PLC-Sleuth: Detecting and Localizing PLC Intrusions Using Control Invariants

Zeyu Yang
Zhejiang University

Liang He
University of Colorado Denver

Peng Cheng
Zhejiang University

Jiming Chen
Zhejiang University

David K.Y. Yau
Singapore University of Technology and Design

Linkang Du
Zhejiang University

Abstract

Programmable Logic Controllers (PLCs) are the ground of control systems, which are however, vulnerable to a variety of cyber attacks, especially for networked control systems. To mitigate this issue, we design `PLC-Sleuth`, a novel non-invasive intrusion detection/localization system for PLCs, grounding 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 prototype of Secure Ethanol Distillation System (SEDS) and a realistically simulated Tennessee Eastman (TE) process.

1 Introduction

Background. Commodity *Programmable Logic Controllers* (PLCs) are proved vulnerable to cyber attacks [1, 2]. Researchers have identified >36.7K PLCs that can be accessed by scanning the ports of common communication protocols, such as Modbus and Siemens S7 [3, 4]. Symantec has also confirmed the feasibility of hijacking a number of mission-critical PLCs [5], such as acquiring credentials of the target PLC to administer destructive payloads. Keliris and Maniatakos have designed an autonomous process to compromise PLCs [6], facilitating the executable program injection [7] or firmware modification [8]. After compromising the PLC, adversaries can mount a variety of attacks, including but not limited to:

- *Command Injection Attacks.* The malware TRITON was launched remotely in Saudi Arabia in 2017 [9] 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.
- *Cooperative Stealthy Attacks.* To hide attacks from being detected, adversaries can even mount advanced stealthy attacks to PLCs, by not only tampering 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 worm Stuxnet which damaged hundreds of centrifuges [10]. A firmware vulnerability of Allen Bradley PLCs exposed in 2017 [11] allows modifying the PLCs' control commands and forging sensor readings.

Protecting PLCs Using Control Invariants. To mitigate the above issues, we design a non-invasive data-driven intrusion detection system (IDS) for PLCs, which we call `PLC-Sleuth`. The foundation of `PLC-Sleuth` is a set of *control invariants* that exist pervasively in all control systems: the control commands issued by PLCs correlate strongly with the concomitant sensor readings. `PLC-Sleuth` automatically identifies these control invariants using system's SCADA logs, and abstracts them as a control graph, in which the nodes represent system variables, such as commands, sensor readings, and control setpoints, and the weights of edges quantify the strength of correlations between system variables. `PLC-Sleuth` then captures the normal behavior of the weights of graph edges using data-driven approaches, and detects, at runtime, the anomalies — edges with abnormal weights indicate the control channels between corresponding system variables have been compromised. `PLC-Sleuth` has the following salient properties.

- *A 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 time series of issued control commands and the concomitant sensor readings. This physically-induced correlation (i.e., control invariant) makes `PLC-Sleuth` *reliable* as the correlation will not change unless the control rules are updated, and *pervasively deployable* because such control invariants exist in all control systems.
- *A Non-Invasive IDS.* Taking the SCADA logs of a control system as input, `PLC-Sleuth` automatically identifies the control invariants, and uses them to protect the PLCs thereof, via data-driven approaches without probing/perturbing the system, i.e., `PLC-Sleuth` is non-invasive and thus easy to deploy.
- *An IDS Localizing Intrusions.* `PLC-Sleuth`, besides detecting intrusions at PLCs, also localizes the compromised con-

trol loops thereof, facilitating swift repair/forensics of the PLCs; the control system otherwise remains unreliable no matter how well the intrusions are detected.

We have implemented and evaluated PLC-Sleuth using our Secure Ethanol Distillation System (SEDS) prototype and a representative/realistic TE chemical process. We first evaluate PLC-Sleuth's identification of control invariants — in the form of a control graph — using various volumes of training data. We then evaluate PLC-Sleuth's intrusion detection/localization using the constructed control graph against command injection attacks and cooperative stealthy attacks, with various deviations from the normal commands. The results show that for SEDS 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, and localizes compromised control loops with {93.22%, 96.76%} accuracy.

2 Preliminaries and Basic Idea

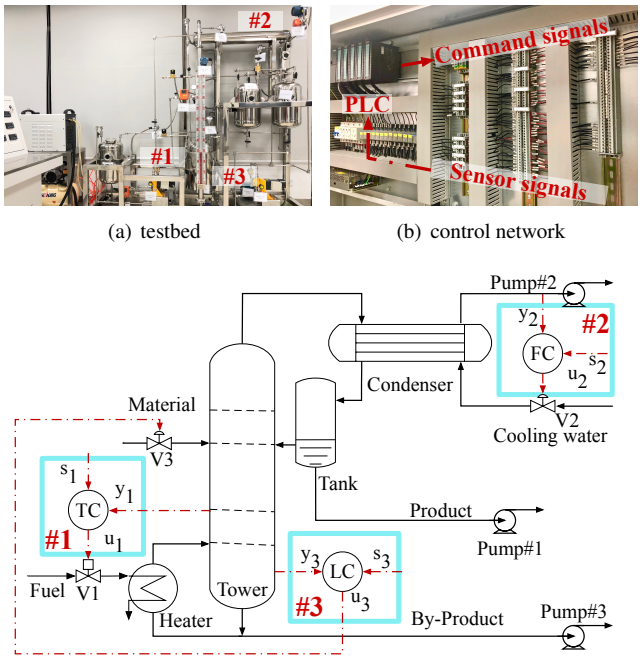
Here we present the necessary background of control systems and the basic idea of PLC-Sleuth, using our prototype of an ethanol distillation control system, called SEDS (Secure Ethanol Distillation System), as an example. Distillation is the process of separating constituents in a liquid mixture based on the differences in their volatility. SEDS is a scaled-down but fully operational distillation plant, which purifies alcohol from water and is capable of producing alcohol with a purity of 90%.

2.1 SEDS Prototype

A typical control system has four main components organized as feedback control loops [12]:

- A *physical process*, e.g., an ethanol distillation process;
- A set of *sensors* that measure the physical states y of the process, e.g., the temperature and liquid level of the relevant distillation tower;
- A set of *controllers* that generate a set of control commands u based on the control error $x := s - y$, i.e., the difference between sensor reading y and the expected setpoint s;
- A set of *actuators* that operate as instructed by the controllers' commands u, e.g., the valves controlling the feeding of materials.

Fig. 1(a) shows our prototype of SEDS, implemented using a control network consisting of sensors, actuators and a SIMATIC S7-300 PLC, as shown in Fig. 1(b). The PLC is equipped with a 315-2 PN/DP CPU, one 32bits DI module, two 8×13 bits AI modules, one 32 bits DO module, and one 8×12 bits AO module. SEDS's operation is monitored/logged by a WinCC SCADA system, including 3 setpoints, 11 sensor readings, and 3 control commands. The



(a) testbed (b) control network

(c) process flow diagram

Figure 1: The Secure Ethanol Distillation System (SEDS) prototype in our lab.

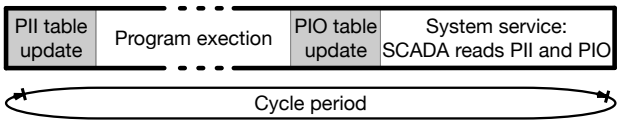
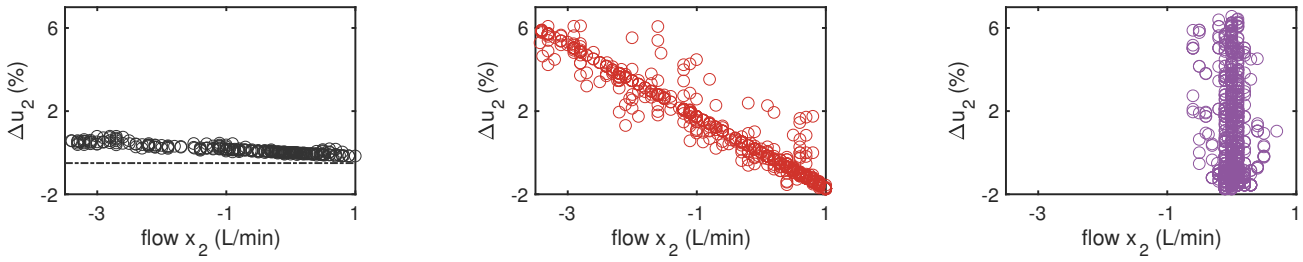


Figure 2: Sequence of CPU executions in PLC [13].

control rules of SEDS are programmed with ladder logic using the Siemens Step7 software. SEDS purifies alcohol from the water with two basic operations — i.e., tower boiling and condenser cooling — realized using three feedback loops, as shown in Fig. 1(c).

