bounds on state variables. CCI enable checks against the rapid change of the state of actuators such as generators, circuit breakers, pumps, and valves. MBI check if the mass of product entering the plant or a stage of the plant is equal to that leaving the plant over a specified duration. Only the SI were included in DAD in the study described here. Such invariants are included in DAD regardless of whether or not they exist in the control code inside a PLC. SI can easily detect straightforward uncoordinated attacks that aim to damage equipment or move the system to an unsafe state though can be made to remain true through coordinated attacks.

D. Invariant implementation

In the case study reported here, the invariants were coded as monitors in Structured Text [45] and placed inside PLCs (see Section III-B for details). The invariants were also coded in Python and placed in a server that sits on the network on which the PLCs communicate among themselves and with the SCADA workstation. The following example illustrates the challenge in coding an invariant.

Example 6: Consider the SCG for MV101 in Fig. 3. A straightforward implementation of this invariant follows.

if (v_2==H) **and** (**not** (v_1==Closed)) **then**
  genAlert("T101: High, MV101 not closed.")
end if

Consider the case when a PLC completes three scan cycles before MV101 moves from Open to Closed state. If the command to close MV101 is sent in scan cycle k, then alerts

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2018.2875008, IEEE Transactions on Dependable and Secure Computing

TABLE I
INVARIANTS THAT DETECTED ATTACKS IN THE CASE STUDY.

ID	Invariant
P1:I1	MV101=Open $\implies$ FIT101 $> \gamma$
P1:I2	T101$\leq$L $\implies$ MV101 = Open
P1:I3	T101 $\geq$ H $\implies$ MV101= Closed
P1:I4	T101 $\leq$ LL $\implies$ P101=Off $\wedge$P102= Off
P1:I5	LIT301 $\leq$ L $\implies$ P101=On $\wedge$ P102=On
P1:I6	LIT301$\geq$ H $\implies$ P101=Off $\wedge$ P102 =Off
P1:I7	P101=ON $\wedge$ P102=On $\implies$ FIT201$> \gamma$
P1:I8	State check using Eq 8 for T101
P1:I9	Overflow prediction for T101
P2:I5	AIT201$\leq$ 250 $\implies$ P201=On $\vee$ P202= On
P2:I9	AIT202$>$7.05 $\implies$ P203=On $\vee$ P204= On
P2:I11	AIT203$<$420 $\implies$ P205=On $\vee$ P206= On
P2:I13	AIT402$>$ 250 $\implies$ P205=On $\vee$ P206= On
P3:I1	LIT301$\leq$LL $\implies$ P301=Off $\wedge$ P302= Off
P3:I2	P301=On $\implies$ FIT301 $> \gamma$
P3:I3	DPIT301$> p$ $\implies$ P301=Off
P3:I4	LIT401 $\leq$ L $\implies$ P301=On $\vee$ P302= On
P3:I5	LIT401 $\geq$ H $\implies$ P301=Off $\vee$ P302= Off
P3:I6	State check based on Eq 8 for T301

γ is set to a suitable value to indicate that there is minimum flow in the pipe to which an FIT is connected.

will be generated in the immediately following scan cycles. These are likely to be false alerts as MV101 is in the process of moving from the Open to its Closed state. The following alternate implementation avoids false alarms.

if (v_2==H) **then**
  wait(δ)
  if (**not** (v_1==Closed)) **then**
    genAlert("T101: High, MV101 not closed.")
  end if
end if

The above code forces the PLC to wait for δ time units before checking the state of MV101. While the above code gets rid of the false alarms, assuming the wait time is correctly determined, it causes the scan cycle to be extended by δ time units. The above solution was not found practical when several invariants are checked and a wait is needed in each case. Multiple waits increase the scan cycle and delay control actions possibly altering process dynamics. Thus, a third solution, shown below, was adopted that uses counters in each scan cycle and causes little increase in its duration. The above code segment is executed after having initialized variables `LIT_State` and `cycles_elapsed` to 0. Use of the `cycles_elapsed` counter does not require the PLC to wait for the valve to close. The value of τ –threshold for the counter–is selected using the physical specifications of the valve and the duration of each scan cycle. An estimate of the duration of each scan cycle is obtained after all invariants have been coded and added to the control code inside a PLC. ■

```
CASE (LIT_State):
0:
if (v2==H) then
  LIT_State=1
end if
1:
if (cycles_elapsed> τ) then
  if not (v1==Closed) then
    genAlert("T101: High, MV101 not closed.")
    LIT_State=2
  end if
else
  cycles_elapsed=cycles_elapsed+1
end if
if (v2==H) then
  LIT_State=1
end if
2:
LIT_State=0
```

III. CASE STUDY

The effectiveness of DAD was assessed experimentally on a fully autonomous 6-stage water treatment plant named SWaT [47]. Architecture of SWaT, experiments conducted, and the results, are described next.

A. Plant architecture and operation

SWaT produces 5 gallons/minute of filtered water. It mimics a large modern water treatment plant found in cities. As shown in Fig. 4, SWaT consists of six stages labeled 1 through 6. Each stage is controlled by its own set of PLCs labeled PLC1 through PLC6. Details of SWaT components including sensors and actuators, can be found in [47]. The operation of PLC1 is reviewed next while that of the remaining PLCs is available in [47].

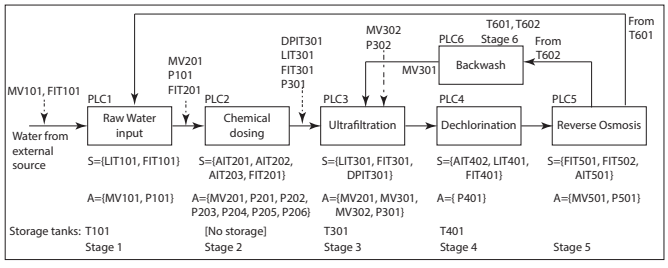

Fig. 4. Six stages in SWaT with corresponding PLCs, sensors, and actuators used by the corresponding level indicator labeled as LITxxx. AITxxx, FITxxx, and DPITxxx measure, respectively, chemical properties of water, flow rate in a pipe, and differential pressure across the ultrafiltration unit. Pxxx denote pumps at various stages.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2018.2875008, IEEE Transactions on Dependable and Secure Computing

PLC1 controls the flow of water into and out of tank T101. As shown in Fig. 2, the tank has four level markers HH, H, L and LL. When water level in T101 is below the L marker, the valve MV101 is opened causing water to flow in. The inflow rate is measured and communicated to PLC1 by flow meter FIT101. When water reaches above the H marker, the valve is closed. Pump P101 is started when water in tank T301 is below the L marker and water in T101 is above the LL marker. P101 is stopped when any one of the following conditions is true: water in tank T301 is at or above the H marker, MV201 is closed, and water level in tank T101 is at or below the LL marker.

