Mismatched Control and Monitoring Frequencies: Vulnerability, Attack, and Mitigation

Zeyu Yang[1], Liang He[2], Peng Cheng[1], and Jiming Chen[1]

Abstract—Stealthy attacks manipulate the operation of Industrial Control Systems (ICSs) without being undetected, allowing persistent manipulation of system operation and thus the potential to cause destructive damage. This paper introduces a new vulnerability of ICS that can be exploited to mount stealthy attacks without requiring any domain knowledge. This vulnerability is caused by a common practice in system monitoring, i.e., the SCADA monitors ICS operation at a much lower frequency than system execution, causing a loss of precision when the SCADA tries to cross-validate the issued control commands using the collected sensory data. Exploiting this vulnerability, an attack called PLC-SAGE is designed to stealthily manipulate the system operation by identifying and injecting malicious control commands that will not be concluded as abnormal by the SCADA. This paper further discusses a preferred ICS engineering practice and an attestation strategy to mitigate the above vulnerability and protect ICS from PLC-SAGE. Both PLC-SAGE and the proposed mitigations have been experimentally validated on two ICS platforms.

Index Terms—Programmable Logic Controllers, Stealthy Attacks, Mismatched Frequencies

1 Introduction

Programmable Logic Controllers (PLCs) are commonly used to operate Industrial Control Systems (ICSs) and report the states of the concomitant physical processes to the Supervisory Control And Data Acquisition (SCADA) system. Taking the ethanol distillation system in Fig. 1 as an example, PLCs collect sensor readings from the input memory (e.g., $\mathtt{I}^2_{94}$), drive actuators via the output memory (e.g., $\mathtt{O}^2_{74}$), and report the corresponding system states (e.g., *water_flow* and *cooling_valve_opening*) to the SCADA.

However, PLCs can be remotely and illegitimately accessed due to the insecure industrial network [1], [2], [3] and weak authentication of communication protocols [4], [5], [6]. Once connected to a victim PLC, adversaries can tamper with the memory thereof to manipulate the concomitant physical process [7], [8], [9]. Arbitrary manipulation of PLC memory could be easily and swiftly detected by cross-validating the control commands and sensor readings collected by SCADA [10], [11], [12], [13], [14], [15]. For example, an attack was mounted on a water treatment system in Florida in 2021, which brutally modified the PLC memory specifying the NaOH content from 100ppm to 11,100ppm, and was detected the moment it was launched [8]. In comparison, attacks that tamper with PLC memory while keeping themselves undetected, referred to as *stealthy attacks*, can persistently manipulate system operation and potentially cause destructive damage. For example, Stuxnet modified the PLC memory specifying the converter's frequency and manipulated their operation for several months without being detected, until hundreds of devices in nuclear facilities were damaged [9].

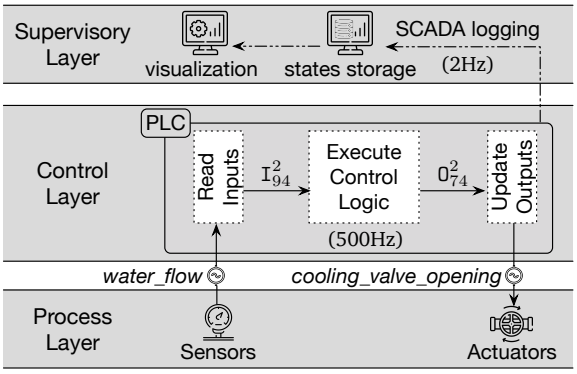


Fig. 1. The role of PLC in an Ethanol Distillation System.

Stealthy attacks are traditionally mounted by replaying the collected (and normal) SCADA data to cover the manipulated system operation from being detected [16], which is, however, no longer feasible when confronted with state-aware anomaly detectors [10], [17], [18]. According to [19], an attack can be claimed as stealthy only if the SCADA observes identical system behaviors before and after the attack is launched. As a result, recent stealthy attacks usually require extensive domain knowledge of the targeted ICS, such as the dynamics of the concomitant physical processes [20], [21], [22], [23].

In this paper, we present PLC-SAGE, an autonomous Stealthy mALicious payload GENeration attack against PLCs, to stealthily manipulate the physical process without requiring any domain knowledge. PLC-SAGE is inspired by a newly discovered system vulnerability caused by a common practice in system monitoring, that is, the SCADA monitors ICS operation at a much lower frequency than PLC execution [24], [25], [26]. As an example, the minimum data ac-

This work was supported in part by the National Natural Science Foundation of China under Grant 62303411 and 62293510/62293511. (Corresponding author: Peng Cheng.)

[1]*Zeyu Yang, 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; lunarheart@zju.edu.cn; jmchen@ieee.org).*

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

quisition period of WinCC, a widely-used SCADA software, is 500ms [26], whereas the PLC execution cycle typically spans from 1ms to 100ms and is limited within a watchdog time, such as 150ms by default for Siemens PLCs [27]. These mismatched frequencies cause a loss of precision when the SCADA tries to restore the control commands using the collected sensory data — cross-checking the restored and empirically collected control commands is a widely adopted principle in anomaly detection [10], [28].

Exploiting this vulnerability, PLC-SAGE manipulates the system operation to the degree that the SCADA cannot differentiate if the deviation between the restored and collected commands is caused due to mismatched frequencies or malicious manipulation. Specifically, PLC-SAGE adopts a 4-step attack procedure: (1) identify and collect the sensor readings and control commands from PLC memory, (2) restore the control logic based on the collected system states, (3) generate stealthy (and malicious) payloads based on the restored control logic, and (4) write the malicious payloads to PLC memory to manipulate system operation without being detected. This paper also discusses a preferred practice in implementing ICS to mitigate the vulnerability caused by the mismatched frequencies in system monitoring and execution, and an attestation strategy to void PLC-SAGE by randomizing the SCADA monitoring frequency without affecting system operation.

We have implemented and evaluated PLC-SAGE on two ICS platforms: an Ethanol Distillation System (EDS) and the Tennessee-Eastman System (TES). The experimental results show PLC-SAGE to identify sensor readings and control commands from PLC memory with an accuracy of 100%. Using these information, PLC-SAGE is stealthily mounted to both platforms, and successfully destabilized and damaged the concomitant physical processes. We have also experimentally validated the effectiveness of the proposed mitigation strategies in protecting ICS from PLC-SAGE.

In summary, this paper makes the following major contributions.

- Uncover a new ICS vulnerability caused by a common practice of monitoring system operation at a much lower frequency than system control.
- Design PLC-SAGE, a novel stealthy attack that autonomously generates and injects malicious payloads to manipulate system operation without being detected or requiring any domain knowledge of the targeted ICS. The source code of PLC-SAGE is openly available at: https://github.com/zeyuid/PLCSage.
- Propose two mitigation strategies to address the newly discovered vulnerability and protect ICSs from PLC-SAGE.

2 Related Work

The increasing threat of cyber attacks that have the potential of causing physical damage, referred to as *cyber-physical attacks*, magnifies the concerns of ICS operators on the safety of system operations. To address this, researchers have devoted much effort to the design of attack detectors for ICS and the analysis of their vulnerabilities.

By utilizing the legitimate behaviors of ICS in network traffic [29], program execution [30], SCADA data [28], many

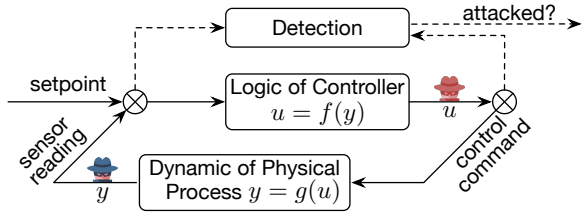
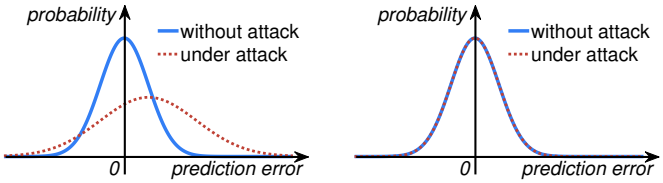


Fig. 2. Building attack detectors on the dynamics of control loops.

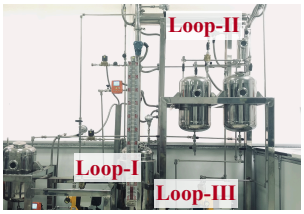


(a) detectable attacks (b) stealthy attacks

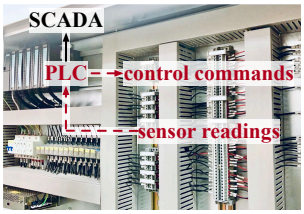
Fig. 3. The exampled probability density functions of prediction error without/under attacks.

attack detection strategies have been proposed. In particular, thanks to their non-intrusive and real-time detection capabilities, attack detectors based on SCADA data have been widely deployed in real-world ICSs [31]. Grounded on the dynamics of ICS control loops, including the dynamics of physical process and logic of controller as shown in Fig. 2, researchers have proposed numerous attack detection methods based on the interactions between sensor readings y and control commands u in SCADA [10], [15], [16], [32], [33], [34], [35]. Given the dynamics of physical processes, that is, the law of physics $y = g(u)$, the sensor readings y can be predicted (denoted as $\hat{y}^-$) based on the corresponding control commands [10], [32], [33]. On the other hand, given the control logic of controllers, that is, the calculation of $u = f(y)$, the control commands u can be predicted (denoted as $\hat{u}^-$) based on the corresponding sensor readings [15], [34]. Based on the prediction error $y - \hat{y}^-$ (or $u - \hat{u}^-$), SCADA can detect attacks based on the outliers in the prediction error. As illustrated in Fig. 3(a), the common detection metrics are designed based on the deviation of the probability distribution of prediction errors.

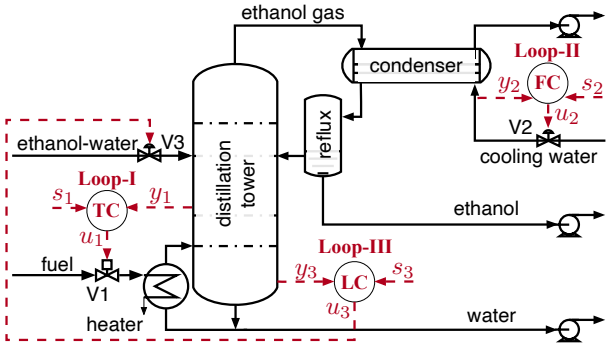
From the perspective of white-hat attackers, researchers have also explored the feasibility of voiding the above data-driven attack detectors, that is, the risk of damaging the physical process without being detected by the SCADA [18], [19], [20], [21], [22]. Clearly, SCADA will not be able to detect attacks if the detection metric is indistinguishable from their counterparts when the ICS operates normally, as illustrated in Fig. 3(b). Following this idea, researchers have tried to hide anomalies in the legitimate prediction errors by exploiting the normal operation of ICSs (see Fig. 2). For example, atop the dynamics of physical processes $y = g(u)$, adversaries can manipulate sensor readings y to y^a, making $y^a - \hat{y}^-$ belongs to the probability distribution of $y - \hat{y}^-$ and thus the manipulation not detectable [18], [19], [20], [21], [22]. However, due to the complexity of physical processes (e.g., structure, materials), the accurate dynamics of $y = g(u)$ is difficult to obtain. Different from existing stealthy attacks, this paper presents PLC-SAGE, a novel stealthy attack that leverages a newly discovered vulnerability of ICSs: the mismatched frequencies in PLC execution and



(a) testbed



(b) network



(c) process flow diagram

Fig. 4. The Ethanol Distillation System (EDS) platform.

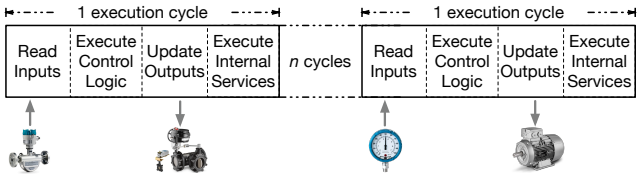


Fig. 5. The typical cyclic procedure of PLC execution.

SCADA logging. PLC-SAGE does not require any domain knowledge of the target ICS.

