Detecting PLC Intrusions Using Control Invariants

Zeyu Yang, Liang He, *Senior Member, IEEE*, Hua Yu, Chengcheng Zhao, *Member, IEEE*, Peng Cheng, *Member, IEEE*, and Jiming Chen, *Fellow, IEEE*

***Abstract*—Programmable logic controllers (PLCs), i.e., the core of control systems, are well-known to be vulnerable to a variety of cyber attacks. To mitigate this issue, we design `PLC-Sleuth`, a novel noninvasive intrusion detection/localization system for PLCs, which is built on a set of control invariants—i.e., the correlations between sensor readings and the concomitantly triggered PLC commands—that exist pervasively in all control systems. Specifically, taking the system's supervisory control and data acquisition log as input, `PLC-Sleuth` abstracts/identifies the system's control invariants as a control graph using data-driven structure learning, and then monitors the weights of graph edges to detect anomalies thereof, which is in turn, a sign of intrusion. We have implemented and evaluated `PLC-Sleuth` using both a platform of ethanol distillation system (EDS) and a realistically simulated Tennessee Eastman (TE) process. The results show that `PLC-Sleuth` can: 1) identify control invariants with 100%/98.11% accuracy for EDS/TE; 2) detect PLC intrusions with 98.33%/0.85‰ true/false positives (TPs/FPs) for EDS and 100%/0% TP/FP for TE; and 3) localize intrusions with 93.22%/96.76% accuracy for EDS/TE.**

***Index Terms*—Control invariants, intrusion detection system (IDS), programmable logic controllers (PLCs).**

I. Introduction

PROGRAMMABLE logic controllers (PLCs) are widely used in industrial control systems (ICSs) [2]. By reading the sensor readings and issuing the concomitant control commands, PLCs operate ICSs according to the specified control rules.

However, the PLCs are well-known to be vulnerable to cyber attacks [3], [4]. Even worse, the advancement of Industrial IoTs and increasingly networked ICSs have made the attacks against PLCs much easier to launch [5], [6]. Researchers have identified more than 36.7k PLCs can be directly accessed from the Internet [7]. Symantec has also confirmed the intrusions of ICSs network [8], from which the targeted PLCs can be accessed. At the same time, as most of the commercial PLC protocols usually support only weak authentication [9]–[11], adversaries are allowed to administer the destructive payloads (e.g., malicious code and forged commands) to PLCs to manipulate the embedded control rules [12], [13]. Moreover, both the developed autonomous PLC binaries de-compilers [14], [15] and the proposed fuzzing framework [16] facilitate the weaponization of PLC vulnerabilities [17]. Once compromised the PLC, adversaries can mount a variety of attacks are included in the following.

1) *Command Injection Attacks:* The malware TRITON was launched remotely in Saudi Arabia in 2017 [18] to disturb the operations of safety actuators in a petrochemical facility. These actuators were originally designed to take remedial actions in case of emergency, while TRITON instead sent them adversarial commands to shut down the facilities.
2) *Cooperative Stealthy Attacks:* To hide attacks from being detected, adversaries can even mount advanced stealthy attacks to PLCs, by not only tampering with control commands, but also cooperatively forging the supervisory control and data acquisition (SCADA) logs with historical (and normal) data. An example of this stealthy attack is the Stuxnet which damaged hundreds of centrifuges secretly [19]. A firmware vulnerability of Allen Bradley PLCs exposed in 2017 [20] also allows modifying the PLCs' control commands and forging sensor readings.

To protect PLCs against the above attacks, we design a noninvasive data-driven intrusion detection system (IDS) for PLCs, called `PLC-Sleuth`. Specifically, `PLC-Sleuth` detects illegitimate control commands using a set of *control invariants* that exist pervasively in all control systems, i.e., the control commands issued by PLCs correlate strongly with the concomitant sensor readings. Taking the system SCADA logs as inputs, `PLC-Sleuth` automatically identifies the control invariants and abstracts them as a control graph, in which the nodes represent system variables (i.e., system setpoints, control commands, and sensor readings) and the weights of edges quantify the correlations between system variables. `PLC-Sleuth` then describes the legality of control command as the weights of graph edges using data-driven approaches, and detects, at runtime, the illegalities—edges with abnormal weights indicate the control channels between corresponding system variables have been compromised.

We have implemented and evaluated `PLC-Sleuth` using our ICS platform of ethanol distillation system (EDS)

Manuscript received December 24, 2021; revised February 23, 2022; accepted March 27, 2022. Date of publication April 4, 2022; date of current version June 7, 2022. This work was supported in part by the Science and Technology Innovation 2030 Program under Grant 2018AAA0101605; in part by the National Natural Science Foundation of China under Grant 61833015 and Grant 61903328; in part by the Zhejiang Provincial Natural Science Foundation under Grant LZ22F030010; and in part by the Institutional Grant at UC Denver. This article was presented in part at the 23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020). *(Corresponding author: Peng Cheng.)*

Zeyu Yang, Hua Yu, Chengcheng Zhao, Peng Cheng, and Jiming Chen are with the State Key Laboratory of Industrial Control Technology, Zhejiang University, Hangzhou 310027, China (e-mail: zeyuyang@zju.edu.cn; hua.yu@zju.edu.cn; chengchengzhao@zju.edu.cn; pcheng@iipc.zju.edu.cn; jmchen@ieee.org).

Liang He is with the Department of Computer Science and Engineering, University of Colorado, Denver, CO 80204 USA (e-mail: liang.he@ucdenver.edu).

Digital Object Identifier 10.1109/JIOT.2022.3164723

and a representative/realistic TE chemical process. We first corroborate the real-time capability of PLC-Sleuth by comparing the time needed for PLC-Sleuth to acquire data (i.e., an averaged 0.0016 s) and detect attacks (i.e., an averaged 0.015 s), with the logging period of SCADA (i.e., 0.5 s by default). We then evaluate PLC-Sleuth's identification of control invariants, and the intrusion detection/localization against the command injection attacks and the cooperative stealthy attacks. The results show that for EDS and TE, respectively, PLC-Sleuth identifies system invariants with {100%, 98.11%} accuracy, detects PLC attacks—even those change the normal commands by only 0.12%—with {98.33%/0.85‰, 100%/0%} true/false positives (TPs/FPs), and localizes compromised control loops with {93.22%, 96.76%} accuracy.

In summary, PLC-Sleuth has the following properties.

1) *Cyber–Physical IDS:* PLC-Sleuth is built on a set of physically induced invariants of control systems, i.e., a given actuation is always triggered by specific real-time sensor measurements, which can be observed as the correlations between the issued control commands and the concomitant sensor readings. This physically induced control invariant makes PLC-Sleuth *reliable* as the correlation remains unchanged unless the control rules are updated, and *pervasively deployable* because such control invariants exist in all control systems.
2) *Noninvasive IDS:* Taking the SCADA logs of an ICS as input, PLC-Sleuth automatically identifies the control invariants, and uses them to protect the PLCs thereof, without perturbing the PLC, i.e., PLC-Sleuth is noninvasive and thus easy to deploy.
3) *IDS Localizing Intrusions:* PLC-Sleuth, besides detecting intrusions at PLCs, also localizes the compromised control loops thereof, facilitating swift repair/forensics of the PLCs; the control system otherwise remains unreliable no matter how well the intrusions are detected.

II. Preliminaries and Basic Idea

Below we present the necessary background of ICSs and the basic idea of PLC-Sleuth, using our platform of EDSas an example. The distillation process separates constituents in a liquid mixture based on the differences in their volatility. EDS is a scaled-down but fully operational distillation plant, which purifies the alcohol from water and is capable of producing alcohol with a purity of 90%.

A. EDS Platform

A typical control system is made up of four main components, organized as feedback control loop(s).

1) A *physical process* to achieve the specific mission, e.g., a distillation tower to purify the alcohol.
2) A set of *sensors* to measure the real-time states y of process, e.g., a liquid level sensor to measure the amount of liquid material in the distillation tower.
3) A set of *controllers* to issue control commands u based on the difference between sensor reading y and the expected setpoint s, i.e., the control error $x \triangleq s - y$.

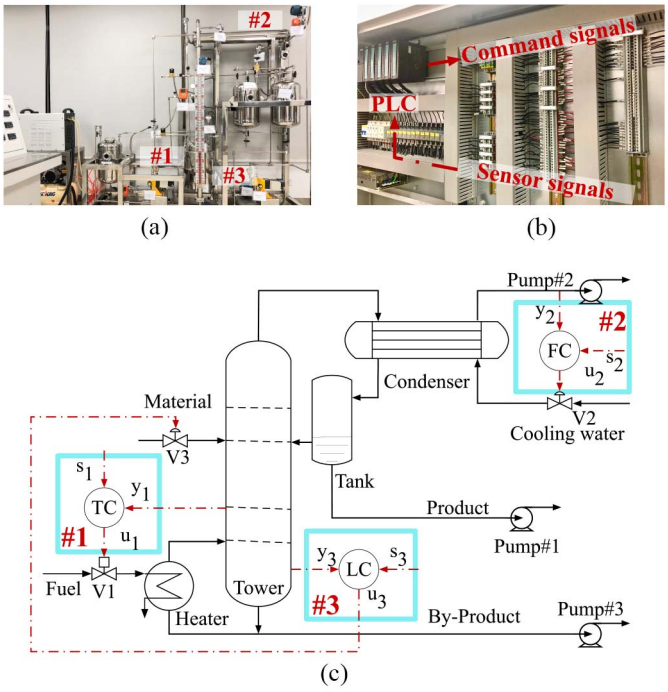


Fig. 1. Platform of EDSin our lab. (a) Testbed. (b) Control network. (c) Process flow diagram.

4) A set of *actuators* to regulate the state of process according to the issued control commands u, e.g., close the inlet valve to decrease the tower liquid level.

Fig. 1(a) shows our platform of EDS, implemented using feedback control loops consisting of one SIMATIC S7-300 PLC, and 11 sensors, and three actuators, as shown in Fig. 1(b). These components together form three feedback loops to achieve the operations of EDS, as shown in Fig. 1(c).

1) *Loop 1* regulates the tower temperature to a predefined setpoint s_1, at which EDS achieves a high separation efficiency. The temperature controller (TC) generates the command u_1 based on the difference between s_1 and temperature y_1 collected using a temperature sensor (i.e., $x_1 = s_1 - y_1$), to control the opening of valve V_1 (and hence the heat) to regulate the tower temperature.
2) *Loop 2* maintains the cooling water flow y_2 of the condenser—measured by a liquid flow sensor—around a predefined setpoint s_2 to condense ethanol from gas to liquid, which is then refluxed to the tower to improve the distillation concentration further. Both excessive or deficient ethanol refluxing impede the heat/mass transfer and degrade the distillation quality. Also, an unstable flow of cooling water may physically damage the pump. The flow controller (FC) issues the command u_2 to the valve V_2 to regulate the cooling water based on $x_2 = s_2 - y_2$.
3) *Loop 3* controls the liquid level y_3 of the distillation tower to an optimized setpoint s_3, which is also affected by the ethanol reflux of Loop 2. An unstable liquid level induces premature flood and thus, degrades the distillation efficiency [22]. Similar to the TC and FC, the level controller (LC) generates the command u_3 based on $x_3 = s_3 - y_3$ and sends it to the valve V_3 to regulate the

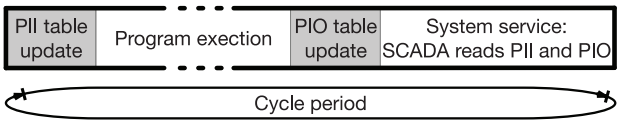


Fig. 2. Sequence of CPU executions in PLC [21].

input of materials and thus the liquid level of distillation tower.