B. Invariant generation, coding, and placement

Figure 5 indicates the steps used in the generation, coding, and placement of invariants. Invariants were generated from plant design and control code (Step 1) as explained in Section II-C and coded as monitors (Step 2) that generate alerts when a condition is violated. Monitors are placed inside PLCs (Step 3) as well as on an independent server on the plant network. The decision to place the invariants in the PLCs and also on a server was guided by the fact that some plant data may be compromised without the knowledge of a PLC. A monitor that uses state variables known to only one PLC is placed inside that PLC.

A monitor that uses state variables from within a single stage is placed in the PLC controlling that stage. A monitor that uses state variables from across multiple stages is placed in the corresponding PLCs. A PLC executes the monitors at the start of each scan cycle (Step 4) and transmits any alerts to the SCADA workstation (Step 5) and also generates visual alarms. Monitors in the server are executed at all times and report the alerts on a separate tablet screen attached to each component rack at each stage of SWaT. The results reported here are from the monitors placed inside the PLCs; results from monitors inside the server are not reported here due to space limitations but are available in a separate report [17].

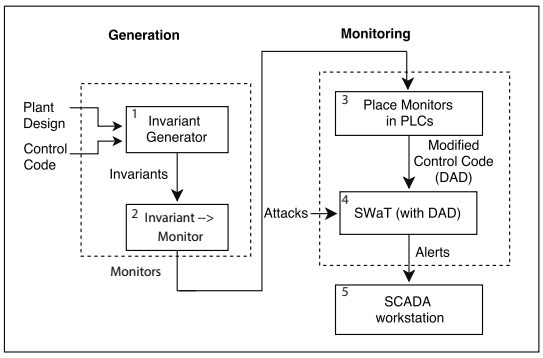

Fig. 5. Steps in the case study: 1. Invariants are generated as explained in Section II-C. 2. The invariants transformed into monitors. 3. The monitors are placed inside the PLCs in SWaT and also on the plant communications network. 4. Attacks are launched on SWaT. 5. Alerts generated are sent to the SCADA workstation.

A total of 53 invariants were generated though only 38 were implemented as monitors. Invariants that relate to the estimation of chemical properties of water, such as pH and conductivity, require more complex models than those in Eqs. 5 or 7, and were ignored in this study. However, invariants based on the correlations between different chemical properties, chemical pumps, level sensors and flow indicators are mentioned. Estimation of chemical properties were difficult due to lack of sensors, and based on our study we observed that it is not possible to estimate at the current state of SWaT. We are able to estimate chemical properties of water after adding few more sensors in different places based on investigation on sensor placement. Invariants that detected attacks in the case study are listed in Table I.

C. Attack design

An CPS-specific adversary model [3] was used to design attacks. It was assumed that an attacker may be physically inside or outside of the plant. Depending on the knowledge and skill set, an attacker may be able to launch a variety of attacks that impact process dynamics. Four types of attacks were designed, namely, static and dynamic attacks that could be launched on single or multiple stages of the plant. Static attacks allow an adversary to tamper with the value of a sensor or send a control command to an actuator. Dynamic attacks are strategic; thus sensor values, or the commands sent to the actuators, are computed during the attack to make the PLC believe that the state of the plant is evolving as expected though in reality it may be diverging to an undesirable state.

An attack scenario for a CPS consists of the following: adversary intent, state of the plant when the attack is launched, plant components compromised, and attack data or control signal sent to a PLC or an actuator. DAD assumes that the adversary has complete knowledge of the plant architecture and thus knows of the physical properties of various components and the communications infrastructure. While important for an adversary and in designing intrusion prevention systems, the exact procedure used by an adversary to intrude into the plant is ignored in the adversary model considered here. Thus, DAD aims to detect attacks launched by adversaries at both extremes: from the weak who have little resources, to the powerful who are rich in resources needed to launch an attack.

Attacks were launched to cause physical damage to components, degrade plant performance measured as the amount of treated water produced per minute, and degrade chemical properties of water such as pH and conductivity. Three types of attacks were launched: Static Single-Point (SS), Static Multi-point (SM), and Dynamic Multi-points= (DM). In a static attack, the attacker does not compute the sensor measurements, instead the measurement is manipulated through a simple addition/subtraction. In a dynamic attack, the sensor value is computed to make the PLC believe that the received measurements are valid in the current system state. In single-point attack, the attacker manipulates exactly one state variable while in a multi-point attack more than one state variables are manipulated. Multi-point attacks could also be coordinated

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2018.2875008, IEEE Transactions on Dependable and Secure Computing

where one or more attackers coordinate their efforts to deceive any attack detection mechanisms. Sample attack scenarios are listed in Table II.

D. Experiments

Evaluation of DAD was carried out using four sets of experiments on the SWaT plant. Exp-A and Exp-S were conducted by the authors while Exp-I by six independent teams. The fourth experiment, Exp-DoS, included one DoS attack that was a replay of the attack launched in Exp-I but was not detected because it was removed due to time constraints. Invariants, listed in Table I, were generated manually using methods described in Section II-C, coded, and installed prior to starting the experiments. Attacks were launched via the PLC communications network or directly from the SCADA workstation that required password authentication.

Exp-A: This experiment involved the design of attacks, bringing SWaT to an appropriate state, launching an attack, and observing its impact. As indicated in Table III (Exp-A) , a total of 20 attacks were launched. An attack was launched either via Level 1, i.e. the plant network, directly from the SCADA workstation. A complete knowledge of the system was assumed. This assumption enabled the design of SS, and more complex SM and DM attacks, to deceive one or more PLCs.