- **Loop-1** regulates the tower temperature to a pre-defined level s_1, at which SEDS achieves a high separation efficiency. The temperature controller (TC) generates a 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/closing of valve V_1 (and hence the fuel flow) to regulate the tower temperature to s_1.

- **Loop-2** maintains the flow of the condenser's cooling water — monitored with a liquid level sensor y_2 — around a pre-defined level 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 damage the pump physically. The flow



(a) normal operation without attack (b) with command injection attack (c) with cooperative stealthy attack

Figure 3: Correlations between SEDS's command u_2 and flow difference x_2.

controller (FC) sends a command u_2 to the valve V_2 to regulate the cooling water based on $x_2 = s_2 - y_2$.

- **Loop-3** controls the level of the tower's liquid material (i.e., sensor reading y_3) to an optimized setpoint s_3, which is also affected by the ethanol reflux in Loop-2. An unstable liquid level induces premature flood and thus degrades distillation efficiency [14]. Similar to the TC and FC, the level controller (LC) of Loop-3 generates a command u_3 based on the difference $x_3 = s_3 - y_3$ and sends it to valve V_3 to control the distillation tower's liquid level, by regulating the input flow.

The specific rules of these control loops are implemented in SEDS's PLC, using the Bang-Bang and Proportional-Integral-Derivative (PID) algorithms, which determine the control signals u based on the current and historical control errors x [15, 16]. In short, SEDS controls system states $\{y_1, y_2, y_3\}$ based on 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, sensors $\{y_4, \ldots, y_{11}\}$ provide more information for system monitoring.

2.2 Attack Model

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

a) The attacker can mount the command injection attack during the program execution period (see Fig. 2) by downloading malicious code to PLC.

b) 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 PII table, which happens after the execution of malicious code (see Fig. 2). This way, the attacker can deceive the system monitor to conclude a normal operation even if a successful attack has been launched [10].

c) The attacker cannot modify the record of actual commands issued during system operation, i.e., any forged commands issued by the attacker will be logged as they are. This is because PIO table logs all commands and remains unchanged after the execution of the program [17]. By reading from the PIO table (see Fig. 2), SCADA obtains control commands that are actually issued.

We next use a cooperative stealthy attack mounted at SEDS to motivate `PLC-Sleuth`. The forged commands disturb SEDS's normal operation, while the faked system logs deceive the system monitor to conclude a normal system operation during/after the attack. Specifically, let us consider the case that an attacker aims to degrade the distillation quality by hacking SEDS's Loop-2 to trigger an unstable reflux. The increment of command u_2 under a normal operation is proportional to x_2, i.e., a $x_2 := (s_2 - y_2)$L/min flow difference triggers the adjustment of the valve by

$$\Delta u_2 = -0.01(\Delta x_2 + 0.17 \times x_2). \quad (1)$$

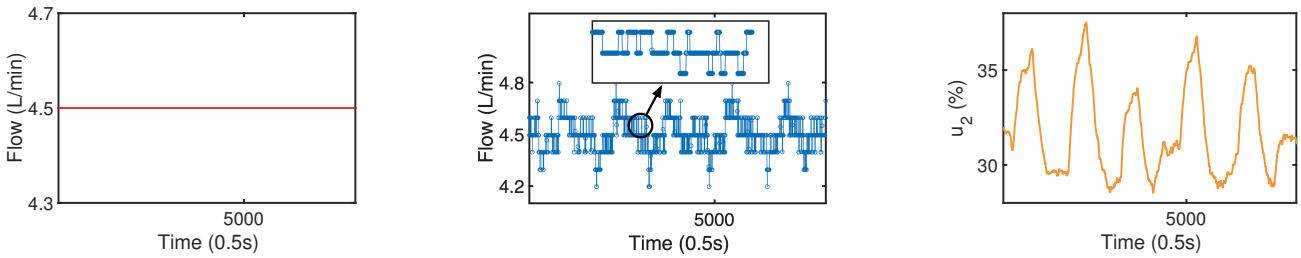
By exploiting the PLC's vulnerabilities, e.g., modifying the s7otbxdx.dll file of Step7 [10], the attacker downloads a payload to the PLC and implement a malicious control rule of

$$\Delta u_2^a = -0.1(\Delta x_2 + 0.17 \times x_2), \quad (2)$$

thus degrading the stability of the cooling process and causing oscillations in ethanol reflux. Note that the tower level y_3 will also be affected by this hacked Δu_2^a due to an unstable ethanol reflux, degrading the distillation quality further. Also, to hide the abnormal deviations of $\{y_2, y_3\}$, the attacker replaces the sensor logs of $\{y_2^a, y_3^a\}$ with historical steady-state records, by overwriting the PLC's process-image table. This way, the sensor logs (i.e., $\{y_2, y_3\}$) appear normal to the operator whereas the SEDS is actually manipulated by the attacker.

2.3 Basic Idea of `PLC-Sleuth`

As stated above, the control command u is generated based on the control error $x = s - y$, using the pre-defined control rules written in the PLC. This incurs strong/reliable correlations between u and x, which we use as the system's *control invariants*. Fig. 3 visualizes such a control invariant between Δu_2 and x_2: when SEDS is attacked as described above, the control invariant in Fig. 3(a) changes noticeably due to the malicious control rule in Eq. (2), for both the command injection attack (Fig. 3(b)) and the cooperative stealthy attack



(a) log of setpoint s_2 (b) log of sensor y_2 (c) log of command u_2

Figure 4: Different patterns of setpoint, sensor, command variables in SEDS's Loop-2.

(Fig. 3(c)). PLC-Sleuth uses these control invariants to detect PLC intrusions, by detecting deviations of these control invariants from their normal behaviors. The challenge is, in turn, to identify/abstract the system's control invariants, which PLC-Sleuth addresses using a control graph.

3 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*. Specifically, the control graph is defined as a weighted and directed acyclic graph $\mathcal{G}(\mathtt{V},\mathtt{E},\mathtt{W})$, where: (i) $\mathtt{V}$ is the set of nodes representing system variables (i.e., setpoints, sensor readings, and commands), (ii) $\mathtt{E}$ is the set of directed edges connecting nodes in $\mathtt{V}$, representing the correlations among system variables, and (iii) $\mathtt{W}$ is the set of edge weights, representing the strength of the correlations described by $\mathtt{E}$. Specifically,

- $\mathtt{V} = \{S, Y, U\}$ consists of a setpoint node set S, a sensor node set Y, and a command node set U, which can be obtained/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., $|U| \geq |S|$.
- $\mathtt{E} = \{E_s^y, E_y^u\}$ 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 exists in $\mathtt{E}$ when its two nodes belong to the same control loop. The commonly used decentralized control scheme — one sensor is fed back for the generation of one targeted control command, and vise versa [18] — makes both the out-degree of nodes in Y and the in-degree of nodes in U equal to 1.
- $\mathtt{W} = \{W_s^y, W_y^u\}$ consists of an error weight set W_s^y for control error edges in E_s^y and a command weight set W_y^u for control command edges in E_y^u. PLC-Sleuth exploits W_y^u to detect and localize PLC intrusions, by monitoring W_y^u in real time and detecting the anomalies thereof. Note that the error weight set W_s^y could also be used to detect PLC attacks/anomalies, i.e., the sensor readings Y should be close to the corresponding setpoint S in a stable control system [12]. This approach, however, is (i) vulnerable to replayed sensor logs in cooperative stealthy attacks [19], and (ii) orthogonal to PLC-Sleuth and thus not discussed here. Also, the fact that $\mathcal{G}$ includes two types of edges/weights differs it from the traditional structure learning graphs [20,21].