The specific control rules of these loops are embedded in the S7-300 PLC of EDS, using the Bang-Bang and proportional-integral-derivative (PID) algorithms, which generate the control signals u based on the current and historical control errors x [23], [24]. In short, EDS maintains system states $\{y_1, y_2, y_3\}$ around setpoints $\{s_1, s_2, s_3\}$ using control commands $\{u_1, u_2, u_3\}$. These interactions among sensors, setpoints, and control commands form the basis of all control systems. At the same time, together with other sensor readings of $\{y_4, \ldots, y_{11}\}$ which are used to achieve a thorough monitoring of the physical process, operations of the above control loops are monitored and logged by a WinCC SCADA system.

B. Attack Model

We consider the following attack model targeting at the PLC of a given control system.

1) The attacker can mount the command injection attack during the program execution period (Fig. 2) by downloading malicious code to PLC.
2) Atop the injection attack, the attacker can deliver cooperative stealthy attacks by further replaying normal sensor readings to the PLC's process-image input (PII) table. Note that SCADA reads sensor readings from the PII table, which happens after the execution of malicious code (Fig. 2). This way, the attacker can deceive the SCADA to conclude a normal operation even if an attack has been launched [19].
3) The attacker cannot modify the records of commands issued during program operation, i.e., any forged commands issued by the attacker will be logged as they are. This is because process-image output (PIO) table logs all commands and remains unchanged after the execution of the program [25]. By reading from the PIO table (Fig. 2), SCADA obtains control commands that are actually issued.

We next use a cooperative stealthy attack mounted at EDS to motivate `PLC-Sleuth`, in which adversaries use forged control commands damage EDS and faked sensor logs to deceive the system monitor to conclude a normal system operation. Specifically, let us consider a case that an attacker aims to degrade the distillation quality by hacking Loop 2 of EDS to destabilize the reflux. Without attack, the increase of command u_2 is proportional to x_2, i.e., a flow difference of $x_2 = (s_2 - y_2)$L/min drives the valve as

$$\Delta u_2 = -\underline{0.01}(\Delta x_2 + 0.17 \times x_2). \tag{1}$$

By exploiting PLC's vulnerabilities, e.g., modifying the s7otbxdx.dll file of the Step 7 programming software [19], the attacker downloads the destructive payload to the PLC and manipulates the generation of u_2 as

$$\Delta u_2^a = -\underline{0.1}(\Delta x_2 + 0.17 \times x_2) \tag{2}$$

thus, degrading the alcohol gas condensing and causing oscillations in ethanol reflux. Note that the unstable ethanol reflux will destabilize the tower level y_3, and degrade the distillation quality further. To hide the abnormal deviations of $\{y_2, y_3\}$, the attacker replaces the real sensor logs with the historical steady records, by overwriting the PLC's PII table. This way, the sensor logs in the SCADA appear normal, whereas the distillation quality of EDS is actually degraded.

C. Basic Idea of `PLC-Sleuth`

As explained above, PLCs generate the control command u based on the control error $x = s - y$. This induces strong and reliable correlations between u and x, which we use as the PLC's *control invariants*. Fig. 3 visualizes such a control invariant between u_2 and x_2: when EDS's PLC is compromised as described above, the control invariant [shown in Fig. 3(a)] changes noticeably due to the manipulated control rule of (2), under both the command injection attack [shown in Fig. 3(b)] and the cooperative stealthy attack [shown in Fig. 3(c)]. `PLC-Sleuth` uses these control invariants to detect PLC intrusions, by detecting deviations of these control invariants from their expected behaviors. The challenge is, in turn, to identify/abstract the system's control invariants, which `PLC-Sleuth` addresses using a control graph.

III. Abstracting Control Invariants

For a given control system, `PLC-Sleuth` identifies and abstracts its control invariants—defined by the system variables and the interactions thereof—using a *control graph* $\mathcal{G}(\mathtt{V}, \mathtt{E}, \mathtt{W})$, where: 1) $\mathtt{V}$ is the set of nodes representing system variables (i.e., setpoints, sensor readings, and commands); 2) $\mathtt{E}$ is the set of directed edges connecting nodes in $\mathtt{V}$, representing the correlations among system variables; and 3) $\mathtt{W}$ is the set of edge weights, capturing the strength of the correlations described by $\mathtt{E}$. Specifically

1) $\mathtt{V} = \{S, Y, U\}$ consists of a setpoint node set S, a sensor node set Y, and a command node set U, which can be obtained and identified automatically from the SCADA logs. Note that in control systems, the number of nodes in U is no less than that in S, i.e., $|U| \geq |S|$;
2) $\mathtt{E} = \{E_s^y, E_y^u\}$ consists of a control error edge set E_s^y connecting nodes in S to nodes in Y, and a control command edge set E_y^u connecting nodes in Y to nodes in U. An edge $e_{s_i}^{y_j} \in E_s^y$ or $e_{y_j}^{u_k} \in E_y^u$ exists when its two nodes belong to the same control loop. The commonly used the decentralized control scheme—one sensor is fed back for the generation of one control command, and vise versa [26]—makes both the out-degree of nodes in Y and the in-degree of nodes in U equal to 1;
3) $\mathtt{W} = \{W_s^y, W_y^u\}$ consists of an error weight set W_s^y and a command weight set W_y^u, capturing the difference of node pairs $\{s_i, y_j\}$s in E_s^y and the correlation of node pairs $\{y_j, u_k\}$s in E_y^u, respectively. `PLC-Sleuth` exploits W_y^u

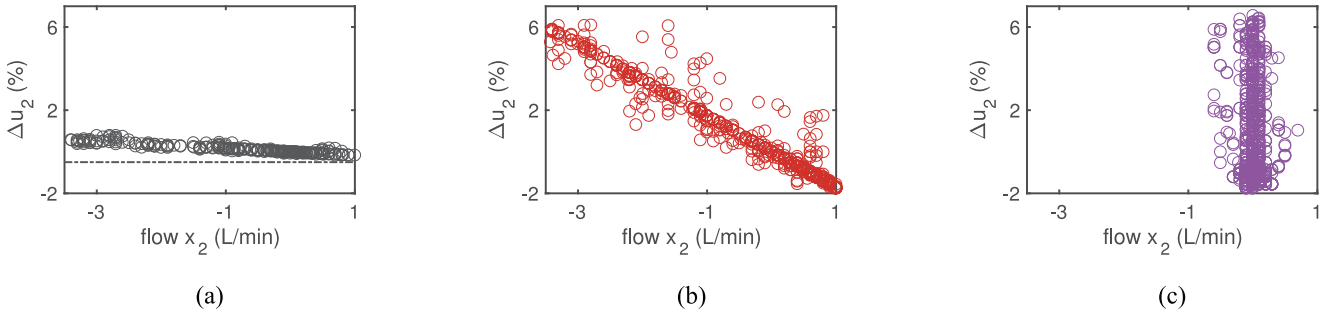


Fig. 3. Correlations between EDS's command u_2 and flow difference x_2. (a) Normal operation without attack. (b) With command injection attack. (c) With cooperative stealthy attack.

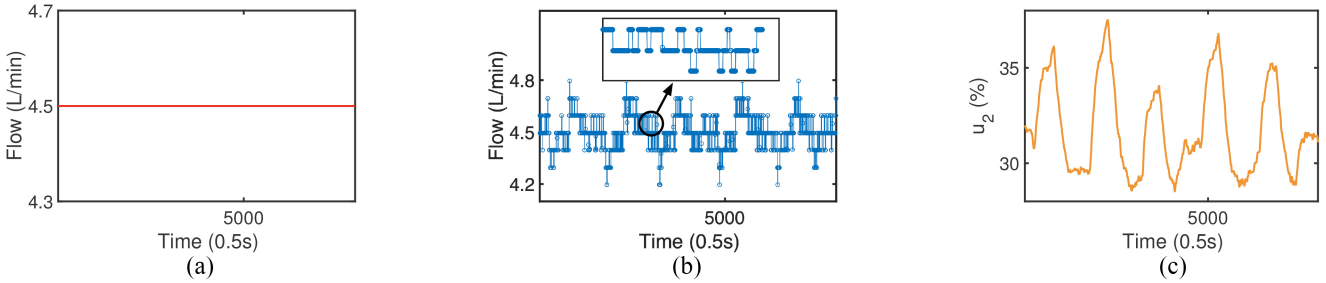


Fig. 4. Different patterns of setpoint, sensor, and command variables in EDS's Loop 2. (a) Log of setpoint s_2. (b) Log of sensor y_2. (c) Log of command u_2.

to detect and localize PLC intrusions, by monitoring W_y^u in real time and detecting the anomalies thereof. Note that the error weight in W_s^y could also be used to detect PLC attacks, i.e., the sensor reading in Y should be close to its corresponding setpoint in S in a stable ICS [27]. This approach, however, is vulnerable to replayed sensor logs in cooperative stealthy attacks [28], as we will corroborate in Section V-C.

Setpoint Node Set S: A setpoint node $s_i \in S$ denotes an expected state of system, which will not fluctuate when the system is operating, as shown in Fig. 4(a).

Sensor Node Set Y: A sensor node $y_j \in Y$ captures the system's real-time state. Because of the environment noise and the sensor measurement error, the readings of y_j show small but consistent vibrations, as observed in Fig. 4(b).

Command Node Set U: A command node $u_k \in U$ reflects the actuation to achieve the predefined system state and fluctuates with the readings of the corresponding sensor. The issued command u_k will be smooth to protect the actuator, as shown in Fig. 4(c).

Error Weight Set W_s^y: The weight of control error edge $w_{s_i}^{y_j} \in W_s^y$ captures the difference between system state y_j and the corresponding setpoint s_i over the recent l samples

$$w_{s_i}^{(y_1,\ldots,y_d)} = \left| \sum_{t=1}^{l} (s_i(t) - f(y_1(t), \ldots, y_d(t))) \right| \tag{3}$$

where d is the number of sensors used to estimate system states using function f, which can be obtained from control algorithms' expressions or binary files [15].