Exp-I: This set of experiments was part of the annual SUTD Security Showdown event [42]. It included four teams from academia and three from companies specializing in cyber security of ICS. Participating teams came from Singapore, The Netherlands, UK, and USA.[1] The competition was divided into two phases: a learning phase followed by a live phase. The learning phase was spread over one month during which participants were given technical details of SWaT architecture, communication protocols and devices used, attack detection mechanisms, and allowed access to a simulated version of the Level 1 network used in SWaT. Publicly available white papers on the attack detection mechanisms were provided to the participating teams. Participants were able to remotely access a training testbed that used the same PLCs and network protocols as SWaT. The live phase was conducted on site in Singapore. Each of the six teams were given a tour of the SWaT plant a day prior to the start of this phase where plant experts answered technical questions. Following the tour each team was given three hours to design and launch attacks. Teams were allowed to launch physical and cyber attacks.

Exp-S: A set of stealthy attacks were launched on SWaT by another group of researchers [48], [49]. These attacks focus on continuous state variables, e.g., water level in a tank, whereas those in Exp-A and Exp-I focus on both continuous and discrete state variables. The attacks revealed that by carefully controlling how the sensor data is spoofed, one could prevent the detection of attack using techniques based on CUSUM or Eq 8. In the work reported here, we explored how DAD will perform under similar stealthy attacks. Thus, two sets of attacks, labeled S1 and S2, were designed as follows.

$$\text{S1} : y'(k) = y(k) \pm \delta; \quad 0 < \delta < \epsilon \tag{9}$$

$$\text{S2} : y'(k) = y(k) \pm \delta * k; \quad k > 1 \tag{10}$$

where $y(k)$ and $y'(k)$ are, respectively, the actual and the spoofed readings from the level sensor in one of the tanks at time instant $k > 0$, $0 < \delta < \epsilon$, and ϵ is as in Eq. 8; δ is the deviation per unit time as in [49]. Attacks of types S1 and S2 simply add (subtract) δ or $k * \delta$, respectively, to (from) each successive sensor reading before sending it to the PLC. In both S1 and S2, the intention of the attacker is to either overflow or underflow the tank. In the case of tank overflow the water simply moves to the drain through the drain pipe. In the case of underflow the tank becomes nearly empty causing the pump, e.g., P101, to run dry if it is ON. All attacks of type S1 and S2 were launched on level sensor LIT101 in stage 1 of SWaT while all other sensors and actuators were operational in their normal conditions, i.e. were not compromised. Thus, only SS single point attacks were considered for S1 and S2.

Exp-DoS: A SYN flood attack was launched on PLC1 when P101 was On and LIT301 was showing a reading between the L and H markers indicating that water is needed in T301 from stage 1. The intent was to disrupt the control of stage 1.

As shown in Table III (Exp-I), 18 attacks were designed and launched on SWaT by the six participating teams. All attacks in this table, except the one marked "SS: Physical," were launched either from the SCADA workstation or Level 1 network. The physical attack was launched directly at the Level 0 network in stage 1 that connects PLC1 to sensors and actuators via a Remote I/O interface unit. Five DoS attacks were launched on the HMI, SCADA workstation and the link between the PLCs and HMI. This network is used by a PLC to communicate with sensors and actuators. The actual attack was launched on stage 1 by physically removing the cable that connects Remote I/O unit to the PLC1. A total of four attacks of type S1 and 13 of type S2 were designed and launched. For attacks of type S2, each of the four combinations of the status of MV101 and P101 was considered.

E. Results

Performance of DAD: In the discussion below, we claim that an attack is *detected* when at least one invariant is violated, and an alert generated, before the attacker is able to realise the intention such as tank overflow. As indicated in Table III, all attacks launched in Exp-A were detected. 11 out of 18 attacks, launched in Exp-I, were detected. All attacks of type S2 were detected. In the following we discuss why some attacks were not detected and the effectiveness of invariants.

DoS attacks: These attacks were not detected because during the period of plant observation immediately following the attack, there was no impact on the process behavior. Thus, while the SCADA workstation and the HMI screens were

[1]Team identities are not included here to maintain privacy.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2018.2875008, IEEE Transactions on Dependable and Secure Computing

TABLE II
SAMPLE ATTACKS ON SWAT.

Intent	**Type**†	**Launch state**	**Components compromised**	**Attack**†	**Comments**
A1: Overflow T101	SS	Any	LIT101 (v_2)	v_2=L	The adversary gains control of the link between the PLC and LIT101; prevents the PLC from receiving data from LIT101; when requested, sends a Low level reading to the PLC; the PLC opens MV101 to let water in eventually causing the tank to overflow.
A2: Overflow T301	SM	v_1=Open v_3=Off v_4=Flow FIT201=No_Flow	MV101 (v_1) P101 (v_3) FIT101 (v_4) FIT201	v_4=No_Flow FIT201=Flow v_1=Closed v_3=On	The adversary gains control of valve MV101, pump P101, and flow meters FIT101, and FIT201 in stages 1 and 2. The state of each of these sensors and actuators is flipped, i.e., from Open to Closed, Flow to No_Flow, and On to Off. This attack may or may not result in the overflow of tank T301 depending on the state of T101 and T301 when the attack is launched.
A3: Damage P101	DM	v_1=Open v_2=L v_3=Any FIT201=Depends on state of v_3	MV101 (v_1) LIT101 (v_2) P101 (v_3)	Set v_1=Closed and maintain this state; compute v_2 and FIT201 assuming that v_1=Open and accounting for the status of P101; send these readings to the PLC when requested.	The PLC believes that T101 is being filled with water; turns P101 On when $v_3 >$L and keeps it On. In reality, v_2 is reducing; this condition will eventually lead to no water in T101 while P101 remains On; unless detected, or there is a mechanical switch to turn P101 Off, P101 will overheat and get damaged.

†In each attack, the adversary sets the state of a component as shown. The attack itself can be launched in a variety of ways such as via the SCADA workstation, HMI, and the communications network.
‡SS: Static Single point; SM: Static Multiple points; DM: Dynamic Multiple points

TABLE III
EFFECTIVENESS OF DAD IN DETECTING ATTACKS

Experiments	**Attack Type**	**Attacks**	**Detected**
Exp-A	SS	10	10
	SM	5	5
	DS	3	5
	DM	2	2
	Total	**20**	**20**
Exp-I	SS	11	9
	SM	1	1
	SS: Physical	1	1
	DoS (HMI)	3	0
	DoS (SCADA)	1	0
	DoS (PLC-HMI)	1	0
	Total	**18**	**11**
Exp-S	S1 (SS)	4	0
	S2 (SS)	13	13
	Total	**17**	**13**
Exp-DoS	DoS (PLC)	1	1
	Total	**1**	**1**

showing incorrect information, or no information in some cases, the process was behaving normally. It is important to note that a DoS attack, when given enough time to evolve and be launched at an appropriate state of the plant, may impact process behavior. When this happens, it is likely that one or more invariant will detect such an attack.