Next we explain PLC-Sleuth's classification of nodes and edges. For the ease of description, we use s_i, y_j, and u_k to denote nodes belonging to S, Y, and U, respectively, and use $e_{s_i}^{y_j}$ and $e_{y_j}^{u_k}$ to represent edges belonging to E_s^y and E_y^u.

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) and Figs. 20(a)-20(c) in Appendix.

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) and Figs. 20(d)-20(n) in Appendix.

Command Node Set U. A command node $u_k \in U$ reflects the actuation to achieve the pre-defined 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) and Figs. 20(o)-20(q) in Appendix.

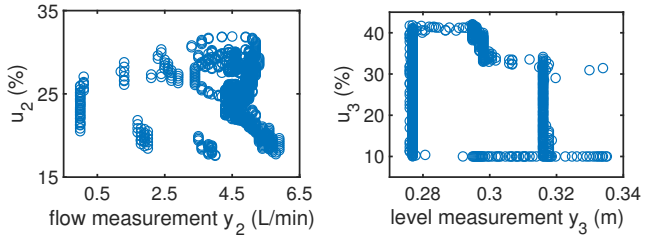
Error Weight Set W_s^y. The weight of control error edge $w_{s_i}^{y_j} \in W_s^y$ captures the difference between system state y_j and the corresponding setpoint s_i over the recent l samples,

$$w_{s_i}^{(y_1,\cdots,y_d)} = |\sum_{t=1}^{l}(s_i(t) - f(y_1(t),\cdots,y_d(t)))|, \qquad (3)$$

where d is the number of sensors used to estimate the system state using function f, which can be obtained from control algorithms' expressions or binary files [22].

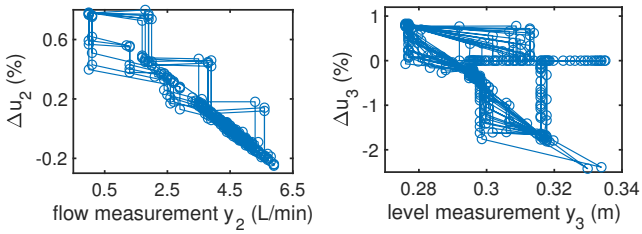
Command Weight Set W_y^u. Mutual information (MI) — a metric commonly used to characterize conditional dependency between variables using their posterior probability distribution [23] — is an intuitive metric to define command weight $w_{y_j}^{u_k} \in W_y^u$. Specifically, the weight of command edge can be defined as the normalized mutual information, i.e.,

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



(a) cooling water loop (b) tower liquid level loop

Figure 5: Static correlations (calculated according to Eq. (4)) between SEDS's commands and sensor readings.



(a) cooling water loop (b) tower liquid level loop

Figure 6: Transition correlations (calculated according to Eq. (7)) between SEDS's commands and sensor readings.

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

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

However, because a 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 SEDS's sensor readings $\{y_2, y_3\}$ and control commands $\{u_2, u_3\}$, showing their low correlation when only considering samples collected at a given time instant — the weighting scheme in Eq. (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 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 mutual information (TMI)* as,

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

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

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

The weighting of MI and TMI are compared statistically in Table 1 of Sec. 5.

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

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

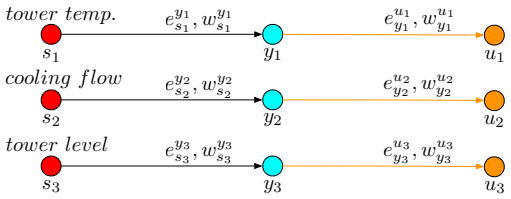


Figure 7: Control graph of SEDS.

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

4 Design of `PLC-Sleuth`

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

4.1 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 [24]. As an alternative, `PLC-Sleuth` constructs the control graph $\mathcal{G}$ automatically with a data-driven approach, using the historical (and normal) SCADA logs.

4.1.1 Identifying Vertex Set

`PLC-Sleuth` identifies the nodes of $\mathcal{G}$ automatically using the SCADA logs: (i) setpoint variables are of constant values, (ii) sensor readings are of consistent and small vibrations, and (iii) 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

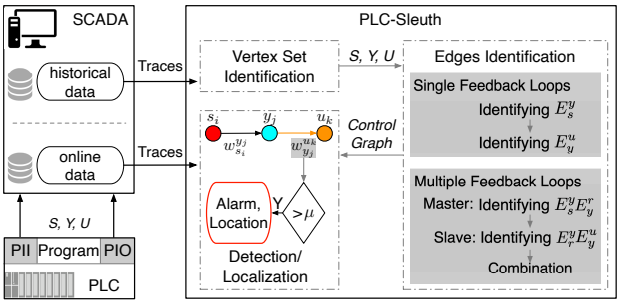


Figure 8: Architecture of PLC-Sleuth.

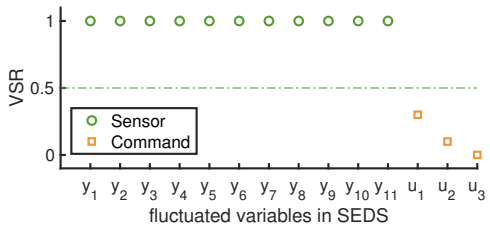


Figure 9: VSR of fluctuated variables in SEDS.

with consistent fluctuation. Here we define vibrations as the curve crests/troughs $y_j(k)$ in y_j, where $y'_j(k) \times y'_j(k+1) < 0$, and VSR_{y_j} as the maximum ratio of crests/troughs in a period samples of y_j. With a 10 persistent samples setting, all sensor nodes in SEDS have the VSR of 1, as shown in Fig. 9. After selecting nodes with high VSR (larger than 0.5 in Fig. 9) as sensor node set, the unclassified variables are classified as the command node set U.

4.1.2 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 SEDS. We will later extend the graph construction to more complex control loops involving multiple (and likely coupled) feedback channels.

• 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 $\tilde{s}_i \in S$, we identify 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 Eq. (3)) is the smallest among all the $\{\tilde{s}_i, y_j\}$ pairs. Taking SEDS as an example, $\{s_1, s_2, s_3\}$ are set as $\{51°\text{C}, 4.5\text{L/min}, 0.29\text{m}\}$, 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.

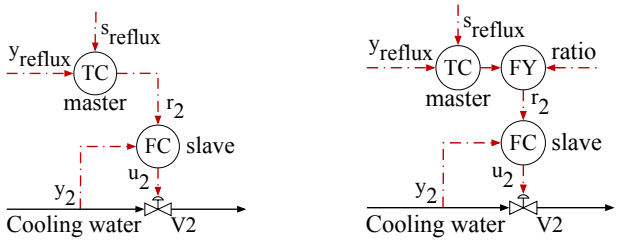
• Step-II: Identifying E_y^u Edges. For each of the above identified error edges $e_{s_i}^{y_j} \in E_s^y$, PLC-Sleuth further matches its sensor node y_j to the command node u_k of the same control loop. Typically, a control algorithm regulates an actuator's action based on sensor measurements to minimize the control error, making a command node u_k closely correlated with the sensor reading y_j of the same control loop. We use Eq. (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, we match 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 Eq. (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 does not only affect control loop to which it belongs, but it may also affect other control loops due to their physical interactions. For example, the command u_3 in SEDS 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^*, we delete its edges from $\mathcal{G}$ by keeping only the edge with the maximal weight. An example of the repeated node is u_3 of SEDS, 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 Step-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.

4.1.3 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 [18, 25] — to improve their efficiency. These coupled loops usually operate in a master-slave manner, as illustrated in Fig. 10. Taking SEDS as an example, to stabilize the reflux temperature at a setpoint level, we can deploy another temper-



(a) cascade control loop (b) ratio control loop

Figure 10: Examples of coupled feedback control schemes.

ature controller TC to SEDS's Loop-#2 to adjust, in real-time, FC's setpoint, making FC the 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. Also, only a subset of U are sent to actuators in the coupled feedback control [1], and the remaining commands form the transfer node set R.