Command Weight Set W_y^u: Mutual information (MI)—a metric commonly used to characterize dependency between variables using their posterior probability distribution [29]—is an intuitive metric to define command weight $w_{y_j}^{u_k} \in W_y^u$. Specifically, the weight of command edge can be defined as the normalized MI, i.e.,

$$w_{y_j}^{u_k} = \frac{I(x_i, u_k)}{H(u_k)} \tag{4}$$

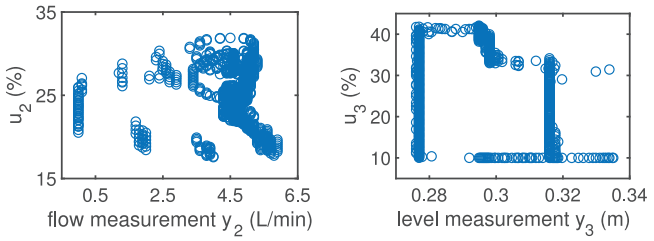
where x_i is the control error between s_i and y_j, $H(u_k)$ is the entropy of the recent sequence of commands u_k, and $I(x_i, u_k)$ denotes the MI between x_i and u_k, which is calculated using the joint probability density function of variable x_i and u_k [denoted as $p(\xi, \eta)$] as

$$I(x_i, u_k) = \sum_{\xi \in \mathcal{X}} \sum_{\eta \in \mathcal{U}} p(\xi, \eta) \log\left(\frac{p(\xi, \eta)}{p(\xi)p(\eta)} \right). \tag{5}$$

However, because the variable's posterior probability only depicts static information, MI has limited ability to quantify the dynamic correlation between sensor measurement y and command u. Fig. 5 shows the scatter plot of EDS's sensor readings $\{y_2, y_3\}$ and control commands $\{u_2, u_3\}$, showing that they have a low correlation when only considering samples collected at a given time instant, i.e., the weighting scheme in (4) may not work well for `PLC-Sleuth`.

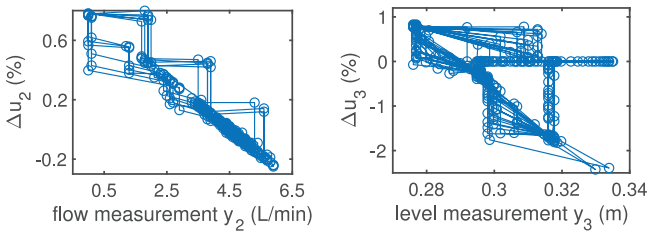
To mitigate the above limitation, we define a novel transition correlation, using the correlation between the two time series of Δu and y, as visualized in Fig. 6 where the close-to-diagonal path indicates a much stronger correlation (when compared to Fig. 5).

Specifically, inspired by the fact that the command u_k is triggered according to the current and historical values of its corresponding error series $x_i = s_i - y_i$, we define the transition



(a) (b)

Fig. 5. Static correlations [calculated according to (4)] between EDS's commands and sensor readings. (a) Cooling water loop. (b) Tower liquid level loop.



(a) (b)

Fig. 6. Transition correlations [calculated according to (7)] between EDS's commands and sensor readings. (a) Cooling water loop. (b) Tower liquid level loop.

MI (TMI) as

$$\mathrm{TMI}(x_i, \Delta u_k) = \sum_{\xi \in \mathcal{X}^\tau} \sum_{\eta \in \mathcal{U}^1} p(\xi, \eta) \log\left(\frac{p(\xi, \eta)}{p(\xi)p(\eta)}\right) \quad (6)$$

where τ is the length of sequences used for characterizing variables' transitions. This way, PLC-Sleuth calculates the command weight $w_{y_j}^{u_k}$ as

$$w_{y_j}^{u_k} = \frac{\mathrm{TMI}(x_i, \Delta u_k)}{H(\Delta u_k)}. \quad (7)$$

The weighting of MI and TMI are compared statistically in Table I in Section V.

Control Graph Example: As an example, Fig. 7 shows the control graph $\mathcal{G}(\mathrm{V}_{\mathrm{SEDS}}, \mathrm{E}_{\mathrm{SEDS}}, \mathrm{W}_{\mathrm{SEDS}})$ of EDS, where

$$V_{\mathrm{SEDS}} = \{\{s_1, s_2, s_3\}, \{y_1, y_2, y_3\}, \{u_1, u_2, u_3\}\}$$
$$E_{\mathrm{SEDS}} = \left\{\left\{e_{s_1}^{y_1}, e_{s_2}^{y_2}, e_{s_3}^{y_3}\right\}, \left\{e_{y_1}^{u_1}, e_{y_2}^{u_2}, e_{y_3}^{u_3}\right\}\right\}$$
$$W_{\mathrm{SEDS}} = \left\{\left\{w_{s_1}^{y_1}, w_{s_2}^{y_2}, w_{s_3}^{y_3}\right\}, \left\{w_{y_1}^{u_1}, w_{y_2}^{u_2}, w_{y_3}^{u_3}\right\}\right\}.$$

Note that, again, both the out-degree of $\{y_1, y_2, y_3\}$ and the in-degree of $\{u_1, u_2, u_3\}$ equal to 1. The sensor variables $\{y_4, \ldots, y_{11}\}$, which are used for system monitoring only, are not shown in Fig. 7 for clarity.

IV. Design of PLC-Sleuth

Fig. 8 presents the logic flow of PLC-Sleuth: constructing the control graph $\mathcal{G}(\mathrm{V}, \mathrm{E}, \mathrm{W})$ of the system-of-interest by identifying the variable's connections (i.e., edges in E) and the corresponding weights in W, and monitoring each weight of w_y^u, in real time, to detect and localize PLC attacks.

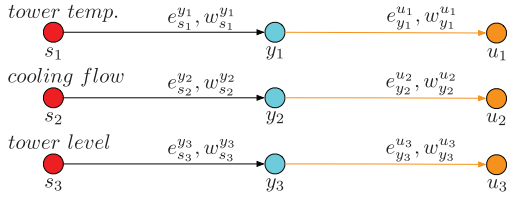


Fig. 7. Control graph of EDS.

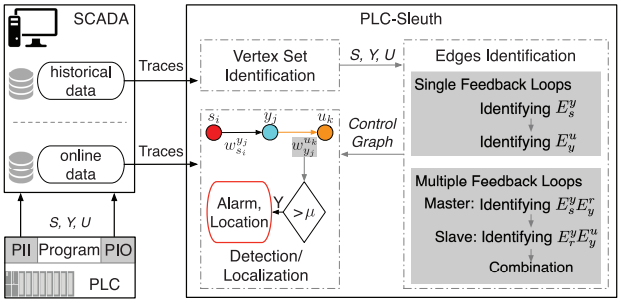


Fig. 8. Architecture of PLC-Sleuth.

A. Construction of Control Graph

An intuitive way to construct the control graph is to have system designers manually extract it from system documents (e.g., engineering flow diagrams, loop diagrams, logic diagrams, electrical control diagrams, etc.). However, this requires significant human efforts or field expertise and is error prone [30]. As an alternative, PLC-Sleuth constructs the control graph $\mathcal{G}$ automatically with a data-driven approach, using the historical (and normal) SCADA logs. Note that the fact that $\mathcal{G}$ consists of two types of edges differ it from the traditional structure learning [31], [32].

Identifying Vertex Set: PLC-Sleuth identifies the nodes of $\mathcal{G}$ automatically using the SCADA logs: 1) setpoint variables are of constant values; 2) sensor readings are of consistent and small vibrations; and 3) control commands are of continuous/smooth values, as shown in Fig. 4. PLC-Sleuth first identifies the setpoint node set S by finding the constant variables (i.e., $H(s_i) = 0$). Then by calculating the vibrations-signal ratio (VSR) of variable y_j, PLC-Sleuth identifies sensor node set Y from the variables with consistent fluctuation. Here, we define vibrations as the curve crests/troughs $y_j(t)$ in y_j, where $y_j'(t) \times y_j'(t+1) < 0$, and VSR_{y_j} as the maximum ratio of crests/troughs in a period samples of y_j. With a ten persistent samples setting, all sensor nodes in EDS have the VSR of 1, as shown in Fig. 9. After selecting nodes with high VSR (e.g., > 0.5 in Fig. 9) as sensor node set, the unclassified variables are classified as the command node set U.

Identifying Edge Set for PLCs With Single Feedback Loops: On top of the identified nodes, PLC-Sleuth has to identify the edges and determine their weights. For the ease of description, let us first consider the construction of control graphs for PLCs consisting of loops with only a single feedback channel, e.g., the three control loops in EDS. We will later extend the graph construction to more complex control loops involving multiple (and likely coupled) feedback channels.

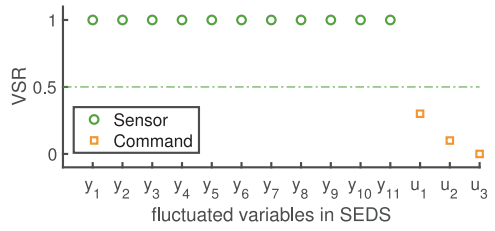


Fig. 9. VSR of fluctuated variables in EDS.

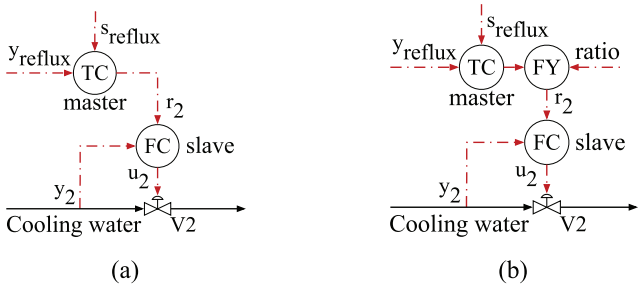


Fig. 10. Examples of coupled feedback control schemes. (a) Cascade control loop. (b) Ratio control loop.

Step I (Identifying E_s^y Edges): PLC-Sleuth first identifies the error edges in E_s^y, which allows the identification of command edges in E_y^u later. It is desirable for control systems to operate at a steady state that leads to a high system efficiency, which is achieved in practice by using a set of setpoints for each feedback loop of the PLC. Hence, we expect the sensor reading y to be close to its corresponding setpoint s, which steers PLC-Sleuth's identification of edges in E_s^y. The basic idea is that, for each node $\tilde{s}_i \in S$, PLC-Sleuth identifies its corresponding edge in E_s^y by finding the node $\tilde{y}_j \in Y$, such that the weight $w_{\tilde{s}_i}^{\tilde{y}_j}$ ([calculated with (3)] is the smallest among all the $\{\tilde{s}_i,\ y_j\}$ pairs. Taking EDS as an example, $\{s_1, s_2, s_3\}$ are set as {51 °C, 4.5 L/min, 0.29 m}, and the sensor readings $\{y_1, y_2, y_3\}$ fluctuate around {51, 4.5, 0.29}. As a result, the weights of $\{w_{s_1}^{y_1}, w_{s_2}^{y_2}, w_{s_3}^{y_3}\}$ tend to be small positive values.

After identifying the edge connecting y_j to s_i, PLC-Sleuth obtains the corresponding time series of control error $x_i(t) = s_i(t) - y_j(t)$, which are used to identify the command edges in E_y^u, as we explain next.