SS attacks: Two SS attacks were not detected. In one attack the adversary altered the status of valve MV301. Under normal circumstances this valve is opened during the backwash process. However, the attacker opened it when there was no backwash. Hence the attack did not affect the physical process except changing the valve status. No invariant was violated due to this attack because the backwash process, i.e., stage 6, is not included in this case study. The second SS attack was performed on chemical dosing pump P203 while the other pump P204 was running. Note that under normal circumstances only one of these two pumps is supposed to be running while the other remains as a backup in case of failure of one pump. Subsequently the attacker shut down pump P204. This attack was not detected because there were no invariants that related to the chemical properties of water.

S1 attacks: No attack of type S1 is detected by any invariant. However, when water level is increasing in tank T101, the PLC closes the motorized valve (MV101) soon after water goes above the H mark in tank T101. Similarly, when the water level is decreasing, the PLC turns the pump (P101) off soon after water level reaches below the L mark. Thus, while such attacks are not detected, there is no damage to the plant. It is important to note that there is a buffer above the H marker (Fig. 2). Given that δ is a small quantity, the control logic in the PLC is able to close the motororized valve and the pump soon after the water level goes above the H or below the LL

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2018.2875008, IEEE Transactions on Dependable and Secure Computing

markers avoiding overflow or underflow. Thus, we consider S1 attacks as stealthy though benign.

S2 attacks: These attacks are detected by invariants P1:I7, P1:I8, and P1:I9. As an example of an S2 attack, consider A1 in Table II. In this case, instead of $v_2 = \text{L}$, the attack was launched with $v2(k) = v_2(k) - \delta * k, \delta = 0.4, k > 0$. Fig. 6 shows the impact of this attack on tank T101. Note that this attack is detected by invariant P1:I9 before the overflow occurs. Detection of such an attack by P1:I8 depends on the value of δ. For values of δ that are much smaller than ϵ, P1:I8 detects the attack after an overflow occurs while for values close to ϵ the attack is detected before the overflow occurs. All underflow attacks are detected by P1:I7 soon after the water level in T101 is below the L mark.

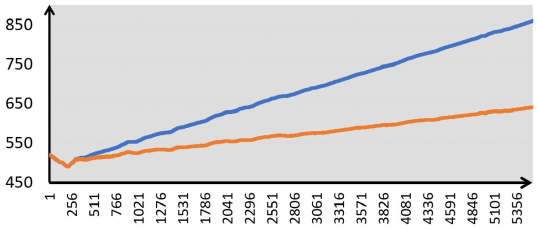

Fig. 6. Impact of stealthy attack on LIT 101. The attack is detected by invariant P1:I9 based on prediction of water level in T101. The horizontal axis indicates successive readings from the sensor (LIT101) that is compromised; the sensor is sampled every 80ms.

Syn-flooding attack: Soon after the attack was launched, portion of the SCADA and HMI screens, that display data from PLC1, turned grey. The CPU utilisation in PLC1 increased to 100% and the PLC was unable to communicate with other PLCs, SCADA workstation, and the HMI. However, pump P101 continued to run. After some time the level sensor LIT301 indicated a high reading implying that tank T301 had enough water and P101 should be turned off. Due to the attack, PLC1 was unable to receive any information from PLC2 and PLC3, and hence P101 was not turned off. However, PLC2 closed MV201 when LIT301 indicated a high water level. Doing so redirected the output of P101 back to tank T101. This is the correct operation of PLC2 as doing so causes water to be redirected to T101 and hence does not cause overflow in T301. However, P101 should be turned off when the water level in T301goes above the H marker. As PLC1 was unable to receive information from PLC3 due to SYN flooding, no invariant in PLC1 was violated. However, the invariant P1:I6, residing at the server on the network, was violated as it was able to receive the T301 level information from PLC3 and already had the information on the state of P101. Thus, the DoS attack was detected. This attack also stresses the importance of placement of the invariants.

Effectiveness of invariants: Table IV lists the number of attacks detected by each invariant. Whether an invariant is violated depends on the attack and its placement, i.e. in the PLC or on the network, or at both locations. Thus, while in this case study, an invariant might be highly effective in detecting attacks, e.g., P1:I2 in Exp-I, it might not be so if the attack scenarios are completely different. Also, note that, for example, P1:I6 did not detect any attacks in both experiments. This is because there was no attack that led to the situation where tank T301 was full and either of the two pumps P101 and P102 was On. This discussion leads to the following question: *Which invariants should be included in a PLC and which ones not?* In an ideal case, when cost is not a constraint and PLC's are infinitely powerful, all invariants ought to be included. However, this is unlikely to be the case in large plants where each PLC may be constrained by the time available for the evaluation of invariants during a scan cycle. In this situation it would be best to include invariants that require only local data for evaluation. P1:I1, P2:I5, and P3:I3 in Table I are examples of such invariants. Such invariants will help detect attacks that are relatively easy to launch, e.g., SS attacks, though will likely miss more difficult to launch, such as the DM, attacks.

Power of distributed attack detection: It is important to understand why the distributed nature of anomaly detection is a powerful mechanism. First, note that the stages in SWaT are highly connected. Thus, an anomaly in stage 1 is likely to be reflected in subsequent stages. This interaction among the interconnected stages leads to one or more monitors in subsequent stages to detect an attack at an earlier stage. Similar arguments could be used to explain how each of the attacks was detected. Section IV contains a discussion on attacks that can and cannot be detected by the invariants generated as described in Section II-C. However, a multi-point attack aimed at causing an overflow in any tank can be used to deceive the corresponding PLC into believing that the water level in the tank is below H. An attack could remain undetected, and cause damage, when all sensors and actuators at a stage are compromised and the attacker replays the dynamics of the stage to the operator. This would happen when neither a preceding nor a successor stage of the stage that is attacked, is affected by the attack. For example, in stage 1 of SWaT, when all sensors, i.e. LIT101, FIT101, FIT102, and actuators MV101 and P101, are compromised, T101 could be made to overflow without being detected by any invariant at stage 1 or any subsequent stages. Such attacks can be detected using mass balance equations which, in the case of SWaT, will check if, over a certain duration, the quantity of water entering T101 is the same as that entering T301.