3 Preliminaries

Below we present the necessary background of ICSs using our platform of Ethanol Distillation System (EDS) as an example (see Fig. 4). EDS is a scaled-down but fully operational distillation process separating ethanol from the ethanol-water mixture with 90% purity. Specifically, using a Siemens S7-315 PLC, EDS realizes two basic operations of the distillation process — tower boiling and condenser cooling — with the following three control loops.

- *Loop-I (Heating Mixture)* maintains EDS work at a pre-defined temperature (e.g., 78°C), at which the ethanol is purified with a high efficiency.
- *Loop-II (Condensing Ethanol Gas)* controls the flow of cooling water at a pre-defined speed (e.g., $5L/min$) to condense ethanol from the gas to liquid, which will then be refluxed to the tower for concentration.
- *Loop-III (Transferring Mixture)* maintains the ethanol-water mixture at a pre-defined level in the tower (e.g., $0.1m$) to prevent premature flood.

Please see Appendix A in the Supplemental Material for more details of EDS.

3.1 PLC Basics

Fig. 5 depicts PLCs' typical cyclic procedure.

- *Read Inputs.* At the beginning of each execution cycle, PLCs read sensor readings and update the corresponding Process-Image Input (PII) memory. For example, the PLC updates the PII memory of $\mathtt{I}^2_{94}$ using $y_2(k)$, i.e., the sensor reading of *water_flow* in the kth cycle.
- *Execute Control Logic.* Based on the updated PII memory, PLCs execute the embedded control logic to generate corresponding control commands. For example, taking the setpiont value $s_2(k)$ and the $y_2(k)$ in $\mathtt{I}^2_{94}$ as inputs, the PLC generates the command $u_2(k)$ of *cooling_valve_opening* as

$$u_2(k) = K_p e_2(k) + K_i \sum_{j=0}^{k} e_2(j), \tag{1}$$

where $e_2 = s_2 - y_2$ denotes the control error, and $\{K_p, K_i\}$ are the specified PLC parameters.

- *Update Outputs.* PLCs store the above-generated control commands in its Process-Image Output (PIO) memory, and send these commands to the corresponding actuators at the end of each execution cycle. For example, the generated $u_2(k)$ is written to the PIO memory of $\mathtt{O}^2_{74}$ and sent to the cooling valve V2.
- *Execute Internal Services.* After completing all the above procedures, PLCs execute the internal services, such as reporting its PII/PIO memory states (including sensor readings and control commands) to the SCADA at a pre-defined frequency.

3.2 SCADA Basics

SCADA is a software system that supervises and controls the operation of physical processes. By cyclically acquiring the states of PLC memory (e.g., the PII memory), SCADA records and visualizes the real-time states of physical processes (e.g., sensor readings and control commands). Due to the storage limitations and data analysis requirements of SCADA, the data collection is usually configured at a lower frequency than system execution [24], [25], [26], such as up to 2Hz in Siemens WinCC software [26]. By sending messages using specific functions, SCADA can also overwrite the memory of the connected PLCs to set the corresponding control parameters, such as the $\{K_p, K_i\}$ in Eq. (1). SCADA may also deploy intrusion detection systems based on the recorded operation data of physical processes.

3.3 Attack Model

According to the widely-adopted ISA-95 architecture of ICS network [36], one PLC is usually connected to multiple workstations (e.g., SCADA and engineering workstation) at the supervisory layer. We consider adversaries who aim to damage the physical process without being detected at the supervisory layer, as illustrated in Fig. 6. Attackers achieve this goal by intruding into the supervisory network and compromising the targeted PLCs with the following abilities commonly observed in real life.

- *Eavesdropping on SCADA Logging Traffic.* The ICS network is insecure due to the lack of preventive measures such as firewalls, virtual private networks, and strong authentication [8], [37], [38], [39], particularly with the recently

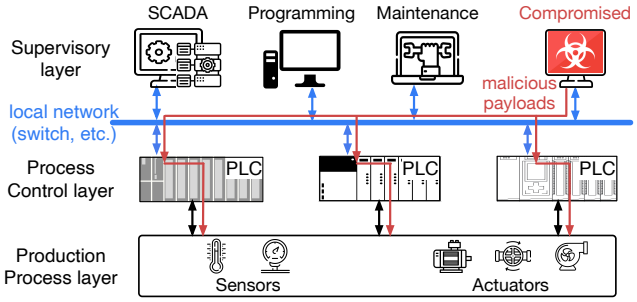


Fig. 6. Attack scenario: mounting stealthy attacks against PLCs at the supervisory layer of ICS network.

exposed malware [40], [41], [42], making the SCADA logging traffic thereof illegitimately accessible [43], [44], [45].

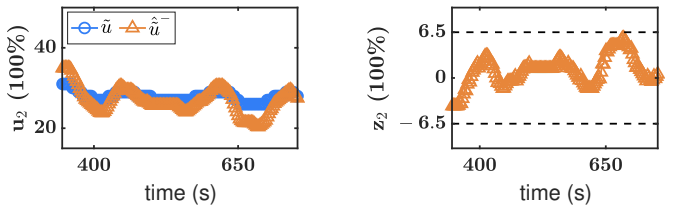
- *Parsing Network Traffic.* Besides the public PLC protocols (e.g., Modbus [46]), the proprietary PLC protocols have also been reverse-engineered, including the widely used Siemens S7Comm [47], [48] and Rockwell PCCC [49], [50]. The network traffic thus can be parsed according to the format of protocols.
- *Reading and Writing PLC Memory.* Most commercial PLCs support the online monitoring and debugging of memory but use insecure communication protocols [4], [5], [6], e.g., the Rockwell PCCC protocol leaks the authentication password to the attackers [4], allowing the adversaries to read and manipulate the PLC states.

Empirical Corroboration. We have experimentally validated the above adversary abilities on EDS. Similar to BlackEnergy [51], by sending from the public Internet a spear-phishing email, `PLC-SAGE` was propagated to a computer in EDS's supervisory layer. Also, we have corroborated the feasibility of manipulating the memory of PLCs from Siemens and Rockwell Automation [52], [53], which collectively account for 53% of the PLC market [54].

4 Motivational Example

A common practice of SCADA monitoring is to acquire PLC memory at a much lower frequency than PLC execution [24], [25], [26]. This common practice causes a large amount of process states unrecorded by the SCADA, preventing existing data-driven attack detectors from accurately restoring system states.

Let us take the condensing loop (Loop-II) of EDS as an example, where the PLC operates at 500Hz and the SCADA monitors at 2Hz. The command record of *cooling_valve_opening* $\tilde{u}_2$ is validated by re-executing the control logic of Eq. (1) with records of *water_flow* $\tilde{y}_2$ and setpoint $\tilde{s}_2$ [34], [55], as shown in Fig. 7(a). By comparing $\tilde{u}_2$ with the thus-obtained command prediction $\hat{\tilde{u}}_2^-$, SCADA can identify anomalies if the corresponding prediction error $z_2 = \tilde{u}_2 - \hat{\tilde{u}}_2^-$ exceeds a normal threshold [10], [21]. However, the mismatched frequencies prevent the SCADA from fully restoring the control commands, even with the complete knowledge of the control logic as in this example: as observed in Fig. 7(b), the command prediction $\hat{\tilde{u}}_2^-$ differs from the recorded $\tilde{u}_2$ obviously.



(a) command prediction (b) command prediction error

Fig. 7. Key observation: mismatched frequencies in system monitoring and execution cause imperfect prediction of ethanol gas condensing commands (i.e., u_2) in EDS even using the same control program.

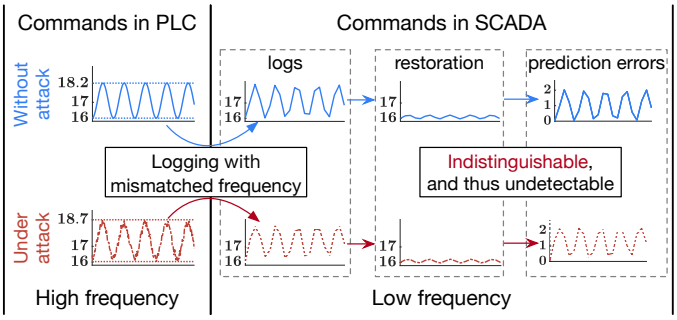


Fig. 8. The mismatched frequencies can be exploited as vulnerability to mount stealthy attacks.

5 Exploiting Mismatched Frequencies as a Vulnerability

The mismatched frequencies can be exploited by adversaries as a vulnerability to mount stealthy attacks. The commands logged at a low frequency will miss a large amount of process states, degrading the precision in restoring control commands (see the blue curves in Fig. 8). As a result, there is a chance that even if the commands are manipulated, the logged and concomitantly restored commands may still behave normally (see the red dash curves in Fig. 8), making the manipulation undetectable. This section theoretically derives the precision loss when restoring control commands using the intermittently logged PLC states, laying a foundation for the stealthy attack designed in Sec. 6.

5.1 Mismatched Frequencies Cause Deviation between Restored and Executed Control Logic

The mismatched frequencies in system monitoring and execution cause certain system states unrecorded by the SCADA. As a consequence, the control logic — i.e., the relationship between sensor readings and the concomitant issued control commands — restored by the SCADA will differ from the actual PLC execution. We first theoretically describe the relationship between this discrepancy and the mismatched frequencies.

Linear control logic, including the classic Proportional-Integral-Derivative (PID) logic [56], is widely deployed in off-the-shelf commercial PLCs, which generates control commands according to

$$x(k+1) = Ax(k) + Be(k), \tag{2}$$

$$u(k) = Cx(k) + De(k), \tag{3}$$

where k denotes the PLC execution cycle, $\{A, B, C, D\}$ specify the PLC control logic, $e(k) = s(k) - y(k)$ denotes the

control error between the real-time measured sensor reading $y(k)$ and the defined working state $s(k)$, $u(k)$ denotes the concomitantly issued control command, and $x(k)$ denotes the intermediate state of PLC in generating $u(k)$. For example, the control logic in Eq. (1) can be translated to Eqs. (2)-(3) with $A = 1$, $B = 1$, $C = K_i$ and $D = K_p + K_i$.

Considering the fact that a large amount of PLC states are not recorded by SCADA, we introduce model deviation $\omega(\tilde{k})$ to capture the error in restoring control logic using intermittently recorded PLC states, and formulate the restored control logic in SCADA as

$$\tilde{x}(\tilde{k}+1) = A\tilde{x}(\tilde{k}) + B\tilde{e}(\tilde{k}) + \omega(\tilde{k}), \quad (4)$$

$$\tilde{u}(\tilde{k}) = C\tilde{x}(\tilde{k}) + D\tilde{e}(\tilde{k}), \quad (5)$$

where $\tilde{k}$ denotes the SCADA logging cycle, $\tilde{e}(\tilde{k}) = s(k) - \tilde{y}(\tilde{k})$ denotes the control error between the logged sensor reading $\tilde{y}(\tilde{k})$ and expected working state $s(k)$, $\tilde{u}(\tilde{k})$ denotes the concomitantly logged control command, and $\tilde{x}(\tilde{k})$ denotes the intermediate state of SCADA in validating $\tilde{u}(\tilde{k})$.

The model deviation $\omega(\tilde{k})$ consists of the unrecorded control errors $\boldsymbol{e}$ and intermediate states $\boldsymbol{x}$ during the logging interval of $[\tilde{k}, \tilde{k}+1)$. We formally describe their relationships using the following theorem.