Step II (Identifying E_y^u Edges): For each of the above identified error edges $e_{s_i}^{y_j} \in E_s^y$, PLC-Sleuth further matches its sensor node y_j to the command node u_k of the same control loop. Typically, a control algorithm regulates an actuator's action based on sensor measurements to minimize the control error, making a command node u_k closely correlated with the sensor reading y_j of the same control loop. We use (7) to quantify the correlation between nodes y_j and u_k, based on the identified E_s^y. Similar to the identification of edges in E_s^y, PLC-Sleuth matches a command node $\tilde{u}_k$ to the sensor node $\tilde{y}_j$ of an identified measurement edge $e_{\tilde{s}_i}^{\tilde{y}_j}$, by finding the maximal $w_{\tilde{y}_j}^{\tilde{u}_k}$ [calculated with (7)] in all the $\{\tilde{y}_j,\ u_k\}$ pairs. This way, PLC-Sleuth matches each sensor node $\tilde{y}_j$ of the identified edge in E_s^y with the most correlated command node $\tilde{u}_k \in U$.

Step III (Rematching of Repeated E_y^u Edges): A control command u operates an actuator in the same control loop. However, the operated actuator not only affects the control loop to which it belongs, but may also affect other control loops due to the physical interactions. For example, the command u_3 in EDS does not only affect the tower liquid level y_3, but also the tower temperature y_1, because more intake of cold materials will cool the tower more. As a result, both y_1 and y_3 are likely to be matched with u_3. In general, if a control system has two or more closely coupled components, it may happen that a command node u_k^* is matched to multiple sensor nodes y_js of different $e_{s_i}^{y_j}$s edges, thus violating that the in-degree of a command node should be 1. We call the command node u_k^* being matched to multiple y_js the *repeated command node*. For a repeated command node u_k^*, PLC-Sleuth deletes its edges from $\mathcal{G}$ by keeping only the edge with the maximal weight. An example of the repeated node is u_3 of EDS, which could be potentially matched to both y_1 and y_3. PLC-Sleuth eventually removes $e_{y_1}^{u_3}$ because $w_{y_1}^{u_3} < w_{y_3}^{u_3}$.

After addressing all repeated command nodes, PLC-Sleuth repeats steps II and III for the remaining unmatched y_j and u_k, until all edges in E_s^y, or their corresponding sensor reading nodes y_js more specifically, are matched with a command node u_k.

Identifying Edge Set for PLCs With Multiple Feedback Loops: Many real-world control systems use control loops with coupled feedback channels—such as cascade control and ratio control [26], [33]—to improve their efficiency. These coupled loops usually operate in a master-slave manner, as illustrated in Fig. 10. Taking EDS as an example, to stabilize the reflux temperature at a setpoint level, we can deploy another TC to EDS's Loop 2 to adjust FC's setpoint, making the FC a slave controller of TC.

Transfer node $r_m \in R$, where $R \subset U$, is introduced to construct the control graph with coupled feedback channels, which is responsible for transferring control commands from master to slave loops (e.g., r_2 in Fig. 10). Note that most transfer nodes also act as setpoints of the slave loops. We refer to transfer nodes that do not act as the setpoints of slave loops as transmit nodes. PLC-Sleuth identify R from U. Specifically, each control loop of a PLC generates a corresponding command, all of which form U. Note that only a subset of $\hat{U} \subseteq U$ will be sent to actuators in the coupled feedback control,[1] and the remaining commands $U \backslash \hat{U}$ form the transfer node set R.

In the following, we explain how PLC-Sleuth constructs the control graph for feedback loops with multiple channels.

Step IV (Constructing the Master Loops): Master loops are those containing a setpoint node in S. We construct a control graph for the master loops from node set $\{S, Y, R\}$. The identification of error edges in E_s^y of master loops is done in the same way as step I. Also, the command edges in E_y^r in master loops are identified similarly as in steps II and III, by replacing node $u_k \in U$ with node $r_m \in R$.

Step-V (Constructing the Slave Loops): The control graph for slave loops is constructed from node set $\{R,\ Y,\ \hat{U}\}$. Error edges in E_r^y of slave loops are identified similarly to

[1]Actuators will send received commands back to SCADA [34], [35].

step I, by replacing (3) with $w_{r_m}^{(y_1,\ldots,y_d)} = \min\{|\sum_{t=1}^{L}(r_m - f(y_1,\ldots,y_d))|, |\sum_{t=1}^{L}(r_m * r_n - f(y_1,\ldots,y_d))|\}$. This step identifies the sensor node $\tilde{y}_j$ (and $\tilde{r}_n$ as well if the ratio control scheme exists) for the transfer node $\tilde{r}_m$. Transfer nodes that are not matched with a sensor node are treated as transmit nodes. Based on the identified edges from set R to set Y, command edges in E_y^u of slave loops are identified with the same approach as in steps II and III.

Step VI (Combining the Control Loops): Master and slave loops are combined using transfer nodes in R. When a sensor node $\tilde{y}_j$ in the master loop is matched to a transmit node $\hat{r}_n$, which transfers the control command but does not act as a slave setpoint, PLC-Sleuth further matches $\hat{r}_n$ with a slave setpoint node $\tilde{r_m}$ with the maximal $w_{\hat{r}_n}^{\tilde{r}_m}$ [calculated with (7)] in all the $\{\hat{r}_n,\ r_m\}$ pairs. For the command edge $e_{\tilde{y}_j}^{\tilde{r}_m}$ of master loop not constructed by the transmit node, the sensor node $\tilde{y}_j$ is matched directly to the setpoint node $\tilde{r_m}$ of the slave loop. This process terminates when all transfer node r_m of the command edge $e_{y_j}^{r_m}$ in master loop is matched with a slave loop. Then, by comparing the weights of $w_{\tilde{y}_j}^{\tilde{u}_k}$ and $w_{\tilde{y}_j}^{\tilde{r}_m}$ (or $w_{\tilde{y}_j}^{\hat{r}_n}$), PLC-Sleuth determines the specific control graph (i.e., single feedback loops or multiple feedback loops) to which the nodes $\tilde{s}_i$, $\tilde{y}_j$, $\tilde{r}_m$, $\hat{r}_n$, and $\tilde{u}_k$ belong.

B. Detection/Localization of PLC Intrusions

PLC-Sleuth then detects/localizes, at runtime, PLC intrusions using the constructed control graph $\mathcal{G}(\mathtt{V}, \mathtt{E}, \mathtt{W})$.

As illustrated in Section II-C, tampering the control command will violate the control invariants between commands and the corresponding sensor readings. The violations of control invariants will be observed as the deviations of weights in $\mathcal{G}$. PLC-Sleuth uses a sliding window T to construct an online detector that monitors the weights of edges in E_y^u, by

$$w_{y_j}^{u_k}(t) = \frac{\mathrm{TMI}(x_i([t-T,t]),\ \Delta u_k([t-T,t]))}{H(\Delta u_k([t-T,t]))} \tag{8}$$

where $x_i([t-T,t])$ corresponds to control errors of $y_j(t)$ in window T and $u_k([t-T,t])$ denotes the PLC's control commands in the same window.

Intrusion Detection: PLC-Sleuth uses a memory-based method, such as the nonparametric CUmulative SUM (CUSUM) [36], to alarm operators if an anomaly is detected in (8). CUSUM is defined recursively as

$$S_0 = 0 \text{ and } S_t = \max(0,\ S_{t-1} + |v_{t-1}| - \delta) \tag{9}$$

where v_t is the weight error from the expected value $\hat{w}_{y_j}^{u_k}(t)$, defined as

$$v_t = w_{y_j}^{u_k}(t) - \hat{w}_{y_j}^{u_k}(t) \tag{10}$$

and δ is a small positive constant, set as $\delta > |v_{t-1}|$ when the system operates normally, preventing S_t from increasing persistently. An alarm is triggered whenever S_t is larger than a predefined threshold, i.e.,

$$S_t > \mu \tag{11}$$

at which time the detection will be reset with $S_t = 0$.

Intrusion Localization: It is trivial to localize the forged command if only one abnormal edge is detected, i.e., the command responsible for that edge is compromised. When multiple anomalies are detected, PLC-Sleuth localizes the forged command as the one triggering the anomaly alarm first. This greedy strategy is effective because cascaded anomalies require a longer time to cause instability to control systems, when compared to the directly forged command. This can be well-justified using EDS as an example: the forged u_2^a in (2) first degrades the stability of the ethanol reflux in Loop 2, causing alarms at edge $e_{y_2}^{u_2}$, and then further oscillates the liquid level y_3 in Loop 3 through fluid transportation, which may induce alarms at edge $e_{y_3}^{u_3}$ after the first alarm is triggered. This way, PLC-Sleuth localizes the forged command as the one controlling Loop 2.

V. Implementation and Evaluation

We have implemented and evaluated PLC-Sleuth as a software module of EDS's SCADA, which is deployed on a computer with Intel Celeron Processor G3930 (2.90 GHz) and 4-GB RAM. PLC-Sleuth acquires the control commands and concomitant sensor readings from the SCADA runtime database (e.g., the OLE DB of Siemens WinCC [37]) using OPC protocol [38]. We have also evaluated PLC-Sleuth using a simulated Tennessee Eastman (TE) process, which is a representative benchmark of continuous chemical process [39].

A. Methodology

Our evaluation of PLC-Sleuth consists of two parts. We first evaluate PLC-Sleuth's construction of the control graph by comparing it with the cases when defining $w_{y_j}^{u_k} \in W_y^u$ using four other metrics proposed in structure learning: 1) the Bayesian scoring metric K2 [40]; 2) log-likelihood (LL) [41]; 3) minimum description length (MDL) [42]; and 4) MI in (4).

Using the thus-identified control graph, we then evaluate PLC-Sleuth's intrusion detection/localization of attacks causing different deviations (from the legitimate commands) under both two scenarios.

Attack 1: A control command injection attack implemented by injecting the malicious control command $u^a(t)$ to the PIO table and tampering with the control algorithm parameters.

Attack 2: A cooperative stealthy attack atop of Attack 1 by replaying the historical normal sensor measurements $y^a(t)$ during attacks.

The above attacks are empirically mounted by exploiting the vulnerabilities of PLC programming software (i.e., implementations of abilities 1 and 2 described in Section II-B). For example, with the modified "s7otbxdx.dll" of Siemens step 7, the motivating attack in Section II-B—i.e., destabilizing the ethanol reflux—is mounted by 1) concatenating the logic of "amplifying the Δu_2 by ten times" at the end of legitimate PLC code and 2) concatenating the logic of "assigning y_2 with stored normal records" at the start of legitimate PLC code.

B. Evaluation With EDS

We first implement and evaluate PLC-Sleuth on our EDS platform.