IV. Discussion

Several issues surfaced during the design, implementation, and evaluation of DAD are discussed next.

Detecting reconnaissance attacks: An adversary might be able to get into a CPS via one of several possible vulnerabilities including those in the SCADA workstation, PLC, or HMI. However, the aim of the intrusion might be to learn system design and its operation. Such an attack cannot be detected

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2018.2875008, IEEE Transactions on Dependable and Secure Computing

TABLE IV
EFFECTIVENESS OF INVARIANTS

Exp-A

Invariant	# of Attacks Detected
P1:I2	2
P1:I3	2
P1:I4	2
P1:I5	1
P1:I8	2
P2:I5	3
P2:I9	2
P2:I11	1
P2:I13	1
P3:I1	1
P3:I4	2
P3:I5	1
P3:I6	2

Exp-I

Invariant	# of Attacks Detected
P1:I2	5
P1:I3	3
P1:I4	2
P1:I5	4
P1:I8	3
P2:I5	1
P2:I9	3
P2:I11	3
P2:I13	4

Exp-S

P1:I7	5
P1:I9	9

Exp-DoS

P1:I6	1

using an approach that relies solely on invariants derived from the physics or chemistry of the CPS. As may be obvious, this is so because unless an adversary manages to fake data to PLC or send undesirable control commands to an actuator, such invariants will continue to hold and hence the attack would remain undetected.

Transient states: Any physical system is likely to include transient states. These states arise in response to a control action that affects an actuator which in turn changes the system state. However, the response time of an actuator is limited by its physical design. Thus, the actuator, as well as its environment, remains in a transient state until the actuator reaches its desired (stable) state. It is possible for an adversary to take advantage of such transient states and launch attacks when the system is moving from one stable to another stable state. This aspect of adversary capability has not been accounted for in the experiments reported here.

Automated derivation of invariants: The invariants were derived manually in the study reported here. Ongoing work aims at deriving the invariants automatically from the CPS design and assessing their strengths using a formal model [23]. A system for complete automation of invariant design would automate each of the steps shown in Fig. 1. Such a system is essential if DAD were to be applied in large plants that contain several thousand sensors and actuators.

State entanglement and combinatorial explosion: Invariants created using state entanglement make it difficult for an adversary to launch attacks that are not detected. Many commercial water treatment plants contain thousands of sensors and actuators. While state entanglement may exist among components as pairs, triples and so on, even using only pairwise entanglement will lead to an exorbitantly large number of invariants. Even if created, coding and embedding the resulting monitors in a PLC, or on the network, may lead to excessive computational load affecting the plant performance. Hence, one proposal to tackle the combinatorial explosion problem in complex plants is to focus on (a) most critical stages of the plant, and (b) pair components that are neighbors and are directly influenced by a connector. For example, in SWaT (see Fig. 2), FIT101 and MV101 are *state neighbors* that are entangled due to a connector *Pipe_In*. However, MV101 and P101 are not state neighbors as their states are not jointly affected by the connectors, e.g., by tank T101 and the joining pipes.

False alarms: The rate at which false alarms are generated is an important attribute of an attack detection method. DAD has been in operation in SWaT since May 2016. No false alarm has been noticed during normal operation of SWaT. Also, during Exp-I, where independent attackers launched attacks on SWaT, we did not observe any false alarm. In addition to the experiments described earlier, experiments were conducted to investigate whether DAD leads to false alarms under certain specific *corner* conditions. To bring the system to a corner state, measurements were manipulated and moved to their extreme range, e.g., near the L(ow) and H(igh) markers (see Fig. 3 for markers). In addition, several other sensors and actuators were manipulated. No false alarm was received in any corner case. Further, all corner cases were detected when the measurements exceeded their preset limits and there were no false alarms.

Why no false alarms?: The source of false alarms in several anomaly detection systems, such as in [8], are the statistical parameters. One could argue that parameters that need to be estimated, such as when using machine learning [15], [34], are the key source of false alarms. DAD does not use any statistical parameters. There are several parameters that are set to suitable values based on component design, e.g., the time to open or close a valve. Other parameters are tuned by observing plant behaviour and performing multiple iterations such as to obtain the values of n and ϵ in Eq. 8. Indeed, false alarms were observed during an early implementation of DAD. Careful examination revealed that these were due to incorrect coding or placement of the invariants. Gradually, as DAD was debugged and the parameters tuned, the number of false alarms reduced to zero.

Generality and limitations of DAD: The ideas underlying detection of cyber attacks in a distributed manner are applicable to plants other than for water treatment. Thus, for example, one may start with a line diagram for a power grid and create invariants. These invariants could then be coded as monitors while accounting for the design parameters. Similar approach has been used by other researchers [33] for power grid though the invariants were not created directly from the line diagram, and only the dynamic model of the process was used. There

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2018.2875008, IEEE Transactions on Dependable and Secure Computing

is a significant difference in the process reaction time in water and power networks consequent to changing the state of an actuator. Thus, one key difference in the application of the approach in DAD to a power grid will be in the selection and tuning of parameters n, ϵ in Eq. 8 and γ used in invariants using bounds on state variables. Certainly, until DAD is applied to realistic plants in domains other than water, one cannot comment on the generality of the approach used.

Completeness of invariants: an open problem?: The completeness of invariants derived using the SCGs depends on the correctness and completeness of the SCGs, and of those derived using state entanglement. In the case of SWaT, a complete and manually verified set of SCGs is available using only pairwise entanglement. The SCGs are amenable to automation and hence one set of invariants can be derived from a complete and accurate set of SCGs. Such derivation could be done at the time of designing a CPS and not necessarily after it has been built as is the case in the case study reported here. There does not appear to be any known criteria to determine the completeness of other invariants derived using state entanglement and process dynamics.