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

• **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 Step-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 $\{R, Y \text{ and } U-R\}$. Error edges in E_r^y of slave loops are identified similarly to Step-I, by replacing Eq.(3) with $w_{r_m}^{(y_1,\cdots,y_d)} = \min\{|\sum_{t=1}^{L}(r_m - f(y_1,\cdots,y_d))|, |\sum_{t=1}^{L}(r_m * r_n - f(y_1,\cdots,y_d))|\}$. 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 Step-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, we further match $\hat{r}_n$ with a slave setpoint node $\tilde{r}_m$ with the maximal $w_{\hat{r}_n}^{\tilde{r}_m}$ (calculated with Eq. (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 each of the 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 $w_{\tilde{y}_j}^{\hat{r}_n}$), we determine the specific control graph (i.e., single feedback loops or multiple feedback loops) to which the nodes $\tilde{s}_i$, $\tilde{y}_j$, $\tilde{r}_m$, $\hat{r}_n$, $\tilde{u}_k$ belong.

[1] Actuators will send received commands back to SCADA system [26,27].

4.2 Detection/Localization of PLC Intrusions

`PLC-Sleuth` then detects/localizes, at runtime, PLC intrusions using an online norm model constructed using $\mathcal{G}$.

As illustrated in Sec. 2.3, tampering the control command will violate the control invariants between commands and the corresponding measurements. The violations of control invariants will be observed as the changes of weights in $\mathcal{G}(\mathtt{V},\mathtt{E},\mathtt{W})$. `PLC-Sleuth` uses a sliding window T to construct an online detector that monitors the weights of edges in E_y^u, by:

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

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) [28], to alarm operators if an anomaly is detected in Eq.(8). CUSUM is defined recursively as

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

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

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

and δ is a small positive constant, set as $\delta > |\nu_{t-1}|$ when the system operates normally, preventing S_t from increasing persistently. An alarm is triggered whenever S_t is larger than a pre-defined 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 intuitive 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 SEDS as an example: the forged u_2^a in Eq. (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

Table 1: Weight of SEDS's command edges obtained with different weighing schemes.

Metric		K2			LL			MDL			MI			PLC-Sleuth		
Score		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
	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**

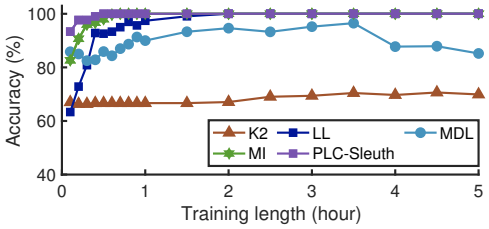


Figure 11: Constructing SEDS's control graph.

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.

5 Evaluation

We have evaluated PLC-Sleuth using both SEDS and a simulated Tennessee Eastman (TE) process, which is a representative benchmark of continuous chemical processes [29].

5.1 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: (i) Bayesian scoring metric K2 [30], (ii) Log-likelihood (LL) [31], (iii) Minimum Description Length (MDL) [32], and (iv) Mutual information (MI) in Eq.(4). Then, we evaluate PLC-Sleuth's intrusion detection/localization under two attack scenarios:

- **Attack-1:** a control command injection attack implemented by injecting time-varied values to the PIO table (including those of only slight deviations from normal commands) and tampering the control algorithm parameters.
- **Attack-2:** a cooperative stealthy attack atop of Attack-1 by replaying normal sensor measurements during attack.

5.2 Evaluation with SEDS

We first evaluate PLC-Sleuth using SEDS.

5.2.1 Constructing SEDS's Control Graph

We first examine if PLC-Sleuth is able to accurately construct SEDS's control graph.

Data Collection. We logged a 9.8h operation of SEDS using its SCADA system, during which a total number of 17 variables are collected at 2Hz, including 3 setpoint variables (i.e., $|S| = 3$), 11 sensor variables (i.e., $|Y| = 11$), and 3 command variables (i.e., $|U| = 3$). We then use these logs to evaluate PLC-Sleuth's construction of the control graph. Note SEDS has three decoupled single feedback loops, as shown in Fig. 7.

Accuracy of Graph Construction. With a training data of 2h, Table 1 compares the weights of $e_{y_j}^{u_k}$s (i.e., $w_{y_j}^{u_k}$s) of SEDS obtained with PLC-Sleuth and when using K2, LL, MDL, and MI as the weighting metric, showing:

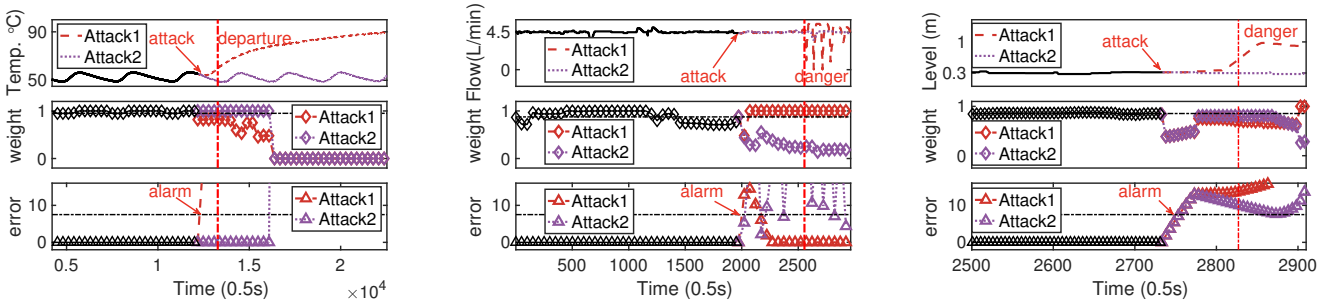
- PLC-Sleuth identifies SEDS's command edges e_y^us accurately and without repeated edges;
- K2 and MDL fail to identify the command edges correctly, due to the falsely matched edges of $\{e_{y_1}^{u_3}, e_{y_3}^{u_1}\}$ and $\{e_{y_1}^{u_2}, e_{y_2}^{u_1}\}$, respectively;
- Although LL and MI also match sensor nodes to their 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. 11, where the accuracy is averaged over 1,000 runs by randomly selecting the start time of training sequences from the 9.8h 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 0.5h.

5.2.2 Intrusion Detection/Localization with SEDS

We next evaluate PLC-Sleuth's intrusion detection/localization under two attack scenarios. The command injection attacks are launched in two ways, i.e., 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 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 10 times, i.e., a total number of $10 \times 3 = 30$ attacks are mounted to SEDS. Besides, by replaying the corresponding sensor's normal logs, we mount another $10 \times 3 = 30$ cooperative stealthy attacks.

Detection/Localization Accuracy. Table 2 lists the evaluation results, showing that PLC-Sleuth achieves an average of 98.33%/0.85‰ true/false positive (TP/FP) alarm rate, and localizes the forged command with 93.22% (i.e., 55 out of 59) positive predictive value (PPV).



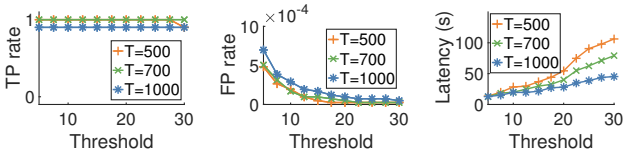
(a) Loop-1: tower temperature (b) Loop-2: cooling water flow (c) Loop-3: tower liquid level

Figure 12: Detecting the two attacks mounted at SEDS's three control loops.

Table 2: Evaluating PLC-Sleuth's intrusion detection/localization using SEDS.

Loop No.	*Attack-1*			*Attack-2*		
	TP	FP	PPV	TP	FP	PPV
#1	100%	1.3‰	90%	90%	1.3‰	66.7%
#2	100%	0.35‰	100%	100%	0.35‰	100%
#3	100%	0.26‰	100%	100%	0.26‰	100%

[1] Detection parameters: $T = 500$ for Loop-#1 and Loop-#2, $T = 4000$ for Loop-#3; $\mu = 7.5$; $\tau = 5$.



(a) TP rate (b) FP rate (c) latency

Figure 13: Applying PLC-Sleuth to SEDS with varying detection window and threshold.