TABLE I
WEIGHT OF EDS'S COMMAND EDGES OBTAINED WITH DIFFERENT WEIGHING SCHEMES

Metric		K2			LL			MDL			MI			PLC-Sleuth		
		u_1	u_2	u_3	u_1	u_2	u_3	u_1	u_2	u_3	u_1	u_2	u_3	u_1	u_2	u_3
Score	y_1	38918	37101	**145**	**-31778**	-27892	-6405	5183	**13628**	82	**0.04**	0.02	0.03	**0.75**	0.08	0.12
	y_2	39006	**42114**	37	-31253	**-20940**	-6484	**-4256**	10942	-100	0.06	**0.27**	0.02	0.14	**0.85**	0.40
	y_3	**41192**	37648	5373	-24187	-24796	**-787**	-12723	-9470	**5420**	0.27	0.13	**0.88**	0.50	0.51	**0.97**

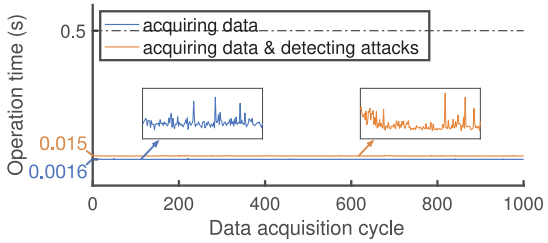


Fig. 11. Operation time of PLC-Sleuth to complete each data acquisition.

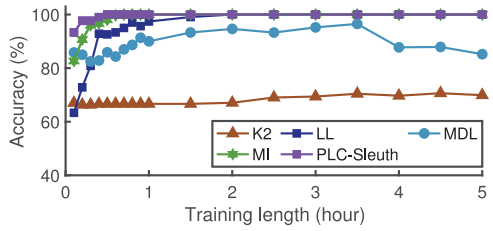


Fig. 12. Constructing PLC-Sleuth's control graph when using different metrics to define $w_{y_j}^{u_k}$ in (7).

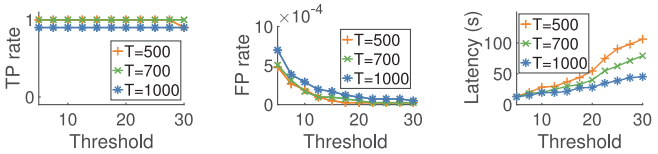


Fig. 13. Applying PLC-Sleuth to EDS with varying detection window T and threshold μ. (a) TP rate. (b) FP rate. (c) Latency.

Real-Time Capability: Once installed, PLC-Sleuth will periodically acquire system states from the database of SCADA. As shown in Fig. 11, the averaged operation time of 0.0016 s to acquire the data is much shorter than the SCADA logging period (the minimum default is 0.5 s [43]). After constructing the control graph, PLC-Sleuth detects attacks by verifying each edge's weight. The averaged operation time of 0.015 s $\ll$ 0.5 s to acquire the data and then detect attacks in Fig. 11 also corroborates the real-time capability of PLC-Sleuth.

Constructing EDS's Control Graph: We next examine if PLC-Sleuth is able to accurately construct EDS's control graph. Note that EDS has three decoupled single feedback loops as shown in Fig. 7.

Data Collection: We logged a 9.8-h operation of EDS using its SCADA system, during which a total number of 17 variables are collected at 2 Hz, including three setpoint variables (i.e., $|S| = 3$), 11 sensor variables (i.e., $|Y| = 11$), and three command variables (i.e., $|U| = 3$). We then use these logs to evaluate PLC-Sleuth's construction of the control graph.

Accuracy of Graph Construction: With a training data of 2 h, Table I compares the weights of $e_{y_j}^{u_k}$s (i.e., $w_{y_j}^{u_k}$s) of EDS obtained with PLC-Sleuth and when using K2, LL, MDL, and MI as the weighting metric, showing.

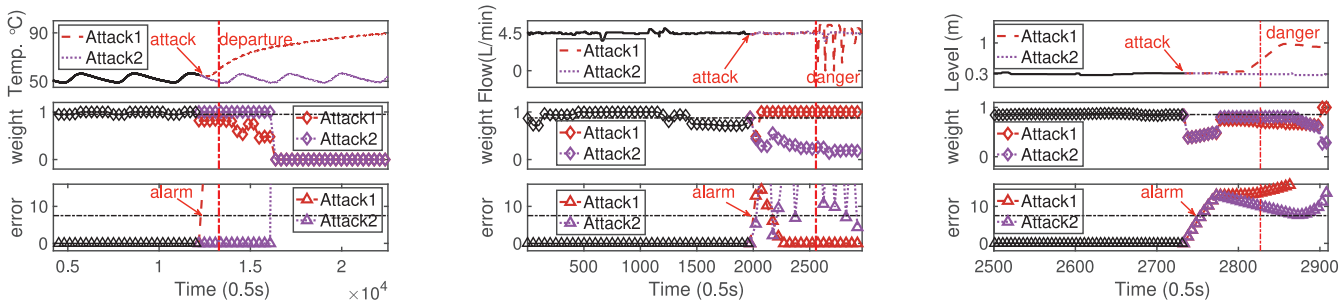
1) PLC-Sleuth identifies EDS's command edges e_y^us accurately and without repeated edges.
2) K2 and MDL fail to identify the command edges correctly, due to the falsely matched edges of $\{e_{y_1}^{u_3}, e_{y_3}^{u_1}\}$ and $\{e_{y_1}^{u_2}, e_{y_2}^{u_1}\}$, respectively.
3) Although LL and MI match sensor nodes to corresponding command nodes successfully, they cannot accurately characterize the correlation strength between nodes u_1 and y_1, as evident by their results of $w_{y_1}^{u_1} < w_{y_3}^{u_1}$.

We have further evaluated PLC-Sleuth with varying volumes of training data, as plotted in Fig. 12, where the accuracy is averaged over 1000 runs by randomly selecting the start time of training sequences from the 9.8-h system logs. PLC-Sleuth's accuracy of graph construction outperforms all other four weighting schemes, and increases with a longer training data—achieving 100% when being trained with a system log of 0.5 h.

Intrusion Detection/Localization With EDS: We next evaluate PLC-Sleuth's intrusion detection/localization under two attack scenarios. The command injection attacks are launched in two ways, i.e.; 1) replacing output command series with designed attack vector in PIO table (e.g., replacing $u_2 = [0.2757, 0.2762, 0.2766]$ with $u_2^a = [0.2967, 0.2968, 0.2969]$) and 2) tampering parameters of control rules (e.g., changing K_p in the rule of $\Delta u_2 = -K_p(\Delta x_2 + 0.17 \times x_2)$ from $K_p = 0.01$ to $K_p^a = 0.1$). Each of the control loop is attacked ten times, i.e., a total number of $10 \times 3 = 30$ attacks are mounted to EDS. Besides, by replaying the corresponding sensor's normal logs, we mount another $10 \times 3 = 30$ cooperative stealthy attacks.

Detection/Localization Accuracy: PLC-Sleuth achieves an average of 98.33%/0.85‰ TP/FP alarm rate, and localizes the forged command with 93.22% (i.e., 55 out of 59) positive predictive value (PPV).

Attack 1: The injected forged command incurs instability to EDS's distillation process, as shown with the red dashed line of the upper subplots in Fig. 14. PLC-Sleuth detects the weight changes of the monitored edges immediately after the attack, and thus triggers alarms. Among the mounted command injection attacks against the tower liquid level (i.e., Loop 3 of EDS), Fig. 13(a) and (b) give PLC-Sleuth's detection results with various configurations of detection window (T) and threshold (μ). Increase of the threshold value hardly affects the TP rate, but reduces FP alarms noticeably.



(a) (b) (c)

Fig. 14. Detecting the two attacks mounted at EDS's three control loops. (a) Loop 1: tower temperature. (b) Loop 2: cooling water flow. (c) Loop 3: tower liquid level.

However, a larger detection window causes a relatively lower TP rate due to its insensitivity to parameter-changing attacks.

Attack 2: The cooperatively replayed sensor logs hide attacks from the operators, as shown with the purple dotted line of the upper subplots in Fig. 14. However, the monitored weight deviates from its normal value noticeably. Note that one missed TP alarm occurs to Loop 1, (i.e., the temperature loop), which has a long control cycle of 1800 s, causing insensitive variation to variables' correlation. The PPV of localization for Loop 1 is relatively low (when compared to Loops 2 and 3) for the same reason.

Detection Latency: We have also examined the latency of `PLC-Sleuth` in detecting PLC intrusions. Although a delay exists for an attack to physically disturb the system, the weight of the control graph changes instantly when the attack is launched, as shown in the middle subplots of Fig. 14. As a result, `PLC-Sleuth` detects these attacks with a short latency, e.g., {50, 12.5, 10.5} s for attacks in Fig. 14(a)–(c), respectively. Besides, Fig. 13(c) shows that a larger detection threshold increases the detection latency. On the other hand, the detection latency decreases with a larger detection window T, because a larger time window facilitates `PLC-Sleuth` capturing the control invariant (i.e., weight $w_{y_j}^{u_k}$) more reliably.

C. Evaluation With TE Process

To further evaluate `PLC-Sleuth` when being deployed at a large-scale control system, we implement `PLC-Sleuth` on a realistically simulated TE process [39]. The TE process contains 12 setpoint variables in S, 41 measurement variables in Y, 12 terminal command variables in U, and 14 transfer variables in R. These variables together form 17 feedback loops, 16 of which form seven multiple feedback loops. Please see [1] for the detailed implementation of feedback loops in the TE process.

Constructing TE's Control Graph: Again, we first examine `PLC-Sleuth`'s accuracy of constructing TE's control graph. Fig. 15 depicts the (correctly) constructed control graph, in which a total number of 45 edges need to be identified.

Data Collection: By simulating a 72-h operation of the TE process, we obtained the training data logged at 1.8 Hz.

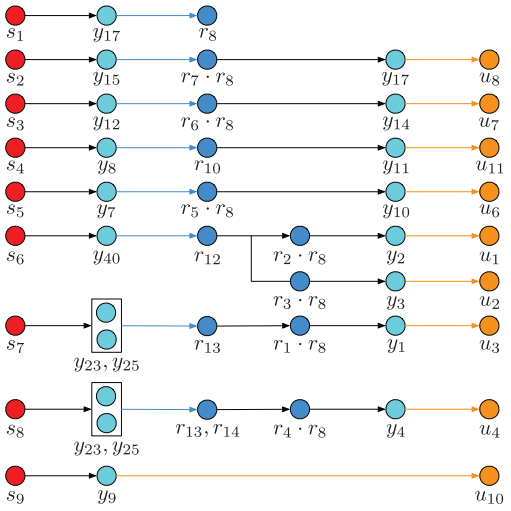


Fig. 15. Control graph of the TE process.

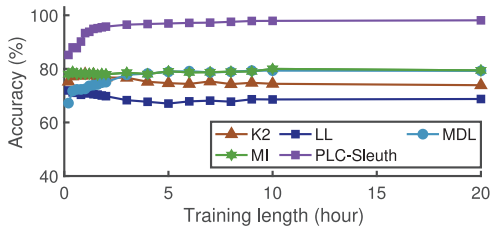


Fig. 16. Constructing TE process's control graph when using different metrics to define $w_{y_j}^{u_k}$ in (7).