V. Related work

The study reported here focuses on cyber attacks on CPS that result in deliberate sensor and actuator data manipulation. Previous work in this area is summarised in the following domains: attack modelling and analysis, attacks and anomaly detection, and open research challenges.

Open Research Challenges: Several researchers have presented challenges in designing safe and secure CP [7]. Problems in computing and network technologies for full fledged design of emerging CPS are presented in [25] The integration of IoT and SCADA systems with a focus on security and how to integrate make intelligent industrial systems with the help of Internet is explained in. [43]; this work also lays out research directions and gaps in the present community. We believe that DAD is an attempt to meet several challenges laid out in the work cited above.

Attack modeling and analysis: Attack models designed specifically for CPS include deception attacks such as surge, bias, and geometric [8], [52]. Control theoretic models [24] reduce the entire attack space to a mathematically tractable noise and abstract the physical aspects whereas the attacks designed in evaluating DAD are from a cyber physical attacker model [3] and affords an opportunity to widen the attack surface.

Attack detection: The use of invariants for detecting attacks on CPS has been proposed by several researchers. The work that relates most closely to the techniques used in DAD is in [18], [38], [41]. In these works it is claimed that the use of controlled invariant sets in detecting cyber attacks uses little information about the controller and hence is useful for a large range of control laws. The invariant sets are derived using methods in Section II-C. The method of using invariant sets is demonstrated theoretically using a second order unstable system. The experiments reported here derive and use invariants in a realistic setting, and do account for control laws. In [46] the authors generate a bank of input observers to detect and isolate faults in a network. Fabio et al. have proposed distributed state estimation and detection [37] in power networks. The distributed detection method computes a minimum variance estimate of the network state via distributed computation. The work in [37] focuses on power networks and is theoretical in nature whereas that reported here focuses on water treatment system and is experimental. Further, the work reported here accounts for carefully crafted correlations by the adversary across system measurements. Thus, using multi-point attacks, the adversary attempts to thwart the detection mechanism in a controller.

The existing approaches for distributed attack detection differ significantly from the approach proposed in DAD. First, DAD uses a broader class of invariants than the ones generated in existing works that are primarily based on using control theoretic approaches. By using multi-point attacks, e.g., DM attacks in Section III-C, an adversary could deceive a PLC into believing that the system is in a normal state whereas in reality it is not. Second, the approaches cited here have not been tested on realistic plants such as SWaT used in this work or a power network. Third, the adversary model used in this work is powerful and has been used to generate a variety of attacks for the experimental validation of DAD in a manner unlike any other study cited above.

Hsio et al. [20] have proposed a distributed security monitoring solution to detect attacks on a CPS. They describe a coordinated architecture that employs security monitors over the network. Monitors analyze and assess packets for anomalous behavior. Finite state machine models are used to describe subjects at each network level. Pre-conditions are defined for state transitions within and across the machines and used to identify attack symptoms using a statistical approach. While the approach described is similar to DAD, it is different in that it does not use physics-based invariants. Thus, it would be best to consider Hsio et al.'s approach as complementary to that described in this work.

The work of Perelman et al. [39] focuses on chemical or biological contamination in a water distribution system. The approach is to measure water quality parameters, such as pH and conductivity, and use these to distinguish normal from abnormal behavior. While this approach is similar to the one presented here, it does not consider compromised sensors. For example, an adversary could add pollutants and, knowing the normal system behavior, replay all the measurements as if the system is behaving normally. Nevertheless, the approach proposed in [39] seems effective when measurements are taken at multiple stages and the adversary compromises only one stage as in the SM or DM attacks.

Attack resilient state estimation for noisy cyber physical systems has been proposed [35]. In this case the attacker is assumed to be attempting to spoof sensor measurements within the sensor noise bounds and thus perform stealthy attacks.

Attack types: Attacks of type SM in our work are similar to those in [8], [49]. In [49] the authors have proposed a metric to capture the maximum deviation per unit time in sensor readings resulting from undetected attacks, and the expected time between false alarms. This metric is not suitable in the case of multi-point coordinated attacks involving multiple actuators and sensors. Further, attacks in the case study as in Section III were launched on a larger scale than those in [49]. Researchers have also explored false data injection attacks in electric power grids using simulation [27]; these attacks are also single-point; and not coordinated as in our study. The Weaselboard [32] uses PLC backplane to get the sensor, actuator values and analyses them to prevent zero day vulnerabilities.

ECFI [1] provides protection against run-time attacks targeting PLC control programs but does not protect against data-only attacks, or maliciously modified/replaced control programs or firmware. Orpheus [12] monitors the behavior of a devices control program based on the invoked system calls. Attacks are detected based on a finite-state machine representing the control program's benign system call behavior. Orpheus behavior monitor is placed inside the OS of the device, hence a compromised OS can disable and circumvent its protection mechanism. SOCCA [55] presented contingency analysis in cyber physical systems which uses network level attack graphs based on Markov decision processes. CPAC [16] presents stateful detection mechanisms to detect attacks against control systems. Note that the method used in DAD is immune to zero day vulnerabilities. While any such vulnerability can be explored by an attacker, its impact would likely be the creation of process anomaly which would be detected by one or more monitors.

Attack detection using machine learning: Yuqi et. al. [11] proposed an approach for learning physical invariants that combines machine learning with ideas from mutation testing. Initial models are learned using support vector machines. These learned models are used for code attestation and identifying standard network attacks. Configuration based intrusion detection system have also been proposed for Advanced Metering Infrastructure [40]. The AMI behavior is modeled using event logs collected at smart meters. Event logs are modeled using the Markov chains and linear temporal logic for the verification of specifications. However, such models depend on the completeness of the training data set used for the learned models. Sequence aware intrusion detection [9] is presented to detect the semantic attacks. Sequence attack is one of the semantic attacks that models how a series of permitted operations can elude intrusion detection and damage the physical process.

Failure mode and effects analysis: Failure mode, vulnerabilities and effects analysis (FMVEA) [44] and Combined Harm Assessment of Safety and Security for Information Systems (CHASIS) focuses on the automotive domain. It is a safety and security analysis of cyber physical systems. A risk assessment framework for automotive embedded systems [22] was presented. It is introduced in alignment with ISO 26262.