- ***Attack-1.*** The injected forged command incurs instability to SEDS's distillation process, as shown with the red dashed line of the upper subplots in Fig. 12. PLC-Sleuth detects the weight changes of the monitored edges immediately after the attack, and thus triggers alarms. Among the 10 command injection attacks against the tower level loop, Figs. 13(a) and 13(b) plot 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. 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. 12. 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 1,800s, causing insensitive variation to variables' correlation. The PPV of localization for **Loop-**#1 is relatively low (when compared to Loop-2 and 3) for the same reason.

Detection Latency. We have also examined PLC-Sleuth's latency in detecting the PLC intrusions. Although a delay exists for an attack to physically disturb the system, the weight of control graph changes instantly when the attack is launched, as shown in the middle subplots of Fig. 12. As a result, PLC-Sleuth detects these attacks with a short latency, e.g., {50, 12.5, 10.5}s for attacks in Fig. 12(a), 12(b), and 12(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.

5.3 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 (see Fig. 21 in Appendix) [33]. 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 7 multiple feedback loops.

5.3.1 Constructing TE's Control Graph

Again, we first examine PLC-Sleuth's accuracy of constructing TE's control graph.

Data collection. Simulating the TE process with Matlab, we obtained a 72h training data logged at 1.8Hz. The (correctly) constructed control graph is shown in Fig. 14, in which a total number of 45 edges need to be identified.

Accuracy of Graph Construction. We have evaluated PLC-Sleuth with various volumes of training data in different running periods. Fig. 15 plots the results averaged over 1,000 runs.PLC-Sleuth's graph construction achieves a high accuracy of 95.76% even when being trained with only a short trace of 2h, which increases further with a larger volume of training data. The average accuracy reaches 98.11% with a 20h training data, i.e., PLC-Sleuth accurately identifies almost all of TE's 45 edges. Note that the construction errors mainly occur at transfer variables, because of the similarities

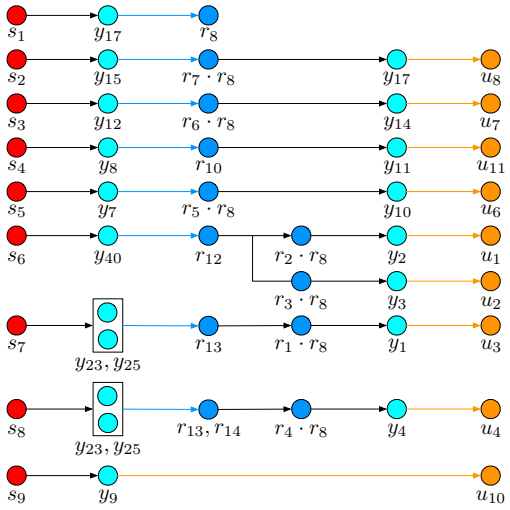


Figure 14: Control graph of the TE process.

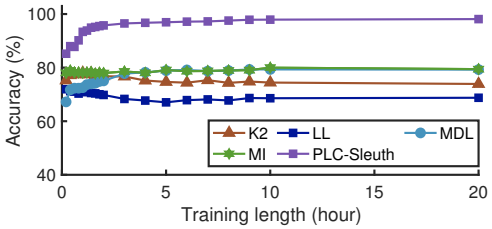


Figure 15: Constructing the TE process's control graph.

between transfer nodes and transmit nodes (e.g., mismatching r_2 to y_{40} in Fig. 14). Fig. 15 also plots the results of control graph construction when using different metrics as $w_{y_j}^{u_k}$, showing that PLC-Sleuth achieves the highest construction accuracy.

5.3.2 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 10 times in both scenarios. 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). Detailed detecting/localizing results of the $17 \times 10 \times 2 = 340$ attacks are listed in Table 3 of Appendix.

- ***Attack-1.*** The forged time series, which are generated by (i) adding random noise to normal output commands, and (ii) mixing constant values with random white noises to hide the attack, are injected to each of the 17 control loops. Fig. 16 plots the weight $w_{y_j}^{u_k}$ under the four example attacks. We can see that the weights of all the edges $e_{y_j}^{u_k}$ corresponding to the compromised control loops change due to the attack, making them detectable by PLC-Sleuth. Figs. 23(a) and 23(b) in Appendix present a visualization of the weight change. We further evaluate the impacts of detection window and threshold using the 10 attacks against the TE's reactor level control, as plotted in Fig. 17, 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.

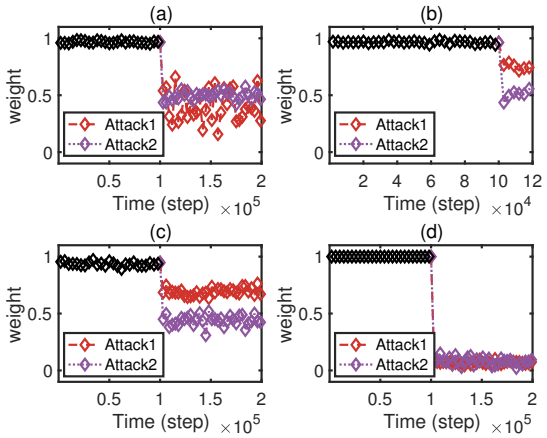


Figure 16: Command weights $W_{y_j}^{u_k}$ of four control loops under two attacks: (a) reactor temperature; (b) reactor level; (c) reactor pressure; (d) product quality.

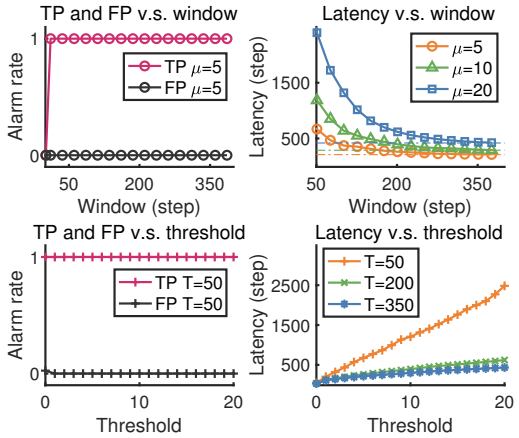
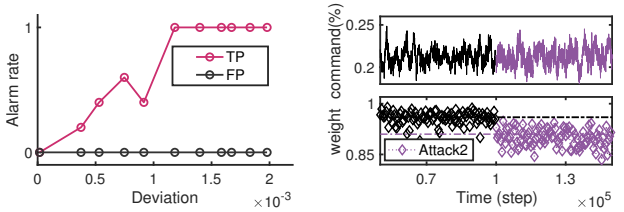


Figure 17: Impact of the detection window (T) and threshold (μ) on PLC-Sleuth's attack detection for the reactor level control in TE.

- ***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. 16 and visualized in Figs. 23(a) and 23(c) in Appendix. Note PLC-Sleuth detects Attack-2 against the reactor pressure control loop with {100%,0%} true/false positives, as long as the deviation from normal



(a) TP and FP v.s. deviation

(b) $w^{r_5}_{y_7}$ under a stealthy attack with a 0.16% deviation.

Figure 18: PLC-Sleuth's sensitivity to deviations from the normal commands for the reactor pressure control in TE.

commands (defined as $\sum|u^a(k)-u(k)|/\sum u(k)\times 100\%$) is larger than 0.12%, as shown in Fig. 18(a). Fig. 18(b) visualizes the weight change with a 0.16% deviation.

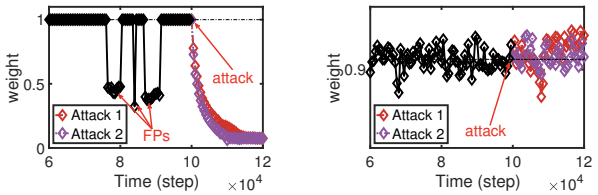
5.3.3 Tolerance to Inaccurate Control Graph

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 inner-loop fault, such as the edge connecting node y_{40} to r_2 in Fig. 14. For the control graph constructed by other learning metrics, falsely matched edges also include the pairs of nodes from different control loops, which we call the inter-loop fault, such as edge connecting node y_{11} to u_8 in Fig. 14. Fig. 19 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 inter-loop faults, however, is not applicable anymore.