Accuracy of Graph Construction: We have evaluated `PLC-Sleuth` with various volumes of training data in different running periods. Fig. 16 plots the results averaged over 1000 runs. `PLC-Sleuth`'s graph construction achieves a high accuracy of 95.76% even when being trained with only a short trace of 2 h, which increases further with a larger volume of training data. The average accuracy reaches 98.11% with a 20-h training data, i.e., `PLC-Sleuth` accurately identifies almost all of TE's 45 edges. Note that the construction error mainly occurs at the transfer variables (e.g., mismatching y_{40} to r_2 in Fig. 15), because of the similarities between transfer nodes and transmit nodes. Fig. 16 also plots the results when constructing the control graph using different weight

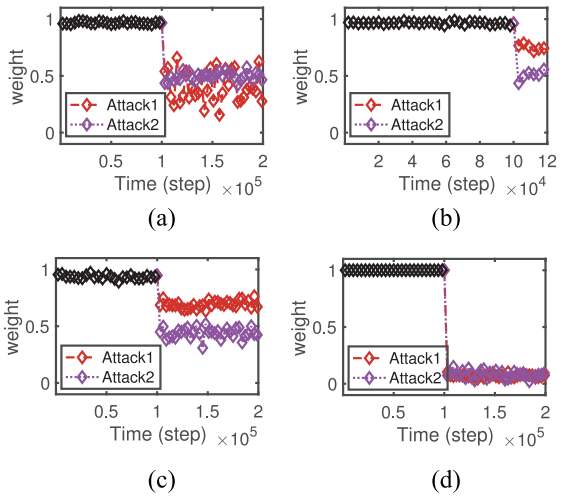


Fig. 17. Command weights $W_{y_j}^{u_k}$ of four control loops under two attacks. (a) Reactor temperature. (b) Reactor level. (c) Reactor pressure. (d) Product quality.

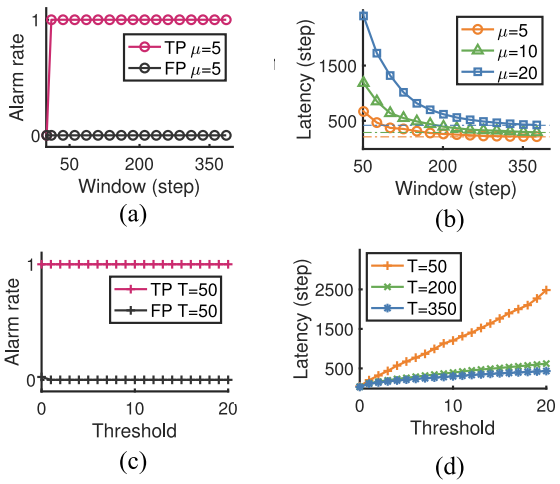


Fig. 18. Impact of the detection window (T) and threshold (μ) on PLC-Sleuth's attack detection for the reactor level control in TE. (a) TP and FP versus window. (b) Latency versus window. (c) TP and FP versus threshold. (d) Latency versus threshold.

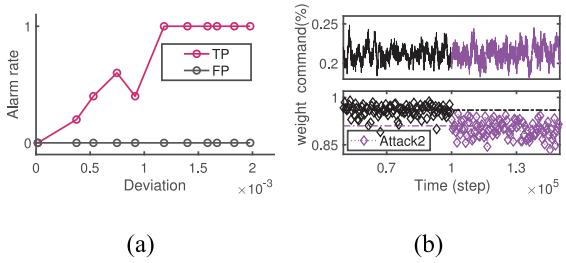


Fig. 19. PLC-Sleuth's sensitivity to deviations from the normal commands for the reactor pressure control in TE. (a) TP and FP v.s. deviation. (b) $w_{y7}^{r_5}$ under a stealthy attack with a 0.16% deviation.

metrics of $w_{y_j}^{u_k}$, showing that PLC-Sleuth achieves the best construction accuracy.

Intrusion Detection/Localization With TE: We next examine PLC-Sleuth's intrusion detection/localization with two attack scenarios in TE. Each of the 17 loops is attacked for ten times in both attack scenarios.

Detection/Localization Accuracy: PLC-Sleuth detects attacks with 100% TP alarm rate, without any FP alarms. For the 340 alarms, PLC-Sleuth localizes the forged command with a PPV of 96.76% (i.e., 329 out of 340).

Attack 1: The forged time series, which are generated by: 1) adding random noise to normal output commands and 2) mixing constant values with random white noises to hide the attack, are injected to each of the 17 control loops. These attacks degrade the physical process by: 1) destabilizing the reactor temperature; 2) overwhelming the reactor; 3) enlarging the reactor pressure; 4) degrading product quality; etc. Fig. 17 shows the changes of weights $w_{y_j}^{u_k}$ under the above attacks. We can see that all the weights of edges $e_{y_j}^{u_k}$ corresponding to the compromised control loops change when the attacks happen, making them detectable by PLC-Sleuth. We further evaluate the impacts of detection window (T) and threshold (μ) using the ten attacks against the TE's reactor level control, as plotted in Fig. 18, showing that detection window and threshold have limited impacts on the TP/FP rate, but have significant influence on the detection latency—the detection latency is inversely proportional to the window size. This is because the detection data with a long period of logs is able to capture better the normal command weight and thus, identify the anomalies.

Attack 2: By injecting the forged command u^a into each control loop and replacing the sensor measurements with their historical normal records, another $17 \times 10 = 170$ attacks are performed. These attacks have similar impacts on the TE process as with Attack 1. The monitored weight $w_{y_j}^{u_k}$ changes significantly regardless of the replayed sensor measurements, as plotted in Fig. 17. Note PLC-Sleuth detects Attack 2 against the reactor pressure control loop with {100%,0%} TPs/FPs, as long as the deviation from normal commands (defined as $\sum |u^a(k) - u(k)| / \sum u(k) \times 100\%$) is larger than 0.12%, as shown in Fig. 19(a). Meanwhile, Fig. 19(b) visualizes the weight change under a 0.16% deviation attack.

Tolerance to Inaccurate Control Graph: We examine PLC-Sleuth's tolerance to the few cases of inaccurately constructed control graph. With PLC-Sleuth, falsely matched edges mainly occur at control loops containing transfer variables, which we call the inner-loop fault (e.g., an edge connecting node y_{40} to r_2 of Fig. 15). For the control graph constructed using other learning metrics, falsely matched edges also include the nodes from different control loops, which we call the interloop fault (e.g., an edge connecting node y_{11} to u_8 in Fig. 15). Fig. 20 shows detection results under the two attack scenarios, by using the control graph with two different construction errors. The control graph with inner-loop construction faults still detects the attack, although with a higher FP rate; the one with interloop faults, however, is not applicable anymore.

Comparison of W_s^y and W_y^u in Intrusion Detection: The control error weights in W_s^y can also be used for attack detection: a persistent deviation from the normal error weight indicates an attack. Fig. 21(a) shows the increases of CUSUM error under a command injection attack, using weights $w_s^y \in W_s^y$ and

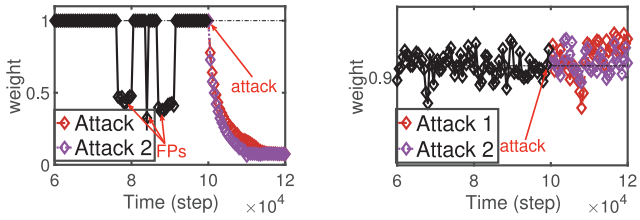


Fig. 20. Weights of falsely matched edges under the two attacks scenarios. (a) Inner-loop fault. (b) Inter-loop fault.

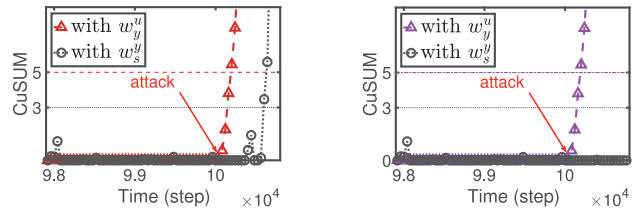


Fig. 21. Comparison of attack detection using two types of edge weight. (a) Attack 1. (b) Attack 2.

$w_y^u \in W_y^u$, respectively. Grounding on the w_y^u, PLC-Sleuth works even for the scenario of cooperative stealthy attack, where the detection scheme with w_s^y failed, as observed in Fig. 21(b). Moreover, PLC-Sleuth raises alarms more timely than the detector using w_s^y in both attack scenarios, providing more response time for ICSs.

VI. Customized Evasive Attacks

We further explore the possible evasive attacks to PLC-Sleuth based on the adversary's knowledge of control systems: 1) the dynamics of the physical process [i.e., how the system state $y_j(t)$ evolves based on command $u(t-1)$] and 2) the control rules of PLCs [i.e., how the command $u_k(t)$ generates based on the sensor reading $y(t)$].

According to the dynamics of physical process, adversaries can predict sensor reading $\hat{y}_j(t)$ at each time instant t, by taking the control commands $u(t-1)$ as inputs. With the prediction of $\hat{y}(t)$, adversaries can further predict the concomitant control commands $\hat{u}(t)$ based on the understanding of PLC control rules. Using the predicted command $\hat{u}_k(t)$ and sensor reading $\hat{y}_j(t)$, adversaries can compromise the PLC with $u_k^a(t) \triangleq \hat{u}_k(t)+\epsilon$ and $y_j^a(t) \triangleq \hat{y}_j(t)+\varepsilon$, where ϵ and ε are the manipulated control command and sensor reading, respectively. This way, adversaries can damage the physical process without being detected by PLC-Sleuth, because the correlation between $u_k^a(t)$ and corresponding $y_j^a(t)$ remains the same as with the normal cases. One of such attack against the TE process, namely, the replaying records attack for both sensors and commands, is shown in Fig. 22. As can be seen, depending on the dynamics of the specific physical process, the impact of such evasive attacks diverges from each other.

However, it is challenging to mount such attacks in practice. First, control loops among the same physical process (e.g., the reactor and the heater of EDS) are interdependent, and do not have a standard dynamics model, making it very difficult for adversaries to predict sensors' behavior. Second, the adversary will need thorough knowledge of not only the pairs of correlated variables in the same control loop but also the specific control algorithm that defines their relationships.

VII. Related Work

Attacks to PLCs: A number of systematic attacks against PLCs have been reported. By exploiting the vulnerabilities of the communication protocols of PLCs, adversaries can manipulate the PLC memory, including the in/output memory [44], control logic memory [17], [45], etc. Moreover, the PLC firmware, which provides an interface between the PLC memory and the hardware input/output modules, has been hacked by malicious rootkits [20], [46], [47]. The compromised firmware issues illegal control commands to the physical process, and reports the faked system states to SCADA at the same time. Atop the above attacks, the adversaries also hide the infected PLC memory by hijacking the SCADA logging traffic [48].