Other related work: Data validity of cyber physical systems through path redundancy has been studied [54]. The proposed approach is to prevent data integrity attacks and detect false command attacks from a single compromised path or PLC. Function based methodology to evaluate the resilience of gas pipeline systems under two different attack scenarios is presented in [50]. The authors analyzed the cyber attacks that propagate from cyber space to a gas pipeline. Granular data flow management [19] has been used to achieve ICS resilience and security. Using models of physical attacks [10] in CPS, the authors analyze the security of CPS using a meta-modeling approach. They consider the effects of physical attacks. Distributed runtime monitor [51] for ICS/SCADA was proposed to detect violations of safety properties. Characterizing disruptive events to model cascade failures in critical infrastructures is presented in [6].

VI. Conclusions

Distributed Attack Detection (DAD) for detecting cyber-physical attacks on a CPS is described. The method has been implemented in a water treatment plant and its effectiveness assessed using a number of attacks launched by the authors and by independent teams. DAD uses invariants, derived from process dynamics and state entanglement among the physical components, to detect both cyber and physical attacks. The effectiveness of DAD depends on three key factors: (a) completeness of invariants, (b) availability of sensors, and (c) placement of invariants in the plant. Experiments revealed that manual generation of invariants is likely to lead to an incomplete set of invariants including some that are erroneous. Thus, automated generation of invariants is essential for DAD to be applicable in any large legacy plants. However, DAD can be used effectively during the design of new plants where control engineers create the invariants while they design the control algorithms. Invariants so created could be placed in the PLCs and also on the plant communications network.

References

[1] A. Abbasi, T. Holz, E. Zambon, and S. Etalle. ECFI: Asynchronous control flow integrity for programmable logic controllers. In *Proceedings of the 33rd ACSAC*, pages 437–448, 2017.

[2] M. Abrams and J. Weiss. Malicious control system cyber security attack case study–Maroochy Water Services, Australia. *McLean, VA: The MITRE Corporation*, 2008.

[3] S. Adepu and A. Mathur. Generalized attacker and attack models for cyber-physical systems. *The 40th IEEE COMPSAC*, 2016.

[4] S. Adepu and A. Mathur. Introducing cyber security at the design stage of public infrastructures: A procedure and case study. In *Proceedings of the 2nd CSD&M Asia*, 2016.

[5] G. Bogdanovskà and M. Pavlcovà. Hazard and operability study. In *2016 17th ICCC*, pages 82–85, 2016.

[6] E. Canzani, H. Kaufmann, and U. Lechner. Characterising disruptive events to model cascade failures in critical infrastructures. In *Proceedings of the 4th ICS-CSR*, pages 1–8, 2016.

[7] A. Cardenas, S. Amin, and S. Sastry. Secure control: Towards survivable Cyber-Physical Systems. In *28th ICDCS*, pages 495 –500, 2008.

[8] A. A. Cárdenas and et al. Attacks against process control systems: Risk assessment, detection, and response. In *Proceedings of the 6th ASIACCS*, pages 355–366, 2011.

[9] M. Caselli, E. Zambon, and F. Kargl. Sequence-aware intrusion detection in industrial control systems. In *Proceedings of the 1st ACM Workshop on Cyber-Physical System Security*, CPSS '15, pages 13–24, 2015.

[10] C. Cheh, K. Keefe, B. Feddersen, B. Chen, W. G. Temple, and W. H. Sanders. Developing models for physical attacks in cyber-physical systems. In *Proceedings of the CPS-SPC*, pages 49–55, 2017.

[11] 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. IEEE S&P*, 2018.

[12] L. Cheng, K. Tian, and D. D. Yao. Orpheus: Enforcing cyber-physical execution semantics to defend against data-oriented attacks. In *Proceedings of the 33rd ACSAC*, pages 315–326, 2017.

[13] G. Dán and H. Sandberg. Stealth attacks and protection schemes for state estimators in power systems. In *First IEEE SmartGridComm*, pages 214–219, Oct 2010.

[14] ICS-CERT Advisories https://ics-cert.us-cert.gov/advisories.

[15] M. Esmalifalak, L. Liu, N. Nguyen, R. Zheng, and Z. Han. Detecting stealthy false data injection using machine learning in smart grid. *IEEE Systems Journal*, PP(99):1–9, 2014.

[16] S. Etigowni, D. J. Tian, G. Hernandez, S. Zonouz, and K. Butler. Cpac: securing critical infrastructure with cyber-physical access control. In *Proceedings of the 32nd ACSAC*, pages 139–152, 2016.

[17] F. FURTADO, L. GOH, S. RAJAGOPAL, E. CHEON, E. THIANG, T. J. Hui, and I. LEE. Swat security showdown (s3-17) event report. Technical report, iTrust, Singapore University of Technology and Design, 2017.

[18] T. Gamage, B. McMillin, and T. Roth. Enforcing information flow security properties in cyber-physical systems: A generalized framework based on compensation. In *IEEE 34th Annual COMPSACW*, pages 158 –163, 2010.

[19] B. Green, M. Krotofil, and D. Hutchison. Achieving ICS resilience and security through granular data flow management. In *Proceedings of the 2nd ACM CPS-SPC*, pages 93–101, 2016.

[20] S.-W. Hsiao, Y. Sun, M. C. Chen, and H. Zhang. Cross-level behavioral analysis for robust early intrusion detection. In *IEEE International Conference on Intelligence and Security Informatics (ISI)*, pages 95–100, May 2010.

[21] https://ics-cert.us-cert.gov/.

[22] M. M. Islam, A. Lautenbach, C. Sandberg, and T. Olovsson. A risk assessment framework for automotive embedded systems. In *Proceedings of the 2nd ACM CPSS*, pages 3–14, 2016.

[23] E. Kang, S. Adepu, D. Jackson, and A. P. Mathur. Model-based security analysis of a water treatment system. In *In Proceedings of 2nd SEsCPS*, 2016.

[24] C. Kwon, W. Liu, and I. Hwang. Security analysis for cyber-physical systems against stealthy deception attacks. In *American Control Conference (ACC), 2013*, pages 3344–3349, 2013.

[25] E. A. Lee. CPS: Design challenges, http://www.eecs.berkeley.edu/Pubs/TechRpts/2008/EECS-2008-8.html. Technical report, 2008.

[26] R. Lipovsky. New wave of cyber attacks against Ukrainian power industry, January 2016. http://www.welivesecurity.com/2016/01/11.