Theorem 5.1. *Given the frequency ratio P between PLC execution and SCADA monitoring, the model deviation $\omega(\tilde{k})$ satisfies:*

$$\omega(\tilde{k}) = (A^P - A)x(k) + \sum_{i=1}^{P} A^{i-1}Be(k+P-i) - Be(k). \quad (6)$$

Proof. Eqs. (2)-(3) executed by the PLC and Eqs. (4)-(5) restored by the SCADA represent the same control logic. To ensure their equivalence, the values of $\{\boldsymbol{x}, \boldsymbol{e}\}$ in PLC and the corresponding values of $\{\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{e}}\}$ in SCADA must satisfy $\tilde{x}(\tilde{k}) = x(k)$, $\tilde{x}(\tilde{k}+1) = x(k+P)$, and $\tilde{e}(\tilde{k}) = e(k)$, where P defines the frequency ratio between PLC execution and SCADA monitoring. When executing Eq. (2) in PLC, the state x evolves as:

$$\begin{aligned} x(k+P) &= Ax(k+P-1) + Be(k+P-1) \\ &= A^2x(k+P-2) + \sum_{i=1}^{2} A^{i-1}Be(k+P-i) \\ &\cdots \\ &= A^Px(k) + \sum_{i=1}^{P} A^{i-1}Be(k+P-i). \end{aligned}$$

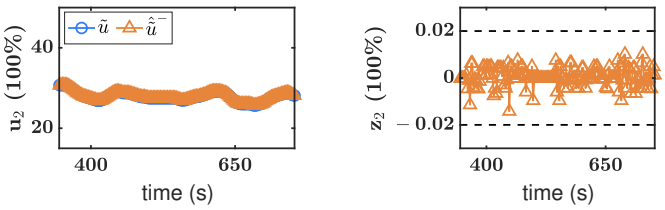
At the same time, the logged state $\tilde{x}$ by SCADA evolves according to Eq. (4) as:

$$\tilde{x}(\tilde{k}+1) = A\tilde{x}(\tilde{k}) + B\tilde{e}(\tilde{k}) + \omega(\tilde{k}).$$

Combining the requirements of $\tilde{x}(\tilde{k}+1) = x(k+P)$, $\tilde{x}(\tilde{k}) = x(k)$, and $\tilde{e}(\tilde{k}) = e(k)$, the model deviation $\omega(\tilde{k})$ satisfies Eq. (6). □

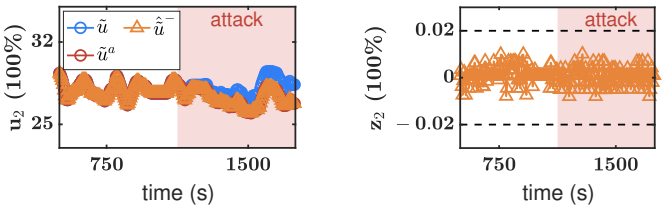
Considering the feature of $A^P = A$, which holds in most linear control logic (including P, PI, PD, PID controller as proved in Appendix B in the Supplemental Material), the model deviation in Eq. 6 can be further simplified as:

$$\omega(\tilde{k}) = \sum_{i=1}^{P} A^{i-1}Be(k+P-i) - Be(k). \quad (7)$$



(a) command prediction (b) command prediction error

Fig. 9. Predicting ethanol gas condensing commands (i.e., u_2) in EDS using the restored control logic.



(a) payload generation (b) command prediction error

Fig. 10. Generating stealthy payloads against the ethanol gas condensing on EDS.

Note that sensor reading $y(k)$ is maintained around the setpoint $s(k)$ during the steady period of ICSs, making the corresponding control error $e(k)$ follows the same distribution as the measurement noise in $y(k)$, which is unchanged and usually conforms to a Gaussian distribution [57], [58]. The model deviation $\omega(\tilde{k})$ thus follows an unchanged Gaussian distribution according to Eq. (7).

5.2 Mismatched Frequencies Cause Deviation between Restored and Collected Control Commands

By executing the restored control logic (i.e., Eqs. (4)-(5)) with the collected sensor readings as input, SCADA can restore the concomitant control commands but with precision loss. According to the theorems of optimal filtering [59], the precision loss in control command prediction can be minimized using the system model as described by Eqs. (4)-(5). In the following, we present a Minimum Mean-Squared Error (MMSE)-based command prediction algorithm to restore control commands with the minimum precision loss, establishing an upper bound of the prediction error below which the SCADA cannot differentiate if the deviation is caused by the mismatched frequencies or the manipulated control commands. Adversaries can thus hide the malicious manipulation from being detected by mimicking such minimal precision loss.

Taking the logged control errors $\{\tilde{e}(\tilde{k}), \tilde{e}(\tilde{k}+1)\}$ and control command $\tilde{u}(\tilde{k})$ as inputs, the MMSE command prediction $\tilde{u}(\tilde{k}+1)$ is recursively derived as:

a) The *a posteriori* state estimation. With the logged command $\tilde{u}(\tilde{k})$ and control error $\tilde{e}(\tilde{k})$, the *a posteriori* state estimation $\hat{\tilde{x}}(\tilde{k})$ is obtained as

$$\hat{\tilde{x}}(\tilde{k}) = C^{-1}[\tilde{u}(\tilde{k}) - D\tilde{e}(\tilde{k})], \quad (8)$$

which has zero estimation error because the command $\tilde{u}(\tilde{k})$ has no model deviation in Eq. (5).

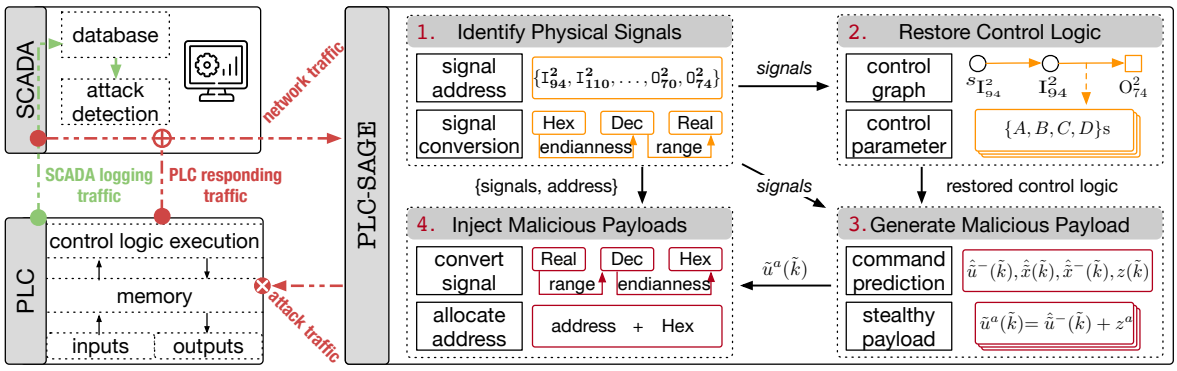


Fig. 11. Overview of PLC-SAGE.

b) The *a priori* state estimation. Using the estimated $\hat{\tilde{x}}(\tilde{k})$, the *a priori* state estimation $\hat{\tilde{x}}^-(\tilde{k}+1)$ is generated as

$$\hat{\tilde{x}}^-(\tilde{k}+1) = A\hat{\tilde{x}}(\tilde{k}) + B\tilde{e}(\tilde{k}). \tag{9}$$

The *a priori* state estimation error follows the same distribution as the model deviation $\omega(\tilde{k})$.

c) The command prediction. Using the estimated $\hat{\tilde{x}}^-(\tilde{k}+1)$, the command $\tilde{u}(\tilde{k}+1)$ is restored as

$$\hat{\tilde{u}}^-(\tilde{k}+1) = C\hat{\tilde{x}}^-(\tilde{k}+1) + D\tilde{e}(\tilde{k}+1), \tag{10}$$

which has the minimum mean-squared prediction error.

The optimality of the above MMSE command prediction can be proved with a similar approach as for the optimal filtering [59] and is omitted here due to the space limit. Given the collected command $\tilde{u}(\tilde{k})$ in SCADA, the prediction error of $\hat{\tilde{u}}^-(\tilde{k})$ is obtained as

$$z(\tilde{k}) = \tilde{u}(\tilde{k}) - \hat{\tilde{u}}^-(\tilde{k}). \tag{11}$$

Taking the ethanol gas condensing (Loop-II) in EDS as an example, the MMSE prediction of $\hat{u}_2$ is shown in Fig. 9(a), which significantly outperforms the prediction by re-executing the PLC control logic (as shown in Fig. 7(a)).

Combining Eq. (11) with Eqs. (8)-(10), we can see the prediction error $z(\tilde{k})$ is linearly related to the model deviation $\omega(\tilde{k})$. Considering the unchanged distribution of $\omega(\tilde{k})$ (as observed in Eq. 7), the corresponding prediction error $z(\tilde{k})$ also has an unchanged distribution, denoted as $\mathcal{Z}$, as shown in Fig. 9(b).

5.3 Generate Stealthy and Malicious Payloads based on the Deviations Caused by Mismatched Frequencies

With the identified distribution $\mathcal{Z}$ of prediction error, adversaries can generate the malicious payload $\tilde{u}^a$ that damages physical processes without being detected by making the resultant prediction error distribution conform to $\mathcal{Z}$. Specifically, the malicious payload at each step $\tilde{k}$ is generated as

$$\tilde{u}^a(\tilde{k}) = \hat{\tilde{u}}^-(\tilde{k}) + z^a(\tilde{k}), \tag{12}$$

where the command prediction $\hat{\tilde{u}}^-(\tilde{k})$ is generated according to Eqs. (8)-(10), and the command manipulation $z^a(\tilde{k})$ follows the identified $\mathcal{Z}$. As an example, the malicious payloads against the ethanol gas condensing in EDS are generated as shown in Fig. 10(a), where the payloads $\tilde{\boldsymbol{u}}_2^a$ deviate from the legitimate command $\tilde{\boldsymbol{u}}_2$ gradually. At the same time, the control commands restored by the SCADA are similar to the malicious payloads (see $\hat{\tilde{\boldsymbol{u}}}_2^-$ and $\tilde{\boldsymbol{u}}_2^a$ in Fig. 10(a)), preventing the SCADA from detecting the attack. As observed in Fig. 10(b), the prediction error behaves identically with the legitimate and the malicious control commands, making the manipulation/attack stealthy.

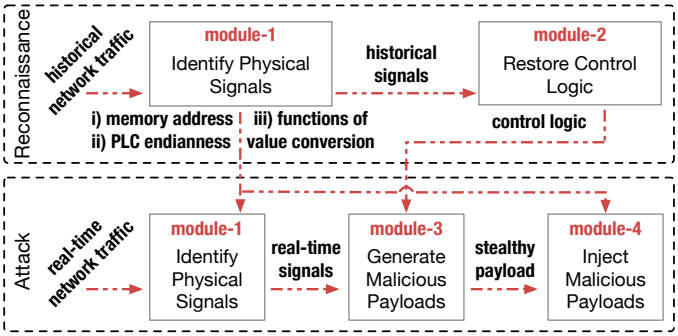


Fig. 12. The workflow of PLC-SAGE to mount stealthy attacks against the PLC.

6 Stealthy Malicious Payload Generation Attack against PLCs

From Sec. 5, we can see that stealthy attacks can be mounted once the adversary has restored the control logic (i.e., Eqs. (4)-(5)) and obtained the real-time physical signals (i.e., sensor readings $\tilde{\boldsymbol{y}}$ and control commands $\tilde{\boldsymbol{u}}$). Following this idea, we design PLC-SAGE to exploit the above-discussed mismatched frequencies to mount stealthy attacks on PLCs. As depicted in Fig. 11, PLC-SAGE consists of four functional modules, which cooperate following an autonomous 2-fold procedure as depicted in Fig. 12 — reconnaissance and attack — without requiring any *a priori* knowledge of targeted ICSs. During the reconnaissance phase, taking the historical PLC responding traffic to memory acquisition and the SCADA monitoring traffic as inputs, PLC-SAGE identifies physical signals in the PLC memory (including their memory addresses, PLC endianness, and value conversion functions), and restores the control logic among the thus-identified physical signals. During the attack phase, atop the above identification results and taking the real-time responding traffic to memory acquisition as input, PLC-SAGE identifies the real-time system states, generates stealthy malicious payloads against each control command,

TABLE 1
The nomenclature of symbols used when explaining the identification of physical signals in Sec. 6.1.