6 Discussion

Stealthy sensor attack. Orthogonal to the targeted cooperative stealthy attacks of PLC-Sleuth, the stealthy sensor attacks have been investigated extensively [34–36], which can be detected effectively using process invariants defined using the correlations among sensors [28, 37–39].

Sophisticated Evasive Attacks. To evade PLC-Sleuth, sophisticated attackers may carefully forge sensor logs to make them consistent with expected correlations with the forged commands. However, it can be challenging to launch such an attack in practice. First, the attacker would need deep knowledge of not only pairs of correlated variables in the same control loop but also the control algorithm that governs their detailed relationships. Second, control loops are interdependent by the laws of physics, thus it is hard for the attacker to mimic sensors' behavior, as he needs to jointly model all the relevant control loops. These limitations make the further characterization of evasive cooperative stealthy attacks an interesting topic for future research.



(a) inner-loop fault

(b) inter-loop fault

Figure 19: Weights of the two types of the false matched edges under the two attacks scenarios.

7 Related work

PLC attacks. Systematic attacks against PLC have been reported. PLC firmware, providing an interface between user program and PLC input/output modules, has been attacked in [8, 11, 40]. The compromised firmware separates legitimate control rules from the physical process. A "ladder logic bombs", instantiated in IEC 61131-3 languages, was designed against a PLC user program to activate malicious behaviour under pre-defined conditions [41]. Besides, a dynamic payload generation against the target physical process has been proposed in [42] and enhanced by [6]. They generated the malicious payloads automatically.

Detecting PLC attacks. Against PLCs' attacks, many detection methods have been proposed based on the following methodologies: program analysis and system modeling. Program analysis based detections [43–46] monitor the control flow integrity (CFI) when controller is processing. However, the CFI checking is time-consuming, which renders it unsuitable for real-time detection in many situations [47]. System modeling methods that characterize variables' static invariants have been investigated [37, 48–52]. Further efforts have also been made to build dynamic models/invariants [39, 53–55]. In comparison, PLC-Sleuth also mines dynamic invariants but does so using significantly less a priori knowledge.

8 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 a control invariant of PLC, with the inherent and essential characteristics of control loops. Using the constructed control graph, PLC-Sleuth flags and localizes attacks when weights in $\mathcal{G}$ deviates from the norm. Evaluation results using SEDS and TE process show that PLC-Sleuth can construct control graph with high accuracy (100% with a log of 0.5h for SEDS and 98.11% with a log of 20h for TE process). PLC-Sleuth achieves detection true/false positives of $\{98.33\%, 0.85‰\}$ for SEDS, 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 SEDS and TE process, respectively.

Acknowledgments

This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFB0803501, the National Natural Science Foundation of China under Grant 61833015, U1911401, the SUTD-ZJU IDEA grant number SUTD-ZJU (VP) 201805, the Singapore MOE T1 grant number SUTDT12017004, and a startup grant of University of Colorado Denver.

References

[1] Keith Stouffer, Joe Falco, and Karen Scarfone. Guide to industrial control systems (ICS) security. *NIST special publication*, 800(82):16–16, 2011.

[2] Amy Babay, John Schultz, Thomas Tantillo, Samuel Beckley, Eamon Jordan, Kevin Ruddell, Kevin Jordan, and Yair Amir. Deploying Intrusion-Tolerant SCADA for the Power Grid. In *49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 328–335, 2019.

[3] Ariana Mirian, Zane Ma, David Adrian, Matthew Tischer, Thasphon Chuenchujit, Tim Yardley, Robin Berthier, Joshua Mason, Zakir Durumeric, J. Alex Halderman, and Michael Bailey. An Internet-wide view of ICS devices. In *14th Annual Conference on Privacy, Security and Trust (PST)*, pages 96–103, 2016.

[4] Qiang Li, Xuan Feng, Haining Wang, and Limin Sun. Understanding the usage of industrial control system devices on the internet. *IEEE Internet of Things Journal*, 5(3):2178–2189, 2018.

[5] Symantec Security Response. Dragonfly: Western energy sector targeted by sophisticated attack group. https://www.symantec.com/blogs/threat-intelligence/dragonfly-energy-sector-cyber-attacks, 2017.

[6] Anastasis Keliris and Michail Maniatakos. ICSREF: A framework for automated reverse engineering of industrial control systems binaries. In *Symposium on Network and Distributed System Security (NDSS)*, 2019.

[7] Stephen McLaughlin and Patrick McDaniel. SABOT: specification-based payload generation for programmable logic controllers. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, pages 439–449, 2012.

[8] Zachry Basnight, Jonathan Butts, Juan Lopez, and Thomas Dube. Firmware modification attacks on programmable logic controllers. *International Journal of Critical Infrastructure Protection*, 6(2):76 – 84, 2013.

[9] Symantec Security Response. After Triton, Will the Industrial Threat Landscape Ever be the Same? https://www.symantec.com/blogs/feature-stories/after-triton-will-industrial-threat-landscape-ever-be-same, 2018.

[10] Nicolas Falliere, Liam O Murchu, and Eric Chien. W32. stuxnet dossier. *White paper, Symantec Corp., Security Response.*, 5(6):29, 2011.

[11] Luis Garcia, Ferdinand Brasser, Mehmet Hazar Cintuglu, Ahmad-Reza Sadeghi, Osama A Mohammed, and Saman A Zonouz. Hey, My Malware Knows Physics! Attacking PLCs with Physical Model Aware Rootkit. In *Network and Distributed Systems Security (NDSS) Symposium*, 2017.

[12] Gene F Franklin, J David Powell, and Abbas Emami-Naeini. *Feedback control of dynamic systems*. Pearson London, 2015.

[13] Gary A Dunning. *Introduction to programmable logic controllers*. Cengage Learning, 2005.

[14] Henry Z Kister. What caused tower malfunctions in the last 50 years? *Chemical Engineering Research and Design*, 81(1):5–26, 2003.

[15] Richard Bellman, Irving Glicksberg, and Oliver Gross. On the "bang-bang" control problem. *Quarterly of Applied Mathematics*, 14(1):11–18, 1956.

[16] Karl Johan Åström, Tore Hägglund, and Karl J Astrom. *Advanced PID control*. ISA-The Instrumentation, Systems, and Automation Society, 2006.

[17] Hans Berger. *Automating with STEP7 in STL and SCL: programmable controllers Simatic S7-300/400*. Publicis Publishing, 2012.

[18] Pedro Albertos and Sala Antonio. *Multivariable control systems: an engineering approach*. Springer Science & Business Media, 2003.

[19] Yilin Mo and Bruno Sinopoli. Secure control against replay attacks. In *2009 47th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 911–918, 2009.

[20] Xue-Wen Chen, Gopalakrishna Anantha, and Xiaotong Lin. Improving Bayesian network structure learning with mutual information-based node ordering in the K2 algorithm. *IEEE Transactions on Knowledge and Data Engineering*, 20(5):628–640, 2008.

[21] Xiannian Fan and Changhe Yuan. An improved lower bound for bayesian network structure learning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 3526–3532, 2015.

[22] Pengfei Sun, Luis Garcia, and Saman Zonouz. Tell Me More Than Just Assembly! Reversing Cyber-physical Execution Semantics of Embedded IoT Controller Software Binaries. In *49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 349–361, 2019.

[23] Ralf Steuer, Jürgen Kurths, Carsten O Daub, Janko Weise, and Joachim Selbig. The mutual information: detecting and evaluating dependencies between variables. *Bioinformatics*, 18(suppl_2):S231–S240, 2002.

[24] Thomas McAvinew and Raymond Mulley. *Control System Documentation: Applying Symbols and Identification*. ISA, 2004.

[25] Sigurd Skogestad and Ian Postlethwaite. *Multivariable feedback control: analysis and design*. Wiley New York, 2007.

[26] Emerson report. Fisher 4320 Wireless Position Monitor. https://www.emerson.com/en-us/catalog/fisher-4320, 2009.