[27] Y. Liu, P. Ning, and M. K. Reiter. False data injection attacks against state estimation in electric power grids. *ACM TISSEC*, 14(1):13, 2011.

[28] A. Matrosov, E. Rodionov, D. Harley, and J. Malcho. Stuxnet under the microscope. *ESET LLC*, September 2010.

[29] B. mesnil and P. Petitgas. Detection of changes in time-series of indicators using cusum control charts. *Aquatic Living Resources*, 22:187–192, 2009.

[30] Y. Mo, E. Garone, A. Casavola, and B. Sinopoli. False data injection attacks against state estimation in wireless sensor networks. In *Decision and Control (CDC), 49th IEEE Conference on*, pages 5967–5972, 2010.

[31] Y. Mo and B. Sinopoli. A characterization of the critical value for Kalman filtering with intermittent observations. In *47th IEEE Conference on CDC 2008.*, pages 2692–2697, 2008.

[32] J. Mulder, M. Schwartz, M. Berg, J. R. Van Houten, J. Mario, M. A. K. Urrea, A. A. Clements, and J. Jacob. Weaseboard: Zero-day exploit detection for Programmable Logic Controllers. Technical report, tech. report SAND2013-8274, Sandia National Laboratories, 2013.

[33] H. Nishino and H. Ishii. Distributed detection of cyber attacks and faults for power systems. In *Proceedings of the 19th World Conference, The International Federation of Automatic Control Congress, Cape Town, South Africa*, pages 11932–11937, August 2014.

[34] M. Ozay, I. Esnaola, F. Yarman Vural, S. Kulkarni, and H. Poor. Machine learning methods for attack detection in the smart grid. *IEEE Transactions on Neural Networks and Learning Systems*, PP(99):1–14, 2015.

[35] M. Pajic, I. Lee, and G. J. Pappas. Attack-resilient state estimation for noisy dynamical systems. *IEEE Trans on Control of Network Systems*, pages 82–92, 2017.

[36] M. Pajic, J. Weimer, N. Bezzo, P. Tabuada, O. Sokolsky, I. Lee, and G. Pappas. Robustness of attack-resilient state estimators. In *ACM/IEEE ICCPS*, pages 163–174, April 2014.

[37] F. Pasqualetti, R. Carli, and F. Bullo. A distributed method for state estimation and false data detection in power networks. In *IEEE Conference on Smart Grid Comms*, pages 469–474, 2011.

[38] T. Paul, J. Kimball, M. Zawodniok, T. Roth, and B. McMillin. Invariants as a unified knowledge model for cyber-physical systems. In *IEEE International Conference on Service-Oriented Computing and Applications (SOCA)*, pages 1–8, Dec 2011.

[39] L. Perelman, J. Arad, N. Oliker, A. Ostfeld, and M. Housh. Water distribution systems event detection. In *Complexity in Engineering (COMPENG), 2012*, pages 1–3, June 2012.

[40] A. M. Qasim and A.-S. Ehab. Configuration-based ids for advanced metering infrastructure. In *Proceedings of the ACM CCS*, 2013.

[41] A. Rosich, H. Voos, and M. Darouach. Cyber-attack detection based on controlled invariant sets. In *ECC*, pages 2176–2181, June 2014.

[42] S^3-2016: SWaT Security Showdown (S^3). https://itrust.sutd.edu.sg/scy-phy-systems-week/2016/s3/.

[43] A. Sajid, H. Abbas, and K. Saleem. Cloud-assisted iot-based scada systems security: A review of the state of the art and future challenges. *IEEE Access*, 4:1375–1384, 2016.

[44] C. Schmittner, Z. Ma, E. Schoitsch, and T. Gruber. A case study of fmvea and chassis as safety and security co-analysis method for automotive cyber-physical systems. In *Procd. 1st ACM CPSS '15*, 2015.

[45] A. Scott. *Instant PLC Programming with RSLogix 5000*. 2013.

[46] I. Shames, A. Teixeira, H. Sandberg, and K. Johansson. Distributed fault detection for interconnected second-order systems. *Automatica*, (2011).

[47] SWaT:, 2015. https://itrust.sutd.edu.sg/wp-content/uploads/sites/3/2015/11/Brief-Introduction-to-SWaT_181115.pdf.

[48] D. Urbina, J. Giraldo, N. O. Tippenhauer, and A. Cardenas. Attacking fieldbus communications in ICS: Applications to the SWaT testbed. In *Singapore Cyber-Security Conference (SG-CRC)*, pages 75–89, 2016.

[49] D. I. Urbina and et al. Limiting the impact of stealthy attacks on industrial control systems. In *Proceedings of the 2016 ACM CCS*, pages 1092–1105, 2016.

[50] Y. Wadhawan and C. Neuman. Evaluating resilience of gas pipeline systems under cyber-physical attacks: A function-based methodology. In *Proc. of the 2nd ACM CPS-SPC '16*, pages 71–80, 2016.

[51] A. Wain, S. Reiff-Marganiec, H. Janicke, and K. Jones. Towards a distributed runtime monitor for ics/scada systems. In *Proceedings of the 4th ICS-CSR '16*, 2016.

[52] S. Weerakkody, Y. Mo, and B. Sinopoli. Detecting integrity attacks on control systems using robust physical watermarking. In *IEEE 53rd Annual CDC*, pages 3757–3764, 2014.

[53] L. Xie, D.-H. Choi, S. Kar, and H. Poor. Fully distributed state estimation for wide-area monitoring systems. *Smart Grid, IEEE Transactions on*, 3(3):1154–1169, Sept 2012.

[54] Z. Zheng and A. N. Reddy. Towards improving data validity of cyber-physical systems through path redundancy. In *Proceedings of the 3rd ACM Workshop on CPSS '17*, pages 91–102, 2017.

[55] S. Zonouz and et al. Socca: A security-oriented cyber-physical contingency analysis in power infrastructures. *IEEE Transactions on Smart Grid*, pages 3–13, 2014.

BIOGRAPHY

Sridhar Adepu is a PhD student in ISTD pillar at the Singapore University of Technology and Design. His work focuses on security of cyber-physical systems.

Aditya Mathur is a Professor of Computer Science at Purdue University and Head of Pillar Information Systems Technology and Design at the Singapore University of Technology and Design. His research focuses on design of secure critical infrastructure.