	Symbol	Definition
Paras.	w	byte size of signals
	μ	number of constant insignificant bits
Variables	$b(k)$	value of the bit b at time k
	$\mathtt{I}_i^n$	n-byte PII memory start at the ith byte
	$\mathtt{O}_i^n$	n-byte PIO memory start at the ith byte
	$\mathtt{D}_i^n$	n-byte SCADA logging traffic start at the ith byte
	$\mathbf{I}_0^N$	samples of PII memory (N bytes)
	$\mathbf{O}_0^N$	samples of PIO memory (N bytes)
	$\mathbf{D}_0^L$	samples of SCADA logging traffic (L bytes)
	$\mathcal{E}_{\mathtt{M}}$	entropy distribution with MSB0
	$\mathcal{E}_{\mathtt{L}}$	entropy distribution with LSB0
Sets	$\mathtt{B}_{\mathtt{M}}^l$	least-significant bits of signals with MSB0
	$\mathtt{B}_{\mathtt{M}}^m$	most-significant bits of signals with MSB0
	$\mathtt{B}_{\mathtt{L}}^l$	least-significant bits of signals with LSB0
	$\mathtt{B}_{\mathtt{L}}^m$	most-significant bits of signals with LSB0
	$\mathcal{A}_{\mathtt{M}}^{\mathtt{I}}$	signal address in PII memory with MSB0
	$\mathcal{A}_{\mathtt{M}}^{\mathtt{O}}$	signal address in PIO memory with MSB0
	$\mathcal{A}_{\mathtt{M}}^{\mathtt{D}}$	signal address in SCADA logging traffic with MSB0
	$\mathcal{A}_{\mathtt{L}}^{\mathtt{I}}$	signal address in PII memory with LSB0
	$\mathcal{A}_{\mathtt{L}}^{\mathtt{O}}$	signal address in PIO memory with LSB0
	$\mathcal{A}_{\mathtt{L}}^{\mathtt{D}}$	signal address in SCADA logging traffic with LSB0

and injects the generated payloads into targeted PLC. In the following, we elaborate the four functional modules of PLC-SAGE using the platform EDS as an example.

6.1 Identify Physical Signals

Intuitively, the physical signals of an ICS can be collected from its SCADA database. However, SCADA database does not contain the address at which the physical signal is stored in the PLC's memory, which is needed to write the generated malicious payload to PLC memory to manipulate the ICS operation. Also, the physical signals stored in SCADA database are too outdated to generate real-time stealthy attack payloads, because the acquisition of sensor readings and control commands (to be explained in Sec. 6.3) and manipulation of PLC memory (to be explained in Sec. 6.4) must be completed before SCADA records and validates the PLC memory.

Noticing the fact that PLCs store all the received/issued physical signals in their PII/PIO memory, once connected to the victim PLC, PLC-SAGE collects physical signals by requesting its whole PII/PIO memory using the "Read" function of disclosed PLC communication protocols (e.g., Modbus [46], Siemens S7Comm [48], Rockwell PCCC [49]). The victim PLC will then return the memory state by encoding them into the "Data" field of the responding network traffic, as exampled in Fig. 13. However, even the "Data" field can be parsed according to the protocol format, identifying physical signals thereof is still challenging due to the following two unknowns.

- *The mappings between PLC PII/PIO memory and the physical signals is unknown.* PLCs store physical signals in PII/PIO memory according to specific mappings between memory addresses and sensors/actuators, but respond to the requesting of PII/PIO memory without indicating signal boundaries (see "Data" field in Fig. 13). As a result, identifying memory addresses of physical signals is a prerequisite for identifying physical signals from the responding network traffic.
- *The conversion between Real values in physical world and Hex values in PLCs is unknown.* PLCs store physical signals by converting the Real values in physical world (e.g., [0, 300]°C) to Hex values in PLCs (e.g., 0x0000-0xFFFF), and then arranging the bytes order according to the PLC endianness. Such a 2-fold conversion, such as converting signals of "10.2539°C" to "2240" and then to "0x08c0" rather than "0xc008", is the key to collecting physical signals from network traffic but is unavailable to attackers because: (i) the ranges of Real values in physical world are determined by the properties of individual sensor/actuator; (ii) even PLCs from the same vendor may use different endianness [60].

Format of responding traffic	S7 Header	S7 Paras	Data
responding for $\mathtt{I}_0^4$	\x32\x03\x00.....	\x04\x01.....	\x05\x00\x00\x00
responding for $\mathtt{I}_0^5$	\x32\x03\x00.....	\x04\x01.....	\x05\x00\x00\x00\x02

Fig. 13. The responding network traffic to the request of reading PLC's PII memory in EDS.

Note that the analog sensors and actuators in ICS vary so fast that SCADA cannot completely record their values, causing a significant model deviation $\omega(\tilde{k})$ and thus becoming the main attack targets of PLC-SAGE. We will next explain how PLC-SAGE addresses these two unknowns to identify the physical signals. Table 1 summarizes related symbols and definitions.

6.1.1 Memory Addressing for Physical Signals

With the default functions of PLC protocols, such as "*read_var*" of Siemens S7Comm, PLC-SAGE requests PLC memory states and parses the responding network traffic to abstract the involved physical signals, i.e., the "Data" field. Atop the "Data" field, PLC-SAGE will further identify the signal boundaries thereof and address the corresponding sensors/actuators with PLC memory.

Basic Idea. PLC-SAGE's identification of memory addresses for physical signals is inspired by the following two facts when PLCs store the physical signals in their memories.

- *Bit significance is monotonic to bit address.* The bit significance when encoding physical signals is monotonic to the bit address if the most-significant bit shares the same position with the most-significant byte, no matter what endianness the PLC accepts. For example, a signal "2240" can be stored as the Big-endian "0x08c0" or the Little-endian "0xc008"; and each byte of the signal (e.g., "0x08") can be numbered in the MSB0 (most-significant bit first) or LSB0 (least-significant bit first) order. These four cases of binary encoding for signal "2240" are summarized in Fig. 14, where the bit significance in Figs. 14(a) and 14(d) — where the most-significant bit shares the same

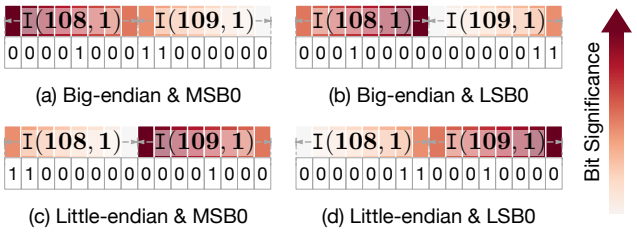


Fig. 14. Binary encoding of integer "2240" using MSB0 (or LSB0) with the byte order of the Big-endian (or Little-endian).

position with the most-significant byte — monotonically increases/decreases with the increase of bit address.

- *Bit flipping frequency is monotonic to bit significance.* The flipping frequency of bits within a physical signal is monotonic to their significance. Because the analog physical signals in PLC PII/PIO memory are constrained by the electric properties of sensors/actuators and thus vary continuously, rendering the bits with minor significance change more frequently than those with considerable significance.

These two facts indicate that the flipping frequency of bits is monotonic to the bit addresses within the same analog physical signals. To capture the bit flipping frequency, we define the bit state transition as $\dot{b}(k) \triangleq \text{abs}(b(k) - b(k-1))$, and its entropy as

$$H_{st}(b) = \begin{cases} H(\dot{b}), & \|\dot{b}\|_1 \leq |\dot{b}|/2, \\ 1, & \|\dot{b}\|_1 > |\dot{b}|/2, \end{cases} \tag{13}$$

where $b(k)$ denotes the value of bit b at time k, and $H(\cdot)$, $\|\cdot\|_1$, $|\cdot|$ denote the entropy, the l_1-norm, the element number of vectors, respectively. Using the bit state transition entropy $H_{st}(b)$, PLC-SAGE can distinctly identify the boundaries of signals from the obtained network traffic. As exampled in Figs. 15(a) and 15(b), all physical signals stored in PII/PIO memory have a nearly non-decreasing state transition entropy as the bit address increases. Note that with a given precision of PLC's input/output modules, a number of $\mu \in \{0, \ldots, 6\}$ insignificant bits of a signal may be constant. For example, the PLC in EDS generates 2-byte signals using a 13-bit input module, making the three least significant bits constant (see Fig. 15(a)).

Following the above observations, PLC-SAGE identifies the mapping between memory addresses and hardware sensors/actuators as follows.

Collect and Parse PLC Responding Traffic. According to the communication intervals between SCADA and the victim PLC, PLC-SAGE identifies the logging frequency of SCADA and requests the PLC PII/PIO memory at the same frequency (e.g., 2Hz). Denote $\mathtt{I}_i^n$ and $\mathtt{O}_i^n$ as consecutive memory with n bytes from the ith byte of PII and PIO memory. Considering the common engineering practices of ICS in prioritizing the low-address memory, PLC-SAGE, once connected to the victim PLC, will first keep reading $\mathtt{I}_0^n$ and $\mathtt{I}_0^{n+1}$ until the responding traffic to the request of $\mathtt{I}_n^1$ differs from $\mathtt{I}_{n-1}^1$. Such a difference flags the high-address memory in the responding traffic. For example, Fig. 13 depicts the responding traffic to the request of $\mathtt{I}_0^4$ and $\mathtt{I}_0^5$, where the emerged $\mathtt{I}_4^1 = \mathtt{0x02}$ in the responding traffic of $\mathtt{I}_0^5$ flags that the high-address memory is at the right of traffic. PLC-SAGE then will request $\mathtt{I}_0^N$ and $\mathtt{O}_0^N$ for T times, and collect the "Data" field in the corresponding responding traffic, denoted as $\mathbf{I}_0^N$ and $\mathbf{O}_0^N$, where N is the size of PII/PIO memory and can be found in the user manual (e.g., $N = 1024$ in Siemens S7-300 PLCs).

Identify Physical Signal Boundaries in Network Traffic. Using the collected $\mathbf{I}_0^N$ and $\mathbf{O}_0^N$, PLC-SAGE identifies the physical signals representing sensor readings and control commands, respectively. Because all PLCs, even from different vendors, store analog signals using 2-byte integers [61], [62], [63], PLC-SAGE selects a byte size $w = 2$ and identifies the signal boundaries in $\mathtt{I}_0^N$ and $\mathtt{O}_0^N$ as follows.

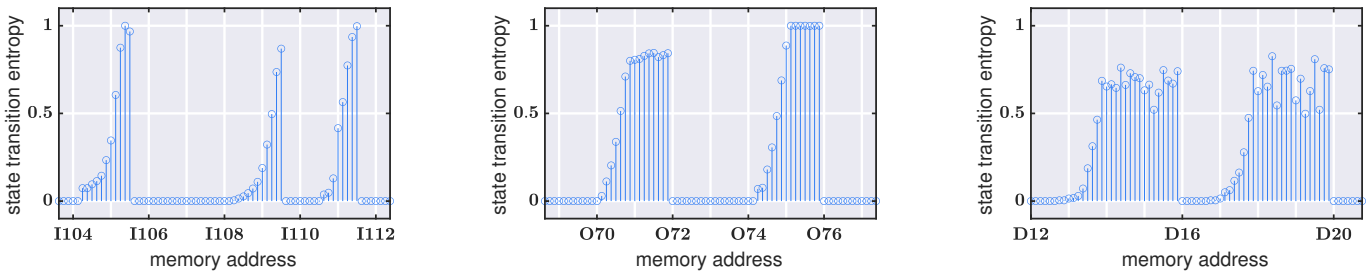
i) *Calculate State Transition Entropy.* For each byte $\mathtt{I}_i^1/\mathtt{O}_i^1$ in $\mathtt{I}_0^N/\mathtt{O}_0^N$, PLC-SAGE addresses its bits with $\{8i, \ldots, 8i+7\}$ in the direction of byte-address increasing. PLC-SAGE then numbers each hex value of $\mathtt{I}_0^N/\mathtt{O}_0^N$ to 8-bits according to the order of MSB0 and LSB0, respectively. By counting the state transition of each bit, PLC-SAGE obtains the corresponding state transition entropy using Eq. (13) and constructs the entropy distribution $\mathcal{E}_\mathtt{M}$ and $\mathcal{E}_\mathtt{L}$ under the order of MSB0 and LSB0, respectively.