[27] SIEMENS report. SIEMENS SIPART PS2 (6DR5...) Electropneumatic positioners. http://www.lesman.com/unleashd/catalog/accessor/Siemens_SIPART-PS2/Siemens-SIPART-PS2-man-A5E03436620-AB-2017-01.pdf, 2017.

[28] David I Urbina, Jairo A Giraldo, Alvaro A Cardenas, Nils Ole Tippenhauer, Junia Valente, Mustafa Faisal, Justin Ruths, Richard Candell, and Henrik Sandberg. Limiting the impact of stealthy attacks on industrial control systems. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1092–1105, 2016.

[29] J.J. Downs and E.F. Vogel. A plant-wide industrial process control problem. *Computers & Chemical Engineering*, 17(3):245–255, 1993.

[30] Gregory F Cooper and Edward Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine learning*, 9(4):309–347, 1992.

[31] Remco Ronaldus Bouckaert. *Bayesian belief networks: from construction to inference*. PhD thesis, 1995.

[32] Wai Lam and Fahiem Bacchus. Learning Bayesian belief networks: An approach based on the MDL principle. *Computational intelligence*, 10(3):269–293, 1994.

[33] N. Lawrence Ricker. Decentralized control of the Tennessee Eastman Challenge Process. *Journal of Process Control*, 6(4):205–221, 1996.

[34] Yao Liu, Peng Ning, and Michael K. Reiter. False Data Injection Attacks against State Estimation in Electric Power Grids. In *Proceedings of the 16th ACM Conference on Computer and Communications Security*, page 21–32, 2009.

[35] Saurabh Amin, Xavier Litrico, Shankar Sastry, and Alexandre M Bayen. Cyber security of water SCADA systems—Part I: Analysis and experimentation of stealthy deception attacks. *IEEE Transactions on Control Systems Technology*, 21(5):1963–1970, 2012.

[36] Pritam Dash, Mehdi Karimibiuki, and Karthik Pattabiraman. Out of control: stealthy attacks against robotic vehicles protected by control-based techniques. In *Proceedings of the 35th Annual Computer Security Applications Conference*, pages 660–672, 2019.

[37] Yong Wang, Zhaoyan Xu, Jialong Zhang, Lei Xu, Haopei Wang, and Guofei Gu. Srid: State relation based intrusion detection for false data injection attacks in scada. In *European Symposium on Research in Computer Security*, pages 401–418, 2014.

[38] Wissam Aoudi, Mikel Iturbe, and Magnus Almgren. Truth Will Out: Departure-Based Process-Level Detection of Stealthy Attacks on Control Systems. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 817–831, 2018.

[39] Raul Quinonez, Jairo Giraldo, Luis Salazar, Erick Bauman, Alvaro Cardenas, and Lin Zhiqiang. SAVIOR: Securing Autonomous Vehicles with Robust Physical Invariants. In *29th USENIX Security Symposium (USENIX Security 20)*, 2020.

[40] Ali Abbasi and Majid Hashemi. Ghost in the PLC Designing an Undetectable Programmable Logic Controller Rootkit via Pin Control Attack. In *Black Hat Europe.*, 2016.

[41] Naman Govil, Anand Agrawal, and Nils Ole Tippenhauer. On ladder logic bombs in industrial control systems. In *Computer Security*, pages 110–126, 2018.

[42] Stephen McLaughlin. On dynamic malware payloads aimed at programmable logic controllers. In *Proceedings of the 6th USENIX conference on Hot topics in security*, 2011.

[43] Lucille McMinn and Jonathan Butts. A firmware verification tool for programmable logic controllers. In *International Conference on Critical Infrastructure Protection*, pages 59–69, 2012.

[44] Sebastian Biallas, Jörg Brauer, and Stefan Kowalewski. Arcade.PLC: A verification platform for programmable logic controllers. In *Proceedings of the 27th IEEE/ACM*

International Conference on Automated Software Engineering, pages 338–341, 2012.

[45] Stephen McLaughlin, Saman A Zonouz, Devin J Pohly, and Patrick D McDaniel. A Trusted Safety Verifier for Process Controller Code. In *Network and Distributed Systems Security (NDSS) Symposium*, 2014.

[46] Saman Zonouz, Julian Rrushi, and Stephen McLaughlin. Detecting industrial control malware using automated PLC code analytics. *IEEE Security Privacy*, 12(6):40–47, 2014.

[47] Irfan Ahmed, Sebastian Obermeier, Sneha Sudhakaran, and Vassil Roussev. Programmable Logic Controller Forensics. *IEEE Security Privacy*, 15(6):18–24, 2017.

[48] Robert Mitchell and Ray Chen. Behavior rule specification-based intrusion detection for safety critical medical cyber physical systems. *IEEE Transactions on Dependable and Secure Computing*, 12(1):16–30, 2014.

[49] Khurum Nazir Junejo and Jonathan Goh. Behaviour-based attack detection and classification in cyber physical systems using machine learning. In *Proceedings of the 2nd ACM International Workshop on Cyber-Physical System Security*, pages 34–43, 2016.

[50] Yuqi Chen, Christopher M. Poskitt, and Jun Sun. Learning from Mutants: Using Code Mutation to Learn and Monitor Invariants of a Cyber-Physical System. In *2018 IEEE Symposium on Security and Privacy (S&P)*, pages 240–252, 2018.

[51] Feng Cheng, Palleti Venkata, Reddy, Mathur Aditya, and Chana Deeph. A Systematic Framework to Generate Invariants for Anomaly Detection in Industrial Control Systems. In *Network and Distributed Systems Security (NDSS) Symposium*, 2019.

[52] Mu Zhang, James Moyne, Z Morley Mao, Chien-Ying Chen, Bin-Chou Kao, Yassine Qamsane, Yuru Shao, Yikai Lin, Elaine Shi, Sibin Mohan, et al. Towards Automated Safety Vetting of PLC Code in Real-World Plants. In *2019 IEEE Symposium on Security and Privacy (S&P)*, pages 522–538, 2019.

[53] Pinyao Guo, Hunmin Kim, Nurali Virani, Jun Xu, Minghui Zhu, and Peng Liu. RoboADS: Anomaly detection against sensor and actuator misbehaviors in mobile robots. In *48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 574–585, 2018.

[54] Hongjun Choi, Wen-Chuan Lee, Yousra Aafer, Fan Fei, Zhan Tu, Xiangyu Zhang, Dongyan Xu, and Xinyan Deng. Detecting Attacks Against Robotic Vehicles: A Control Invariant Approach. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 801–816, 2018.

[55] Hamid Reza Ghaeini, Matthew Chan, Raad Bahmani, Ferdinand Brasser, Luis Garcia, Jianying Zhou, Ahmad-Reza Sadeghi, Nils Ole Tippenhauer, and Saman Zonouz. PAtt: Physics-based Attestation of Control Systems. In *22nd International Symposium on Research in Attacks, Intrusions and Defenses (RAID)*, pages 165–180, 2019.

Appendix

Patterns of SCADA logs

SCADA logs present obvious three kinds of patterns, which correspond to the 3 defined node sets:

- Setpoint set S. Setpoint variables are of constant values, shown in Figs. 20(a)-20(c);
- Sensor set Y. Sensor readings are of consistent and small vibrations, shown in Figs. 20(d)-20(n).
- Command set U. Control commands are of continuous/smooth values, as shown in Figs. 20(o)-20(q).