Detecting/Localizing PLC Attacks: To defend against PLC attacks, many detection methods have been proposed using system invariants in PLC program and/or SCADA data. With the PLC program, system invariants of control flow are usually abstracted as explicit rules [49]–[53], by which the legality and correctness of operation code and output variables' states are verified. However, the checking of control flow is time consuming, rendering it unsuitable for real-time attack detection scenarios [54]. With the SCADA data, system invariants of how actuators affect sensor readings are abstracted as physics models [36], [55]–[58] and/or variable correlations [59]–[63]. By predicting system states using these invariants, the manipulated sensor readings are detected. Besides, system invariants between sensor readings and the concomitantly triggered commands are abstracted as a control graph [1] and neural networks [50], [64], [65], which allow the detection of manipulated commands. By actively changing these system invariants, cooperative stealthy attacks are detected based on the deviation between the logged and expected system states [66], [67]. Different from the existing approaches that capture system invariants based on the *a priori* knowledge on if a given data variable represents a sensor reading or a control command [50], [56], [57], [62], PLC-Sleuth identifies system invariants and detects attacks with much less *a priori* knowledge.

VIII. Conclusion

We have proposed PLC-Sleuth, a novel attack detection/localization scheme grounded on a PLC's control graph $\mathcal{G}$. The control graph describes the control invariants of PLC, with the inherent and essential characteristics of control loops. Using the automatically constructed control graph, PLC-Sleuth flags and localizes attacks once the weights in $\mathcal{G}$ deviate from the norm. The evaluation results using EDS and TE process show that PLC-Sleuth can construct a control graph with high accuracy (100% with a log of 0.5 h for EDS and 98.11% with a log of 20 h for TE process). PLC-Sleuth

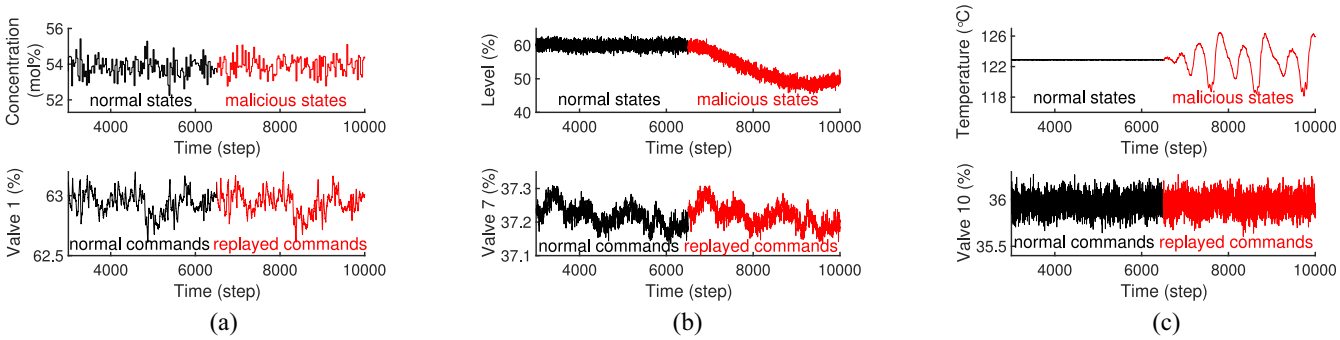


Fig. 22. Impacts of the joint replaying of sensors and commands. (a) No observable impact. (b) Shifting separation level. (c) Unstable reactor temperature.

achieves detection TPs/FPs of {98.33%, 0.85‰} for EDS, and of {100%, 0%} for TE process. In terms of attack localization, `PLC-Sleuth` localizes the forged command with a {93.22, 96.76}% accuracy for EDS and TE process, respectively.

REFERENCES

[1] Z. Yang, L. He, P. Cheng, J. Chen, D. K. Yau, and L. Du, "PLC-Sleuth: Detecting and localizing PLC intrusions using control invariants," in *Proc. 23rd Int. Symp. Res. Attacks Intrusions Defenses (RAID)*, 2020, pp. 333–348.

[2] H. Wu *et al.*, "Dynamic edge access system in IoT environment," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 2509–2520, Apr. 2020.

[3] K. Stouffer, J. Falco, and K. Scarfone, "Guide to industrial control systems (ICS) security," NIST, Gaithersburg, MD, USA, Rep. NIST SP 800-82, 2011.

[4] A. Babay *et al.*, "Deploying intrusion-tolerant SCADA for the power grid," in *Proc. 49th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, 2019, pp. 328–335.

[5] J. Wu, M. Dong, K. Ota, J. Li, and W. Yang, "Sustainable secure management against APT attacks for intelligent embedded-enabled smart manufacturing," *IEEE Trans. Sustain. Comput.*, vol. 5, no. 3, pp. 341–352, Jul.–Sep. 2020.

[6] D. Yuan *et al.*, "Intrusion detection for smart home security based on data augmentation with edge computing," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2020, pp. 1–6.

[7] Q. Li, X. Feng, H. Wang, and L. Sun, "Understanding the usage of industrial control system devices on the Internet," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 2178–2189, Jun. 2018.

[8] "Shamoon: Destructive Threat Re-Emerges With New Sting in its Tail." Symantec Threat Hunter Team. 2018. [Online]. Available: https://symantec-enterprise-blogs.security.com/blogs/threat-intelligence/shamoon-destructive-threat-re-emerges-new-sting-its-tail (Accessed: Sep. 2021).

[9] "Rockwell Automation MicroLogix Controllers and RSLogix 500 Software." Cybersecurity and Infrastructure Security Agency. 2020 [Online]. Available: https://us-cert.cisa.gov/ics/advisories/icsa-20-070-06 (Accessed: Sep. 2021).

[10] "SSA-381684: Improper Password Protection during Authentication in SIMATIC S7-300 and S7-400 CPUs and Derived Products." Siemens Security Advisory by Siemens ProductCERT. 2020. [Online]. Available: https://cert-portal.siemens.com/productcert/pdf/ssa-381684.pdf (Accessed: Sep. 2021).

[11] "Security Notification—Modicon M100/M200/M221 Programmable Logic Controller (V3.0)." Schneider Electric Security Notification. [Online]. Available: https://www.se.com/ww/en/download/document/SEVD-2020-315-05/ (Accessed: Sep. 2021).

[12] J. Klick, S. Lau, D. Marzin, J.-O. Malchow, and V. Roth, "Internet-facing PLCs—A new back orifice," in *Proc. Blackhat USA*, 2015, pp. 1–9.

[13] A. Humayed, J. Lin, F. Li, and B. Luo, "Cyber-physical systems security—A survey," *IEEE Internet Things J.*, vol. 4, no. 6, pp. 1802–1831, Dec. 2017.

[14] A. Keliris and M. Maniatakos, "ICSREF: A framework for automated reverse engineering of industrial control systems binaries," in *Proc. Symp. Netw. Distrib. Syst. Security (NDSS)*, 2019, pp. 1–16.

[15] P. Sun, L. Garcia, and S. Zonouz, "Tell me more than just assembly! Reversing cyber-physical execution semantics of embedded IoT controller software binaries," in *Proc. 49th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, 2019, pp. 349–361.

[16] D. Tychalas, H. Benkraouda, and M. Maniatakos, "ICSFuzz: Manipulating I/Os and repurposing binary code to enable instrumented fuzzing in ICS control applications," in *Proc. 30th USENIX Security Symp. (USENIX Security)*, 2021, pp. 2847–2862.

[17] N. Govil, A. Agrawal, and N. O. Tippenhauer, "On ladder logic bombs in industrial control systems," in *Computer Security*. Cham, Switzerland: Springer Int., 2018, pp. 110–126.

[18] "After Triton, Will the Industrial Threat Landscape Ever be the Same?" Charles Cooper. 2018. [Online]. Available: https://www.symantec.com/blogs/feature-stories/after-triton-will-industrial-threat-landscape-ever-be-same (Accessed: Sep. 2021).

[19] N. Falliere, L. O. Murchu, and E. Chien, "W32. Stuxnet dossier," Symantec Corp., Security Response, Tempe, AZ, USA, Rep., vol. 5, no. 6, 2011.

[20] L. Garcia, F. Brasser, M. H. Cintuglu, A.-R. Sadeghi, O. A. Mohammed, and S. A. Zonouz, "Hey, My malware knows physics! Attacking PLCs with physical model aware rootkit," in *Proc. Netw. Distrib. Syst. Security (NDSS) Symp.*, 2017, pp. 1–15.

[21] G. A. Dunning, *Introduction to Programmable Logic Controllers*, Cengage Learn., 2005.

[22] H. Z. Kister, "What caused tower malfunctions in the last 50 years?" *Chem. Eng. Res. Design*, vol. 81, no. 1, pp. 5–26, 2003.

[23] R. Bellman, I. Glicksberg, and O. Gross, "On the 'bang-bang'İcontrol problem," *Quart. Appl. Math.*, vol. 14, no. 1, pp. 11–18, 1956.

[24] K. J. Åström, T. Hägglund, and K. J. Astrom, *Advanced PID Control*. Research Triangle Park, NC, USA: ISA Instrum. Syst. Autom. Soc., 2006.

[25] H. Berger, *Automating With STEP7 in STL and SCL: Programmable Controllers Simatic S7-300/400*. Erlangen, Germany: Publicis Publ., 2012.

[26] P. Albertos and S. Antonio, *Multivariable Control Systems: An Engineering Approach*. London, U.K.: Springer, 2006.

[27] G. F. Franklin, J. D. Powell, A. Emami-Naeini, and J. D. Powell, *Feedback Control of Dynamic Systems*, vol. 3. Reading, MA, USA: Addison-Wesley, 1994.

[28] Y. Mo and B. Sinopoli, "Secure control against replay attacks," in *Proc. 47th Annu. Allerton Conf. Commun. Control Comput. (Allerton)*, 2009, pp. 911–918.

[29] R. Steuer, J. Kurths, C. O. Daub, J. Weise, and J. Selbig, "The mutual information: Detecting and evaluating dependencies between variables," *Bioinformatics*, vol. 18, no. S2, pp. S231–S240, 2002.

[30] T. McAvinew and R. Mulley, *Control System Documentation: Applying Symbols and Identification*. Research Triangle Park, NC, USA: ISA Instrum. Syst. Autom. Soc., 2004.

[31] X.-W. Chen, G. Anantha, and X. Lin, "Improving Bayesian network structure learning with mutual information-based node ordering in the K2 algorithm," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 5, pp. 628–640, May 2008.

[32] X. Fan and C. Yuan, "An improved lower bound for Bayesian network structure learning," in *Proc. 29th AAAI Conf. Artif. Intell.*, pp. 3526–3532, 2015.

[33] S. Skogestad and I. Postlethwaite, *Multivariable Feedback Control: Analysis and Design*, vol. 2. Chichester, U.K.: Wiley, 2007.