ii) *Identify Signal Boundaries.* With the entropy distribution $\mathcal{E}_\mathtt{M}$ for $\mathtt{I}_0^N/\mathtt{O}_0^N$, PLC-SAGE identifies the least/most-significant bits of a physical signal by verifying $\mathcal{E}_\mathtt{M}[8i-1-\mu] - \mathcal{E}_\mathtt{M}[8i] > \alpha$ for all $i \in \{1, \ldots, N\}$ and $\mu \in \{0, \ldots, 6\}$, where α is the pre-defined threshold. Once a byte $\mathtt{I}_{i^*}^1/\mathtt{O}_{i^*}^1$ is verified, PLC-SAGE will add the bits $\{8i^*-1\}$ and $\{8(i^*-w)\}$ to the least/most-significant bit sets $\mathtt{B}_\mathtt{M}^l$ and $\mathtt{B}_\mathtt{M}^m$, respectively. With the entropy distribution $\mathcal{E}_\mathtt{L}$, PLC-SAGE verifies $\mathcal{E}_\mathtt{L}[8i+\mu] - \mathcal{E}_\mathtt{L}[8i-1] > \alpha$ and appends bits $\{8i^*\}$ and $\{8(i^*+w)-1\}$ to the least/most-significant bit sets $\mathtt{B}_\mathtt{L}^l$ and $\mathtt{B}_\mathtt{L}^m$, respectively. We will experimentally evaluate the impact of α on PLC-SAGE in Sec. 7.

iii) *Validate Signals' Plausibility.* For each signal segmented by bit $b^l \in \mathtt{B}_\mathtt{M}^l$ and corresponding bit $b^m \in \mathtt{B}_\mathtt{M}^m$, PLC-SAGE validates the ascending of $\mathcal{E}_\mathtt{M}[b^m : b^l]$ — if all $j \in \{0, \ldots, 14-\mu\}$ satisfy $\mathcal{E}_\mathtt{M}[b^m + j + 1] - \mathcal{E}_\mathtt{M}[b^m + j] > \beta$, where β is the pre-defined threshold. For each signal segmented by bit $b^l \in \mathtt{B}_\mathtt{L}^l$ and corresponding bit $b^m \in \mathtt{B}_\mathtt{L}^m$, PLC-SAGE validates the descending of $\mathcal{E}_\mathtt{L}[b^l : b^m]$ — if all $j \in \{\mu, \ldots, 14\}$ satisfy $\mathcal{E}_\mathtt{L}[b^l + j + 1] - \mathcal{E}_\mathtt{L}[b^l + j] < -\beta$. Signals that fail the validation will be removed from $\{\mathtt{B}_\mathtt{M}^l, \mathtt{B}_\mathtt{M}^m\}$ or $\{\mathtt{B}_\mathtt{L}^l, \mathtt{B}_\mathtt{L}^m\}$. Considering the signal noise, the state transition entropy may not be strictly monotonic to bit addresses in the same signal. PLC-SAGE thus set β as a negative value. The impact of β on PLC-SAGE will also be evaluated in Sec. 7.

Because the state transition entropy of a Big/Little-endian signal increases/decreases as the bit address increases, PLC-SAGE identifies the signal boundaries sets $\{\mathtt{B}_\mathtt{M}^l, \mathtt{B}_\mathtt{M}^m\}$ for the Big-endian and signal boundaries sets $\{\mathtt{B}_\mathtt{L}^l, \mathtt{B}_\mathtt{L}^m\}$ for the Little-endian.

Address Sensors/Actuators with PLC PII/PIO Memory. Given the signals segmented by bits in $\{\mathtt{B}_\mathtt{M}^l, \mathtt{B}_\mathtt{M}^m\}$ or $\{\mathtt{B}_\mathtt{L}^l, \mathtt{B}_\mathtt{L}^m\}$, PLC-SAGE addresses the corresponding sensors and actuators with PLC PII/PIO memory. Specifically, PLC-SAGE maps sensors with signals in $\mathbf{I}_0^N$ to address $\mathtt{I}_{b^m/8}^2$ or $\mathtt{I}_{b^l/8}^2$ and actuators with signals in $\mathbf{O}_0^N$ to address $\mathtt{O}_{b^m/8}^2$



(a) PII memory (b) PIO memory (c) SCADA logging traffic

Fig. 15. Calculating the state transition entropy for each bit of PLC memory in EDS.

or $\mathtt{O}^2_{b^l/8}$, where $b^m \in \mathtt{B}^m_{\mathtt{M}}$ and $b^l \in \mathtt{B}^l_{\mathtt{L}}$. These addresses make up with signal identity sets $\mathcal{A}^{\mathtt{I}}_{\mathtt{M}}$ and $\mathcal{A}^{\mathtt{O}}_{\mathtt{M}}$ (or $\mathcal{A}^{\mathtt{I}}_{\mathtt{L}}$ and $\mathcal{A}^{\mathtt{O}}_{\mathtt{L}}$).

6.1.2 Value Converting for Hex Stream

Signals processed by PLC control logic have the Real type of values in the physical world. In contrast, the signals identified from $\mathbf{I}^N_0$ and $\mathbf{O}^N_0$ are represented as hexadecimal integers. PLC-SAGE thus needs further to convert the identified signals to Real values with physical range.

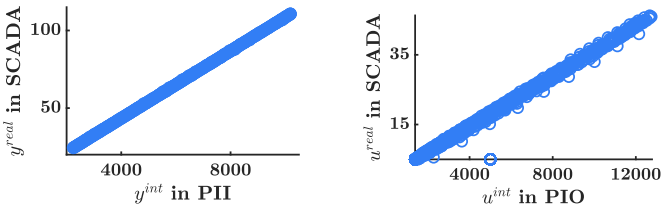
Basic Idea. For ease of monitoring in SCADA, PLCs convert physical signals into Real value, denoted as x^{real}, based on the physical ranges of sensors/actuators and the integer ranges of input/output modules:

$$x^{real} = [(x^{real}_{\max} - x^{real}_{\min})/(x^{int}_{\max} - x^{int}_{\min})] * (x^{int} - x^{int}_{\min}) + x^{real}_{\min},$$

where $x^{real}_{\min}$ and $x^{real}_{\max}$ denote the minimum and maximum values of sensors/actuators, $x^{int}_{\min}$ and $x^{int}_{\max}$ denote the minimum and maximum values of PLC input/output values. As depicted in Fig. 16, the linear correlation between signals in PII/PIO memory and signals in SCADA gives PLC-SAGE the chance to convert Hex stream to Real values.

Identify Physical Signals From SCADA Logging Traffic. Note that the linear relationship between the Real type and Integer type of the same physical signal only exists when their clocks are synchronized, which the SCADA database cannot guarantee. Thus, along with requesting the PLC PII/PIO memory, PLC-SAGE also simultaneously eavesdrops on the SCADA logging traffic, which naturally synchronizes with the responding traffic of PLC memory requesting. According to the protocol format, PLC-SAGE parses the "Data" field of SCADA logging traffic as $\mathbf{D}^L_0$, where L denotes the number of bytes in the "Data" field in SCADA logging traffic.

PLC-SAGE then identifies physical signals from the collected $\mathbf{D}^L_0$. PLCs usually store the Real values of physical signals using 4-byte float numbers [62], [63]. Note that the bit significance in numbering floats also monotonically relates to its address [64], making the flipping frequency of numbered bits also monotonic to the bit addresses. Hence, PLC-SAGE identifies the signal boundaries in $\mathbf{D}^L_0$ in the same way of segmenting $\mathbf{I}^N_0$ and $\mathbf{O}^N_0$ in Sec. 6.1.1, by setting $w = 4$ and $\mu = 0$. PLC-SAGE addresses the segmented signals using identity sets $\mathcal{A}^{\mathtt{D}}_{\mathtt{M}} = \{\mathtt{D}^4_{b^m/8}\}$ and $\mathcal{A}^{\mathtt{D}}_{\mathtt{L}} = \{\mathtt{D}^4_{b^l/8}\}$. Due to the signal noise and quantification errors of PLC input/output modules, the state transition entropy of the 23-bit faction in float number may not be strictly monotonic, as exampled by physical signals in Fig. 15(c). PLC-SAGE thus also selects a negative β to validate signals' plausibility.



(a) $\mathtt{I}^2_{108}$ vs. $\mathtt{D}^4_0$ (b) $\mathtt{O}^2_{70}$ vs. $\mathtt{D}^4_{48}$

Fig. 16. Correlation between signals in PII/PIO memory and corresponding signals in SCADA.

TABLE 2
Exampled correlations between sensor readings in PLC memory and those in SCADA logging traffic using different endianness.

	Big-endian			Little-endian		
SCADA Traffic	PLC PII/PIO memory			PLC PII/PIO memory		
	$\mathtt{I}^2_{88}$	$\mathtt{I}^2_{90}$	$\mathtt{I}^2_{92}$	$\mathtt{I}^2_{88}$	$\mathtt{I}^2_{90}$	$\mathtt{I}^2_{92}$
$\mathtt{D}^4_{20}$	**0.998**	-0.522	-0.038	0.047	-0.001	0.001
$\mathtt{D}^4_{24}$	-0.521	**0.999**	-0.358	0.266	0.243	-0.042
$\mathtt{D}^4_{32}$	-0.039	-0.357	**0.997**	-0.052	0.018	0.029

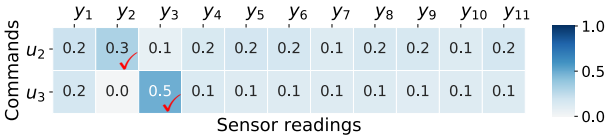
The pairs of $\{\mathtt{I}^2_{88}, \mathtt{D}^4_{20}\}$, $\{\mathtt{I}^2_{90}, \mathtt{D}^4_{24}\}$ and $\{\mathtt{I}^2_{92}, \mathtt{D}^4_{32}\}$ specify three sensor readings of EDS, respectively.

Generate Signals With Physical Range. Based on signal boundaries identified from the PLC responding traffic and the SCADA logging traffic, PLC-SAGE generates the physical signals in Real values by identifying PLC endianness and scaling their value ranges.

PLC-SAGE first converts the signals in $\{\mathcal{A}^{\mathtt{I}}_{\mathtt{M}}, \mathcal{A}^{\mathtt{O}}_{\mathtt{M}}, \mathcal{A}^{\mathtt{D}}_{\mathtt{M}}\}$ and $\{\mathcal{A}^{\mathtt{I}}_{\mathtt{L}}, \mathcal{A}^{\mathtt{O}}_{\mathtt{L}}, \mathcal{A}^{\mathtt{D}}_{\mathtt{L}}\}$ to decimal values according to the byte order of Big-endian and Little-endian, respectively. Note that the float numbers in $\mathcal{A}^{\mathtt{D}}_{\mathtt{M}}$ and $\mathcal{A}^{\mathtt{D}}_{\mathtt{L}}$ are converted following the IEEE-754 standard which is adopted by most PLCs vendors [26], [65]. As signal correlations in Table 2 show, only if integers in PII/PIO memory and float numbers in SCADA logging traffic are converted to decimals with a correct endianness can they have a linear correlation in representing the same physical signal. PLC-SAGE thus identifies the PLC endianness as the one that makes the converted decimal integers have the most significant correlations with the converted float numbers. PLC-SAGE, in the end, keeps the physical signals under the inferred PLC endianness and re-denotes the corresponding address sets as $\{\mathcal{A}^{\mathtt{I}}, \mathcal{A}^{\mathtt{O}}, \mathcal{A}^{\mathtt{D}}\}$.

Based on the fact that physical signals from different sensors/actuators usually follow nonlinear relationships, such as the nonlinear Bernoulli's equation in chemical engineering, resulting in weak correlation as exemplified in Table 2. Therefore, atop the linear correlation among signals

(a) weighting causalities using Mutual Information



(b) weighting causalities using metric defined in Eq. (14)

Fig. 17. Quantifying causalities of control commands in EDS using two weight metrics.

in PII/PIO memory and SCADA logging traffic, PLC-SAGE maps each signal $\mathtt{I}_i^2/\mathtt{O}_i^2$ in $\mathcal{A}^{\mathtt{I}}/\mathcal{A}^{\mathtt{O}}$ with signal $\mathtt{D}_i^4$ in $\mathcal{A}^{\mathtt{D}}$ that has the highest linear correlation coefficient (and ≥ 0.95). PLC-SAGE then identifies the scaling function for each identified signal $\mathtt{I}_i^2$ (or $\mathtt{O}_i^2$) based on its mapped $\mathtt{D}_i^4$.

6.2 Restore Control Logic

Based on the identified and converted sensor readings and control commands from PLC memory, PLC-SAGE further restores control logic among them.