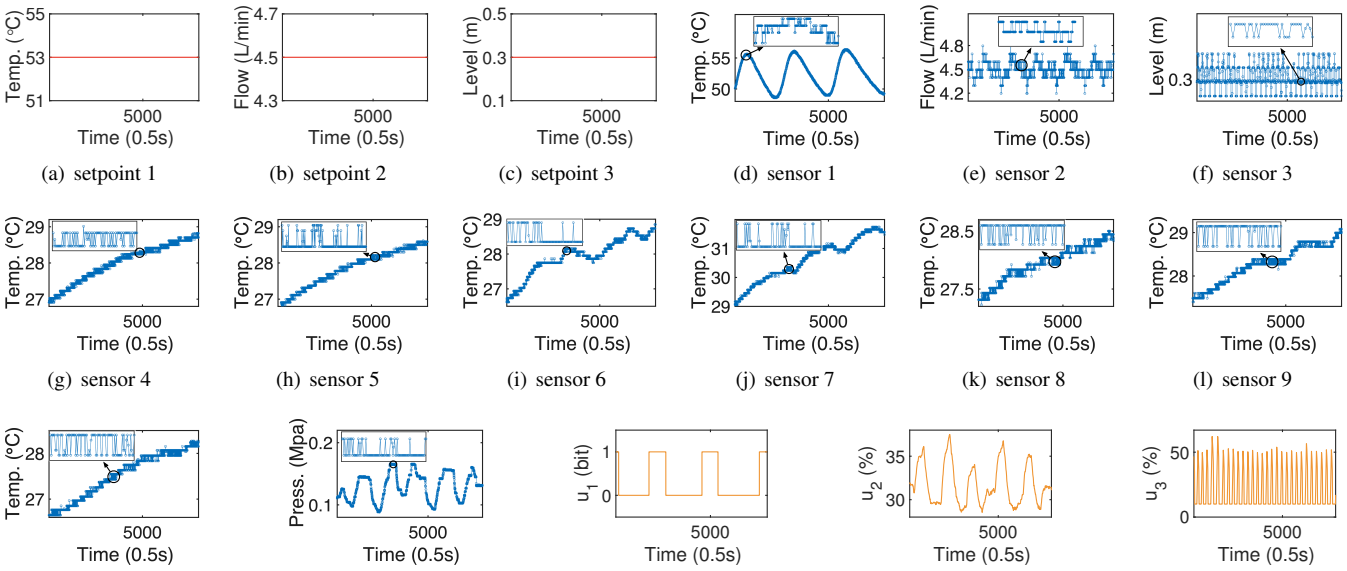
Evaluating `PLC-Sleuth` on TE

Tennessee Eastman (TE) process, as shown in Fig. 21, is widely used for evaluating threats and protection methods designed for cyber physical systems.

We deploy the command injection attack and the cooperative stealthy attack on the 17 control loops of TE. Fig. 22 shows the impacts of four example attacks.

As a typical control system, the control commands in TE have strong correlations with the concomitant sensor readings, shown as the black line in Fig. 23. This correlations can be quantified as command weights $w_{yj}^{u_k}$s. However, the two scenarios of attacks both damage the correlations and cause command weights' variation, shown as the red and purple line in Fig. 23.

By utilizing the variation of command weights, `PLC-Sleuth` detects attacks with $\{100\%/0\%\}$ true/false positives, and localizes the compromised control loops with 96.76% accuracy. Details of the detection/localization results are listed in Table 3.



(a) setpoint 1 (b) setpoint 2 (c) setpoint 3 (d) sensor 1 (e) sensor 2 (f) sensor 3

(g) sensor 4 (h) sensor 5 (i) sensor 6 (j) sensor 7 (k) sensor 8 (l) sensor 9

(m) sensor 10 (n) sensor 11 (o) command 1 (p) command 2 (q) command 3

Figure 20: Different patterns of setpoints, sensors, commands variables in SEDS.

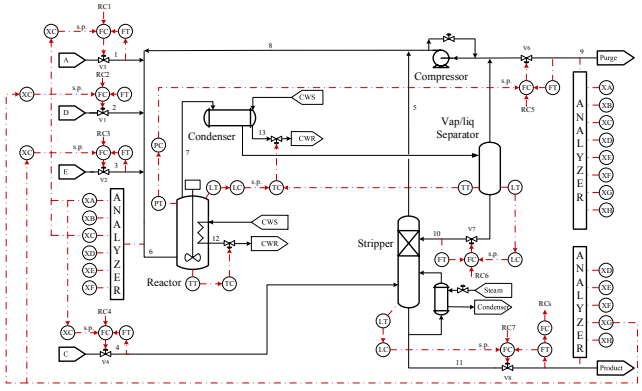


Figure 21: The Tennessee Eastman (TE) control process [33].

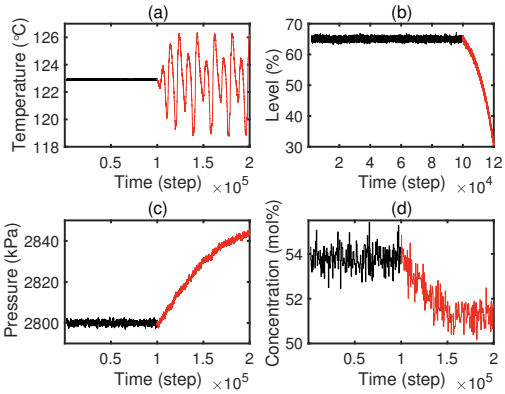
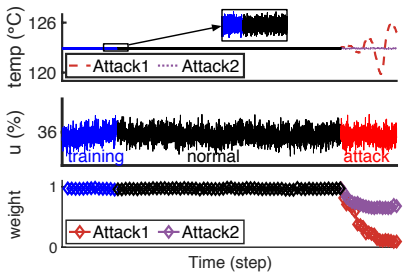
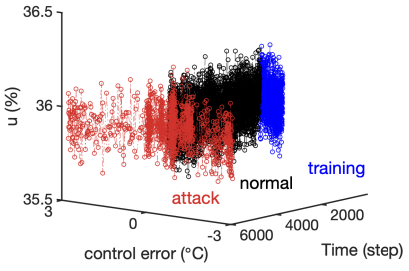


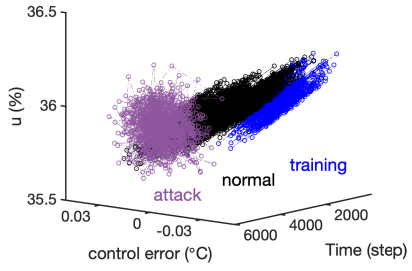
Figure 22: System evolution under command injection attacks: (a) unstable reactor temperature; (b) stopped reactor level; (c) dangerous reactor pressure; (d) degraded product quality.



(a) the raw measurements and commands in SCADA, and weight $w^{II}_{(y,u)}$ in detector



(b) change of correlated variables' joint distribution under a command injection attack



(c) change of correlated variables' joint distribution under a cooperative stealthy attack

Figure 23: A visualization of abnormal weight in TE's control graph under the two attacks against reactor temperature control.

Table 3: Evaluating PLC-Sleuth's attack detection and localization with the TE process.

Compromised Control loop Number	Controlled variable◇	Command variable◇	Detection				Localization	
			command injection		stealthy attack		command injection	stealthy attack
			TP rate	FP rate	TP rate	FP rate		
1	A feed rate	$xmv(3)$	100%	0	100%	0	100%	100%
2	D feed rate	$xmv(1)$	100%	0	100%	0	100%	100%
3	E feed rate	$xmv(2)$	100%	0	100%	0	100%	100%
4	C feed rate	$xmv(4)$	100%	0	100%	0	100%	50%
5	Purge rate	$xmv(6)$	100%	0	100%	0	100%	100%
6	Sep.liq.rate	$xmv(7)$	100%	0	100%	0	100%	100%
7	Strip.liq.rate	$xmv(8)$	100%	0	100%	0	100%	100%
8	Production rate	F_p	100%	0	100%	0	80%	80%
9	Strip.liq.level	Ratio in loop 7	100%	0	100%	0	100%	100%
10	Sep.liq.level	Ratio in loop 6	100%	0	100%	0	100%	100%
11	Reac.liq.level	Setpoint of loop 17	100%	0	100%	0	100%	100%
12	Reac.pres	Ratio in loop 5	100%	0	100%	0	100%	100%
13	Mol%G in stream 11	E_{adj}	100%	0	100%	0	100%	100%
14	y_A	Ratio in loop 1,r_1	100%	0	100%	0	100%	90%
15	y_{AC}	Sum of $r_1 + r_4^{'}$	100%	0	100%	0	100%	90%
16	Reac.temp	$xmv(10)$	100%	0	100%	0	100%	100%
17	Sep.temp	$xmv(11)$	100%	0	100%	0	100%	100%

◇ Controlled variables and command variables are defined in [29, 33].