[34] "Fisher 4320 Wireless Position Monitor." Emerson. 2009. [Online]. Available: https://www.emerson.com/en-us/catalog/fisher-4320 (Accessed: Sep. 2021).

[35] "SIEMENS SIPART PS2 (6DR5···) Electropneumatic Positioners." Siemens. 2017. [Online]. Available: http://www.lesman.com/unleashd/catalog/accessor/Siemens_SIPART-PS2/Siemens-SIPART-PS2-man-A5E03436620-AB-2017-01.pdf

[36] D. Urbina *et al.*, "Limiting the impact of stealthy attacks on industrial control systems," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security (CCS)*, 2016, pp. 1092–1105.

[37] "Exporting Archived Data From WinCC With the OLE DB Provider." Siemens Industry Online Support. 2019. [Online]. Available: https://cache.industry.siemens.com/dl/files/261/38132261/att_946466/v3/38132261__Application_Reverse_Osmosis_DOC_en.pdf (Accessed: Sep. 2021).

[38] W. Mahnke, S.-H. Leitner, and M. Damm, *OPC Unified Architecture*. Heidelberg, Germany: Springer, 2009.

[39] J. J. Downs and E. F. Vogel, "A plant-wide industrial process control problem," *Comput. Chem. Eng.*, vol. 17, no. 3, pp. 245–255, 1993.

[40] G. F. Cooper and E. Herskovits, "A Bayesian method for the induction of probabilistic networks from data," *Mach. Learn.*, vol. 9, no. 4, pp. 309–347, 1992.

[41] R. R. Bouckaert, "Bayesian belief networks: From construction to inference," Ph.D. dissertation, Faculteit Wiskunde Informatica, Universiteit Utrecht, Utrecht, The Netherlands, 1995.

[42] W. Lam and F. Bacchus, "Learning Bayesian belief networks: An approach based on the MDL principle," *Comput. Intell.*, vol. 10, no. 3, pp. 269–293, 1994.

[43] "WinCC: Working with WinCC." Siemens AG. 2016. [Online]. Available: https://cache.industry.siemens.com/dl/files/220/109736220/att_879788/v1/WinCC_Working_with_WinCC_en-US_en-US.pdf (Accessed: Dec. 2021).

[44] A. Erba *et al.*, "Constrained concealment attacks against reconstruction-based anomaly detectors in industrial control systems," in *Proc. Annu. Comput. Security Appl. Conf.*, 2020, pp. 480–495.

[45] J. H. Castellanos, M. Ochoa, A. A. Cardenas, O. Arden, and J. Zhou, "AttkFinder: Discovering attack vectors in PLC programs using information flow analysis," in *Proc. 24th Int. Symp. Res. Attacks Intrusions Defenses*, 2021, pp. 235–250.

[46] Z. Basnight, J. Butts, J. Lopez, and T. Dube, "Firmware modification attacks on programmable logic controllers," *Int. J. Crit. Infrastruct. Protect.*, vol. 6, no. 2, pp. 76–84, 2013.

[47] A. Abbasi and M. Hashemi, "Ghost in the PLC designing an undetectable programmable logic controller rootkit via pin control attack," in *Proc. Black Hat Europe*, 2016, pp. 1–35.

[48] S. Kalle, N. Ameen, H. Yoo, and I. Ahmed, "CLIK on PLCs! Attacking control logic with decompilation and virtual PLC," in *Proc. Binary Anal. Res. Workshop Netw. Distrib. Syst. Security Symp. (NDSS)*, 2019, pp. 1–12.

[49] L. McMinn and J. Butts, "A firmware verification tool for programmable logic controllers," in *Proc. Int. Conf. Crit. Infrastruct. Protect.*, 2012, pp. 59–69.

[50] M. Zhang *et al.*, "Towards automated safety vetting of PLC code in real-world plants," in *Proc. IEEE Symp. Security Privacy (S&P)*, 2019, pp. 522–538.

[51] S. Biallas, J. Brauer, and S. Kowalewski, "Arcade.PLC: A verification platform for programmable logic controllers," in *Proc. 27th IEEE/ACM Int. Conf. Autom. Softw. Eng.*, 2012, pp. 338–341.

[52] S. McLaughlin, S. A. Zonouz, D. J. Pohly, and P. D. McDaniel, "A trusted safety verifier for process controller code," in *Proc. Netw. Distrib. Syst. Security (NDSS) Symp.*, 2014, pp. 1–15.

[53] S. Zonouz, J. Rrushi, and S. McLaughlin, "Detecting industrial control malware using automated PLC code analytics," *IEEE Security Privacy*, vol. 12, no. 6, pp. 40–47, Nov./Dec. 2014.

[54] I. Ahmed, S. Obermeier, S. Sudhakaran, and V. Roussev, "Programmable logic controller forensics," *IEEE Security Privacy*, vol. 15, no. 6, pp. 18–24, Nov./Dec. 2017.

[55] A. A. Cárdenas, S. Amin, Z.-S. Lin, Y.-L. Huang, C.-Y. Huang, and S. Sastry, "Attacks against process control systems: Risk assessment, detection, and response," in *Proc. 6th ACM Symp. Inf. Comput. Commun. Security*, 2011, pp. 355–366.

[56] H. Choi *et al.*, "Detecting attacks against robotic vehicles: A control invariant approach," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security (CCS)*, 2018, pp. 801–816.

[57] R. Quinonez, J. Giraldo, L. Salazar, E. Bauman, A. Cardenas, and Z. Lin, "SAVIOR: Securing autonomous vehicles with robust physical invariants," in *Proc. 29th USENIX Security Symp. (USENIX Security)*, 2020, pp. 895–912.

[58] M. Salehi and S. Bayat-Sarmadi, "PLCDefender: Improving remote attestation techniques for PLCs using physical model," *IEEE Internet Things J.*, vol. 8, no. 9, pp. 7372–7379, May 2021.

[59] Y. Wang, Z. Xu, J. Zhang, L. Xu, H. Wang, and G. Gu, "Srid: State relation based intrusion detection for false data injection attacks in scada," in *Proc. Eur. Symp. Res. Comput. Security*, 2014, pp. 401–418.

[60] S. Adepu and A. Mathur, "Distributed detection of single-stage multipoint cyber attacks in a wate treatment plant," in *Proc. 11th ACM Asia Conf. Comput. Commun. Security*, 2016, pp. 449–460.

[61] 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 Symp. Security Privacy (S&P)*, 2018, pp. 240–252.

[62] 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. Netw. Distrib. Syst. Security (NDSS) Symp.*, 2019, pp. 1–15.

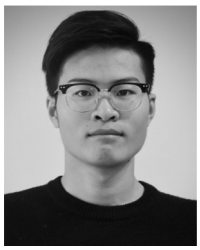
[63] Z. Hau and E. C. Lupu, "Exploiting correlations to detect false data injections in low-density wireless sensor networks," in *Proc. 5th Cyber-Phys. Syst. Security Workshop*, 2019, pp. 1–12.

[64] H. R. Ghaeini *et al.*, "PAtt: Physics-based attestation of control systems," in *Proc. 22nd Int. Symp. Res. Attacks Intrusions Defenses (RAID)*, 2019, pp. 165–180.

[65] J. Sun, Y. Chen, and C. Poskitt, "Code integrity attestation for PLCs using black box neural network predictions," in *Proc. ACM Joint Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng. (ESEC/FSE)*, 2021, pp. 32–44.

[66] J.-H. Cho *et al.*, "Toward proactive, adaptive defense: A survey on moving target defense," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 1, pp. 709–745, 1st Quart., 2020.

[67] M. Liu, C. Zhao, Z. Zhang, R. Deng, P. Cheng, and J. Chen, "Converter-based moving target defense against deception attacks in DC microgrids," *IEEE Trans. Smart Grid*, early access, Nov. 19, 2021, doi: 10.1109/TSG.2021.3129195.



Zeyu Yang is currently pursuing the Ph.D. degree in control science and engineering with Zhejiang University, Hangzhou, China.

His research interests include cyber–physical system security and industrial control system security.

Liang He (Senior Member, IEEE) received the B.E. degree from Tianjin University, Tianjin, China, in 2006, and the Ph.D. degree from Nankai University, Tianjin, in 2011.

He is currently an Assistant Professor with the University of Colorado, Denver, CO, USA. He worked as a Research Fellow with the University of Michigan, Ann Arbor, MI, USA, from 2015 to 2017. His research interests include CPS, IoT, and mobile computing.

Hua Yu received the B.E. degree in automation from Zhejiang University, Hangzhou, China, in 2020, where he is currently pursuing the M.A.Eng. degree in control science and engineering.

His research interests include cyber–physical security and industrial control system security.

Chengcheng Zhao (Member, IEEE) received the B.Sc. degree in measurement and control technology and instrument from Hunan University, Changsha, China, in 2013, and the Ph.D. degree in control science and engineering from Zhejiang University, Hangzhou, China, in 2018.

She worked as a Postdoctoral Fellow with the College of Control Science and Engineering, Zhejiang University from 2018 to 2021, where she is currently an Associate Researcher with the College of Control Science and Engineering. Her research interests include consensus and distributed optimization, distributed energy management and synchronization in smart grids, and security and privacy in networked systems.

Peng Cheng (Member, IEEE) received the B.E. degree in automation and the Ph.D. degree in control science and engineering from Zhejiang University, Hangzhou, China, in 2004 and 2009, respectively.

He is currently a Professor with the College of Control Science and Engineering, Zhejiang University. His research interests include networked sensing and control, cyber–physical systems, and control system security.

Prof. Cheng serves as an Associate Editor of IEEE TRANSACTIONS ON CONTROL OF NETWORK SYSTEMS, *Wireless Networks*, and *International Journal of Communication Systems*. He also serves/served as the Guest Editor for IEEE TRANSACTIONS ON AUTOMATIC CONTROL, IEEE TRANSACTIONS ON SIGNAL AND INFORMATION PROCESSING OVER NETWORKS, and IEEE TRANSACTIONS ON CONTROL OF NETWORK SYSTEMS. He served as the TPC Co-Chair of IEEE IOV 2016, a local arrangement Co-Chair for ACM MobiHoc 2015, and the Publicity Co-Chair for IEEE MASS 2013.

Jiming Chen (Fellow, IEEE) received the B.Sc. and Ph.D. degrees in control science and engineering from Zhejiang University, Hangzhou, China.

He was a Visiting Researcher with the University of Waterloo, Waterloo, ON, Canada, from 2008 to 2010. He is currently a Qiushi Chair Professor with the College of Control Science and Engineering, a Vice Dean of Faculty of Information Technology, and a Deputy Director of the State Key Laboratory of Industrial Control Technology, Zhejiang University. His research interests include sensor networks, IoT, networked control, and cyber security.

Dr. Chen serves/served an Associate Editor for several international Journals, including IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, and *ACM Transactions on Embedded Computing Systems*. He has been a Distinguished Lecturer of IEEE Vehicular Technology Society since 2015.