Instead of characterizing all PLC control logic together as done in previous work [34], PLC-SAGE specifically restores the control logic for each control command, which allows for targeted stealthy attacks on individual commands. PLC-SAGE first identifies the causal relationships of each control command, i.e., the generation of control command is based on which sensor readings, using a control graph $\mathcal{G}(\mathtt{V}, \mathtt{E}, \mathtt{W})$:

- $\mathtt{V} = \{\mathtt{Y}, \mathtt{U}, \mathtt{S}\}$ consists of a sensor node set $\mathtt{Y}$, a command node set $\mathtt{U}$ and a setpoint node set $\mathtt{S}$, representing the sensor readings, control commands and expected system working states, respectively.
- $\mathtt{E} = \{e_y^u\}$ is the set of directed edges connecting nodes $y \in \mathtt{Y}$ to nodes $u \in \mathtt{U}$, representing the causalities between sensor readings and control commands. An edge e_y^u exists if the states of node y can affect the states of node u. The common practice of generating one control command based on one feedback sensor reading makes the in-degree of $u \in \mathtt{U}$ equals to 1.
- $\mathtt{W} = \{w_y^u\}$ is the set of edge weights representing the causality strength of all node pairs $\{y, u\}$s described in $\mathtt{E}$. Note that Mutual Information between $\boldsymbol{u}$ and $\boldsymbol{y}$ is usually used to quantify the correlation thereof. But considering the fact that the historical sensor readings $[y(0), \ldots, y(k-1)]$ also affect $u(k)$ such as in Eq. (1), and the effect has already involved in the command $u(k-1)$, PLC-SAGE quantifies the edge weight using Conditional Mutual Information,

$$w_y^u = I(\mathcal{Y}; \mathcal{U} \mid \mathcal{U}^a)/H(\mathcal{U} \mid \mathcal{U}^a), \tag{14}$$

where $\mathcal{Y}$, $\mathcal{U}$ and $\mathcal{U}^a$ denote the distribution of $\boldsymbol{y}[1:k]$, $\boldsymbol{u}[1:k]$ and $\boldsymbol{u}[0:k-1]$, respectively; $H(\mathcal{U} \mid \mathcal{U}^a)$ denotes the entropy of $\mathcal{U}$ conditional on $\mathcal{U}^a$; $I(\mathcal{Y}; \mathcal{U} \mid \mathcal{U}^a)$ denotes the Mutual Information of $I(\mathcal{Y}; \mathcal{U})$ conditional on $\mathcal{U}^a$. As depicted in Fig. 17, the weight metric of PLC-SAGE outperforms Mutual Information in weighting causalities of EDS by (i) correctly identifying the causal sensor reading y_2 for u_2, (ii) identifying the causality between u_3 and y_3 more distinctively.

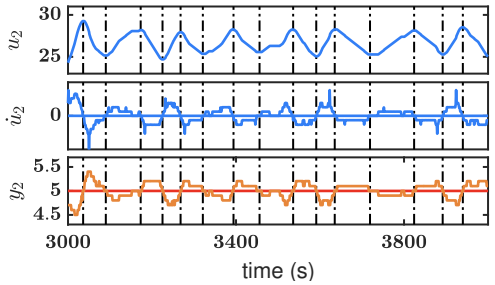


Fig. 18. Identifying the setpoint value based on the differentiation of control commands and corresponding sensor readings.

Note that different from the control graph proposed in [15], the setpoint node set $\mathtt{S}$ in SCADA database is not acquired by PLC-SAGE, and doesn't have any special behaviors in the PLC memory. PLC-SAGE addresses such issue by exploiting the *feedback mechanism* in control systems and constructs the control graph as follows.

i) **Identify Node Sets** $\{\mathtt{Y}, \mathtt{U}\}$**.** Based on the identified sensor readings in $\mathcal{A}^{\mathtt{I}}$ and the identified control commands in $\mathcal{A}^{\mathtt{O}}$, PLC-SAGE construct node sets $\{\mathtt{Y}, \mathtt{U}\}$.

ii) **Identify Edge Set** $\mathtt{E}$**.** According to the defined weight Eq. (14), PLC-SAGE quantifies the causalities of each command node $u \in \mathtt{U}$ with each sensor node $y \in \mathtt{Y}$. Then by identifying the maximum causality of each command node $u \in \mathtt{U}$, denoted as w_y^u, PLC-SAGE determines the edges $e_{y^*}^u \in \mathtt{E}$ for all command nodes.

iii) **Infer Node Set** $\mathtt{S}$**.** For each node pair $\{y, u\}$ defined by the edge $e_y^u \in \mathtt{E}$, PLC-SAGE infers the corresponding expected system working state. Because of the fundamental *feedback mechanism* in control systems, one key behavior of the control command is that it will keep increasing (or decreasing) until the corresponding sensor reading meets the defined working state. As corroborated in Fig. 18, the control command u_2 for cooling water valve of EDS stops changing — i.e., $\dot{u}_2 = 0$ — when all values of cooling water flow y_2 meet the setpoint s_2. This behavior allows PLC-SAGE to infer the setpoint nodes for all edges e_y^u. Specifically, PLC-SAGE builds an array $\boldsymbol{s}$ to collect the values of sensor reading y when the control command u has a zero derivative, and infers the corresponding setpoint s as the average of $\boldsymbol{s}$, i.e., $s = \bar{\boldsymbol{s}}$.

iv) **Verify Edge Set** $\mathtt{E}$**.** The constructed array $\boldsymbol{s}$ may have a large variance if the predefined work state is not constant or the corresponding node pair $\{y, u\}$ is mismatched. PLC-SAGE thus removes e_y^u from $\mathtt{E}$ if the corresponding setpoint s violates $\sigma(\boldsymbol{s}) \leq 0.01$, where $\sigma(\cdot)$ denotes the standard deviation function.

Atop the constructed control graph $\mathcal{G}(\mathtt{V}, \mathtt{E}, \mathtt{W})$, PLC-SAGE restores the control logic for each edge in $\mathtt{E}$. Based on the time series $\boldsymbol{y}$, $\boldsymbol{u}$ and the setpoint value s, PLC-SAGE estimates the parameters $\{A, B, C, D\}$ of Eqs. (4)-(5), using the least-square system identification [66]. Note that the parameters $\{A, B, C, D\}$ can also be updated online using

the latest obtained sensor readings and control commands.

6.3 Generate Malicious Payload

Based on the restored control logic in Sec. 6.2, PLC-SAGE further implements the MMSE command prediction (as described in Sec. 5.2) for each control command u to generate the corresponding stealthy malicious payload $\tilde{u}^a$.

i) **Acquire Sensor Readings and Control Commands.** According to memory addresses in sets $\mathcal{A}^{\mathrm{I}}$ and $\mathcal{A}^{\mathrm{O}}$ (identified in Sec. 6.1), PLC-SAGE reads the hexadecimal integers of sensor readings $\tilde{y}(\tilde{k})$s and control commands $\tilde{u}(\tilde{k})$s at the frequency of SCADA logging. Based on the range scaling functions of all physical signals (identified in Sec. 6.1), PLC-SAGE further converts the received integers to Real values.

ii) **Predict Control Commands.** For each control command $\tilde{u}$, PLC-SAGE generates the MMSE command prediction $\hat{\tilde{u}}^{-}(\tilde{k}+1)$ based on the acquired sensor reading $\tilde{y}(\tilde{k})$ using Eqs. (8)-(10), where the parameters $\{A, B, C, D\}$ and setpoint s are identified in Sec. 6.2. The corresponding prediction error $z(\tilde{k})$ can be obtained based on the simultaneously acquired $\tilde{u}(\tilde{k})$ using Eq. (11).

iii) **Generate Malicious Payload.** PLC-SAGE keeps predicting control commands by iteratively executing the above two steps. When the corresponding prediction error distribution $\mathcal{Z}$ becomes stable, which has been proved to be achieved after a few iterations [67], PLC-SAGE generates the malicious payload $\tilde{u}^a(\tilde{k})$ according to Eq. (12). Note that during the attack period, PLC-SAGE shall use the $\tilde{u}^a(\tilde{k})$ instead of $\tilde{u}(\tilde{k})$ to complete the *a posteriori* and *a priori* state estimation in the next attacking period. This is because the $\tilde{u}^a(\tilde{k})$ is the control command that SCADA actually records.

6.4 Inject Malicious Payload

By exploiting the authentication vulnerabilities of PLC protocols, PLC-SAGE launches the stealthy malicious payload $\tilde{u}^a(\tilde{k})$ to the victim PLC at the frequency of SCADA logging. Once $\tilde{u}^a(\tilde{k})$ is generated, PLC-SAGE will further (i) scale it according to the identified signal conversion to fit the range of PLC output modules; (ii) convert it to a hexadecimal integer according to the identified PLC endianness; (iii) send it to PLC memory according to the mapped address in $\mathcal{A}^{\mathrm{O}}$. With the default debugging function of PLC protocols, such as the "*Force*" function, PLC-SAGE maintains the specified PLC memory can only be written by the $\tilde{u}^a(\tilde{k})$ rather than the legitimate control program.

7 Evaluation

We have implemented and evaluated PLC-SAGE on two ICS platforms: the Ethanol Distillation System (EDS) and the Tennessee-Eastman chemical System (TES) which is a representative benchmark of chemical engineering systems [68].

7.1 Evaluation on EDS

We first evaluate PLC-SAGE on EDS, whose PLC executes the control logic at 500Hz and the SCADA monitors the system states at 2Hz. We implemented and installed PLC-SAGE

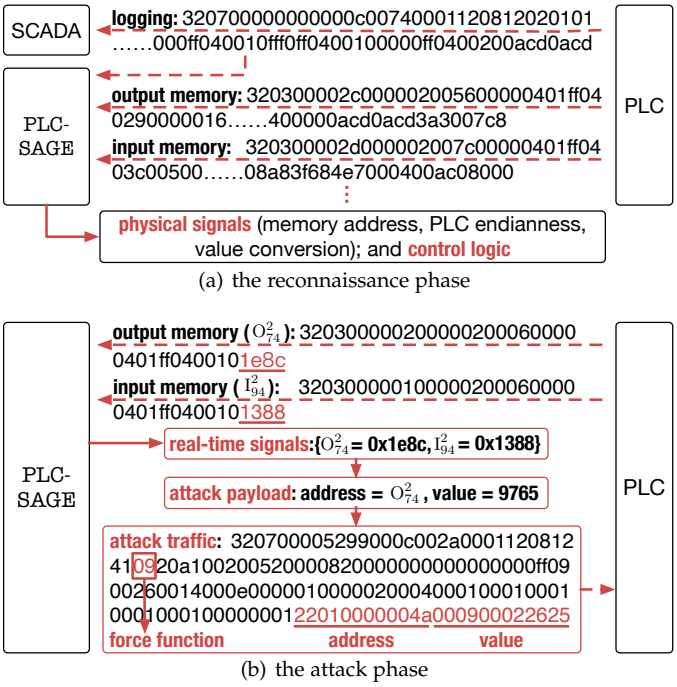


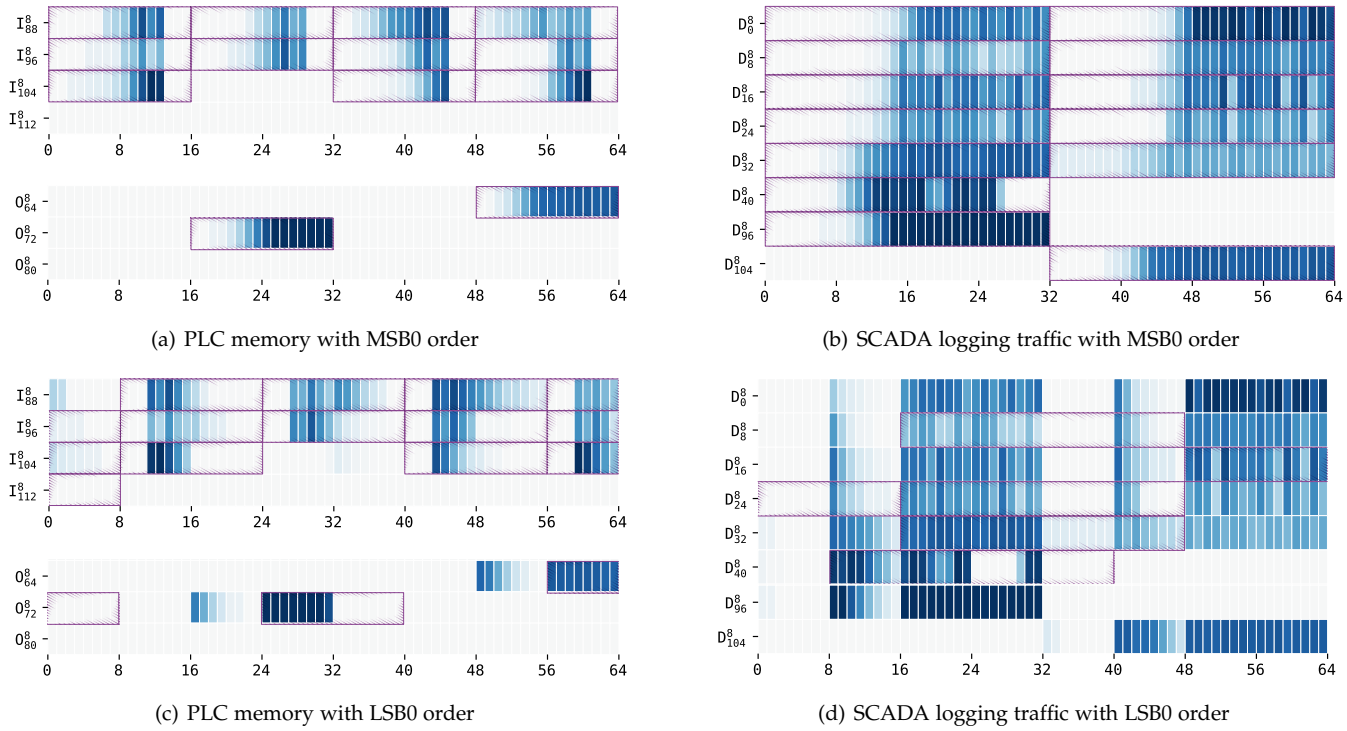
Fig. 19. An example when evaluating PLC-SAGE on EDS.

as a software component on a computer that communicates with the victim PLC and the corresponding SCADA. Fig. 19 gives the experimental examples for evaluating PLC-SAGE's reconnaissance and attack phases. Specifically, (i) based on the responding traffic to the input/output memory acquisition and the SCADA monitoring traffic, we evaluate PLC-SAGE's reconnaissance on physical signals and control logic among them; (ii) building on the above identification results, we further evaluate PLC-SAGE's capability in mounting stealthy attacks.

7.1.1 Evaluation on PLC-SAGE's Reconnaissance Phase

Once breaking into EDS's ICS network, PLC-SAGE eavesdrops on the SCADA logging traffic and identifies the SCADA logging frequency as 2Hz based on the time intervals between packets. PLC-SAGE also identifies EDS is equipped with a Siemens PLC based on the communication protocol and sets the memory size as 1024 bytes (i.e., the memory size of EDS's PLC).

Identification of Physical Signals. Taking 3,000 samples (i.e., a 25-minute log) of PLC memory as inputs, PLC-SAGE correctly identifies all the 13 physical signals of EDS, including 11 sensor readings and 2 control commands. The memory addressing for physical signals with the encoding order of MSB0 and LSB0 is detailed, respectively, in Fig. 20, where the purple frames indicate the identified PLC memory addresses. As listed in Table 3, by calculating the correlation coefficients between signals in PLC memory and signals in SCADA logging traffic — an average of 0.997 (Fig. 20(a) vs. Fig. 20(b)) and 0.209 (Fig. 20(c) vs. Fig. 20(d)) under the binaries encoding order of MSB0 and LSB0, respectively — PLC-SAGE correctly concludes that the PLC in EDS platform uses the Big-endian. The signals identified in PLC memory is also correctly converted to Real values.



(a) PLC memory with MSB0 order

(b) SCADA logging traffic with MSB0 order

(c) PLC memory with LSB0 order

(d) SCADA logging traffic with LSB0 order

Fig. 20. Identifying physical signals in EDS's PLC memory and SCADA logging traffic, with the encoding order of MSB0 and LSB0, respectively.

TABLE 3
Correlating physical signals in PLC memory and those in SCADA logging traffic using Big-endian for MSB0 ordered signals and Little-endian for LSB0 ordered signals, respectively.

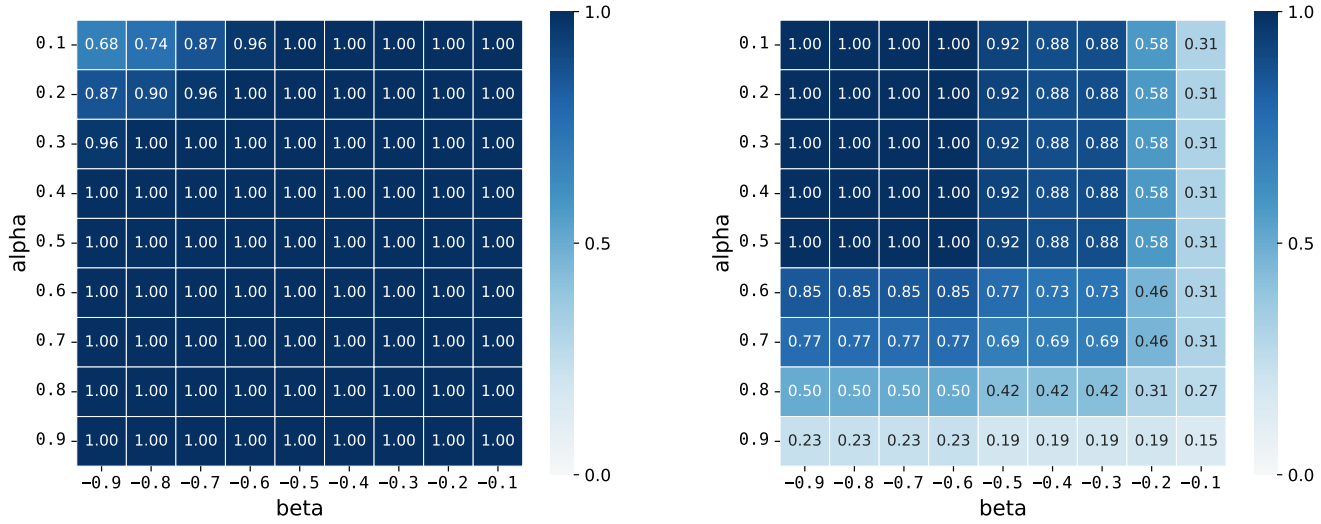
Big-endian															Little-endian													
SCADA Traffic	PLC's PII/PIO memory														SCADA Traffic	PLC's PII/PIO memory													
	I^2_{88}	I^2_{90}	I^2_{92}	I^2_{94}	I^2_{96}	I^2_{98}	I^2_{100}	I^2_{102}	I^2_{104}	I^2_{108}	I^2_{110}	O^2_{70}	O^2_{74}			I^2_{89}	I^2_{91}	I^2_{93}	I^2_{95}	I^2_{97}	I^2_{99}	I^2_{101}	I^2_{103}	I^2_{105}	I^2_{109}	I^2_{111}	O^2_{71}	O^2_{75}
D^4_{20}	**1.0**	–	–	–	–	–	–	–	–	–	–	–	–		D^4_1	–	.04	–	.00	–	–	–	–	–	–	–	.02	.02
D^4_{24}	–	**1.0**	–	.07	–	–	–	–	–	–	.42	.38	.36		D^4_6	.06	–	.01	–	–	–	–	–	–	–	.11	.07	–
D^4_{32}	–	–	**1.0**	–	.43	.23	.31	.31	.42	.42	.02	–	–		D^4_{10}	–	.01	–	.03	.09	.09	.02	.04	–	–	.06	–	–
D^4_{36}	–	.07	–	**.98**	–	.04	.02	.01	–	–	–	.03	.04		D^4_{22}	–	.07	–	.04	.01	.01	–	.02	.01	–	–	.07	–
D^4_8	–	–	.43	–	**1.0**	.68	.81	.82	.93	.90	.02	–	–		D^4_{26}	.03	–	–	–	–	–	–	–	.05	–	.13	.04	–
D^4_{16}	–	–	.23	.04	.68	**1.0**	.94	.85	.52	.47	.08	.03	–		D^4_{33}	–	–	.50	–	.05	.05	–	.09	–	–	.11	–	.02
D^4_{12}	–	–	.31	.02	.81	.93	**1.0**	.93	.67	.59	.15	.02	–		D^4_{37}	.14	–	.18	–	–	–	.00	–	–	.15	–	.00	–
D^4_{28}	–	–	.30	.01	.82	.85	.93	**.99**	.66	.56	.30	–	–		D^4_{41}	.07	–	.04	–	.07	.07	.05	–	.02	.14	–	–	.01
D^4_4	–	–	.41	–	.93	.52	.67	.66	**1.0**	.97	–	.00	–		D^4_{74}	.78	–	.04	–	–	–	–	–	–	.59	.30	–	–
D^4_0	–	–	.42	–	.90	.47	.59	.56	.97	**1.0**	–	–	–		D^4_{110}	.00	–	–	–	–	–	.01	–	–	.01	–	.03	.06
D^4_{40}	–	.42	.02	–	.02	.08	.16	.30	–	–	**.99**	–	–		Null	–	–	–	–	–	–	–	–	–	–	–	–	–
D^4_{72}	–	.38	–	.03	–	.04	.02	–	.01	–	–	**1.0**	.03		Null	–	–	–	–	–	–	–	–	–	–	–	–	–
D^4_{108}	–	.36	–	–	–	–	–	–	–	–	–	.02	**1.0**		Null	–	–	–	–	–	–	–	–	–	–	–	–	–

The notation of "–" denotes negative correlation.

We further evaluate the robustness of PLC-SAGE against the signal segmentation thresholds α and β. As shown in Figs. 21(a) and 21(b), PLC-SAGE identifies signals with 100% precision (the ratio of correctly identified signals in all identified signals) and 100% recall (the ratio of correctly identified signals in all actual signals) when $\alpha \in [0.3, 0.5]$ and $\beta \in [-0.8, -0.6]$. A larger α means a more strict checking on signals' boundaries, excluding more falsely identified signals and improving the precision, as corroborated by Fig. 21. However, too large an α excludes true signals that do not vary frequently, degrading the recall. Fig. 21 also shows that a larger β allows PLC-SAGE to identify signals with higher precision but at the cost of a decreased recall. This is because a larger β drives PLC-SAGE to evaluate the monotonicity of state transition entropy more strictly, and the signal with large measurement noise will be discarded.

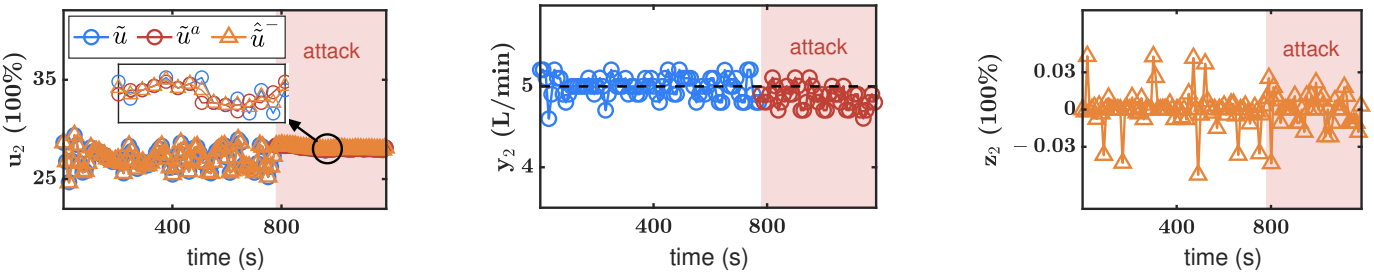
Restoration of Control Logic. For each of the identified 2 control commands, PLC-SAGE identifies their related sensor readings and then infers the corresponding setpoint. With the above-identified $3,000$ samples of physical signals as inputs, PLC-SAGE correctly constructs the control graph $\mathcal{G}(\mathtt{V}_{\mathtt{EDS}}, \mathtt{E}_{\mathtt{EDS}}, \mathtt{W}_{\mathtt{EDS}})$ of EDS as:

$$\mathtt{V}_{\mathtt{EDS}} = \{\{\mathtt{I}^2_{94}, \mathtt{I}^2_{110}\}, \{\mathtt{O}^2_{70}, \mathtt{O}^2_{74}\}, \{s_{\mathtt{I}^2_{94}}, s_{\mathtt{I}^2_{110}}\}\},$$



(a) precision (b) recall

Fig. 21. Identifying signals in PLC memory and the SCADA logging traffic in EDS using different α and β.



(a) generated stealthy payload (b) attack impact (c) command prediction error in SCADA

Fig. 22. Generating and launching stealthy command payloads against the ethanol gas condensing loop in EDS.

$$\mathtt{E}_{\mathtt{EDS}} = \{e^{\mathtt{O}^2_{70}}_{\mathtt{I}^2_{110}}, e^{\mathtt{O}^2_{74}}_{\mathtt{I}^2_{94}}\}, \qquad \mathtt{W}_{\mathtt{EDS}} = \{w^{\mathtt{O}^2_{70}}_{\mathtt{I}^2_{110}}, w^{\mathtt{O}^2_{74}}_{\mathtt{I}^2_{94}}\},$$

where the system setpoint nodes $s_{\mathtt{I}^2_{94}}$ and $s_{\mathtt{I}^2_{110}}$ represent the constant value 5.0 and 0.1, respectively; the control edge $e^{\mathtt{O}^2_{70}}_{\mathtt{I}^2_{110}}$ and $e^{\mathtt{O}^2_{74}}_{\mathtt{I}^2_{94}}$ represent the causal relationships in Loop-II and Loop-III. Note that for the clarity of control graph $\mathcal{G}$, other isolated sensor nodes are omitted in the node set $\mathtt{V}_{\mathtt{EDS}}$. Based on the identified graph $\mathcal{G}$, PLC-SAGE successfully restores the parameters $\{A, B, C, D\}$ for each edge in $\mathtt{E}_{\mathtt{EDS}}$.

7.1.2 Evaluation on PLC-SAGE's Attack Phase

We next validate PLC-SAGE's ability to damage the physical process of EDS without being detected by the SCADA. As shown in Fig. 19(b), PLC-SAGE acquires the real-time states of PLC operation according to the memory address and value conversion of each identified physical signal. Based on $\mathcal{G}(\mathtt{V}_{\mathtt{EDS}}, \mathtt{E}_{\mathtt{EDS}}, \mathtt{W}_{\mathtt{EDS}})$ and the parameters of the restored control logic for each edge in $\mathtt{E}_{\mathtt{EDS}}$, PLC-SAGE predicts the control commands at runtime, and generates and injects the stealthy malicious payloads. Against the ethanol gas condensing loop (i.e., Loop-II), PLC-SAGE forces the generated control command payload $\tilde{u}_2^a$ (see Fig. 22(a)) to memory address $\mathtt{O}^2_{74}$. Note that the malicious control command $\tilde{u}_2^a$ departs from the expected legitimate command $\tilde{u}_2$ at each SCADA logging time (actually at all the PLC execution cycles more specifically), as shown by the zoom-in trace in Fig. 22(a). Although the manipulation of control command is constrained by the prediction error distribution of legitimate control commands (i.e., $\tilde{u}_2 - \hat{\tilde{u}}_2^-$), PLC-SAGE successfully decreases the flow of cooling water in EDS (see Fig. 22(b)). At the same time, the SCADA doesn't detect this attack — i.e., PLC-SAGE is stealthy — because (i) the control commands prediction error (z_2) under attack shows no difference compared to the prediction error before the attack (see Fig. 22(c)), and (ii) the decreased cooling water flow (y_2) with reduced valve opening (u_2) agrees with the laws of physics.

7.2 Evaluation on TES

To evaluate PLC-SAGE's effectiveness and stealthiness when being deployed on a large-scale control system, we further implement PLC-SAGE on TES, a realistic chemical engineering simulation that is commonly used for evaluating attack strategies [69]. In total, 17 control loops involving 41 sensors and 12 actuators are used to stabilize the physical process of TES [70]. Along with the simulation of control loops, we further build a SCADA system to record TES's states and detect attacks. Note that to emulate a critical monitoring constraint in chemical engineering, we set the logging frequency of SCADA to be 10 times slower than

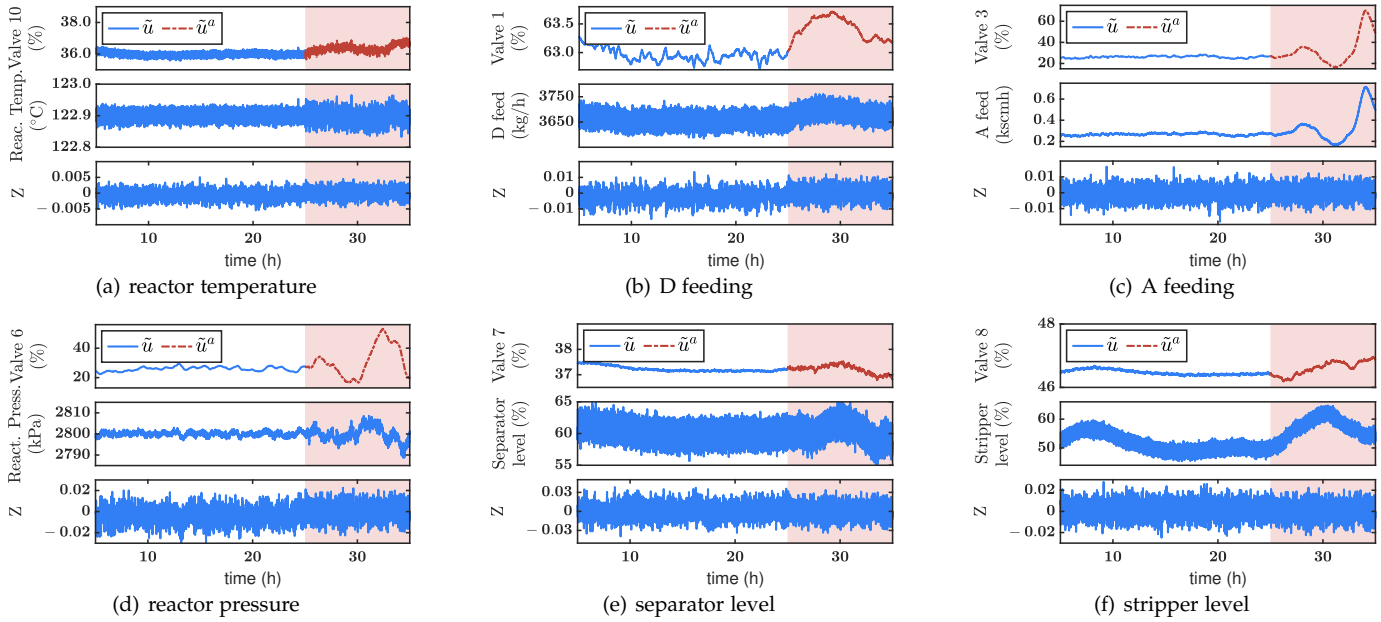


Fig. 23. Generating and launching stealthy command payloads against multiple control loops in TES.

PLC execution. Please see Appendix A in the Supplemental Material for more details of TES's implementation.

By simulating a 2.5-hour execution of TES, PLC-SAGE samples the system states for 500 times and identifies the control logic for each control command. Using the identified control logic, PLC-SAGE generates and launches the stealthy malicious payloads against all control commands. Due to the limited space, we show only a part of the attacks in Fig. 23, where the malicious payloads $\tilde{u}^a$s damage TES by enlarging the system control error, driving system states away from their setpoints, and degrading the system stability (see the middle sub-figures of Figs. 23(a)-23(f)). At the same time, the launched payloads $\tilde{u}^a$s do not change the distribution of control command prediction error in SCADA, as corroborated by the bottom sub-figures of Figs. 23(a)-23(f). This way, the stealthiness of PLC-SAGE in damaging physical processes of the large-scale control system is validated.

8 Further Analysis

Below, we provide several further analyses of PLC-SAGE in terms of scalability, cost, and potential physical impacts on the targeted ICS.

Scalability vs. Different PLCs. The scalability of PLC-SAGE in attacking PLCs from different vendors is primarily determined by the feasibility of manipulating PLC memory at runtime when the PLC executes the control program. For the purposes of testing, commissioning, and debugging, many PLCs are permitted to force their memory to defined values and without being overwritten by the control program [71], [72], [73]. Besides the corroborated feasibility of manipulating the memory of Siemens PLCs in Sec. 7, we have also validated such feasibility with a MicroLogix 1400 PLC from Rockwell Automation. Specifically, we have implemented the control logic of EDS in a MicroLogix 1400 PLC as shown in Fig. 24, and built a SCADA system to monitor its memory variation. Then, by exploiting the weak authentication of the

Fig. 24. Besides the Siemens S7-315 PLC used by EDS, we have also implemented PLC-SAGE on a MicroLogix 1400 PLC from Rockwell.

Rockwell PCCC protocol, PLC-SAGE constructed and sent the malicious communication traffic to the MicroLogix 1400 PLC to force its memory to manipulated values. Please see the demo video [53] for detailed memory manipulation.

Scalability vs. Different ICSs. PLC-SAGE can be applied to any ICS as long as there exists mismatched frequencies between the PLC execution and SCADA logging. This phenomenon is commonly observed in ICSs because: (i) PLCs complete one execution cycle typically within 100ms and set a maximum execution cycle, such as 150ms by default in Siemens PLC [27], to ensure the capability of real-time control; and (ii) SCADA will set a minimum data acquisition period, such as 500ms in Siemens WinCC software [26], to reduce the communication time of PLC (and thus guarantee the real-time control) and also save the storage in recording PLC memory.

Cost of PLC-SAGE. Except for the adversary abilities that are commonly observed in real life and literature (as listed in Sec. 3.3), PLC-SAGE does not require any additional cost. Also note that PLC-SAGE is deployed as a software component and does not require any hardware retrofits.

Physical Impacts. Besides enlarging the control errors and destabilizing the system as illustrated in Fig. 22 and Fig. 23, PLC-SAGE may also drive the physical process to critical dangerous states. For example, PLC-SAGE can shut down the TE process by increasing the reactor pressure to the upper bound (i.e., 3000kPa), as shown in Fig. 25.

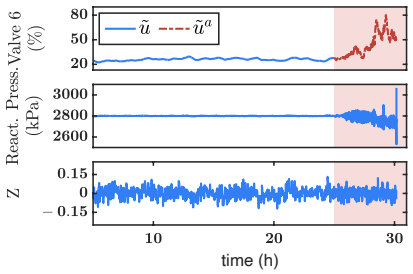


Fig. 25. PLC-SAGE can shut down the TE process by increasing the reactor pressure.

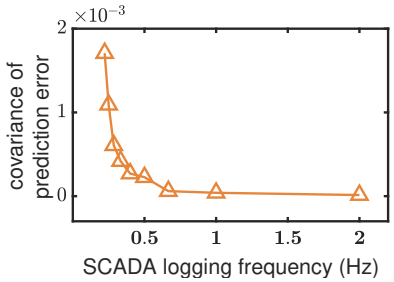


Fig. 26. The covariance of prediction errors for control command decreases when the SCADA monitors system operation with higher frequencies.

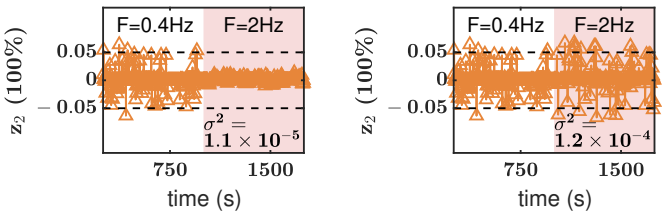
9 Mitigations

PLC-SAGE also sheds light on how to improve the resistance of PLCs to stealthy attacks.

9.1 Moving Target-based Attestation by Varying Monitoring Frequency

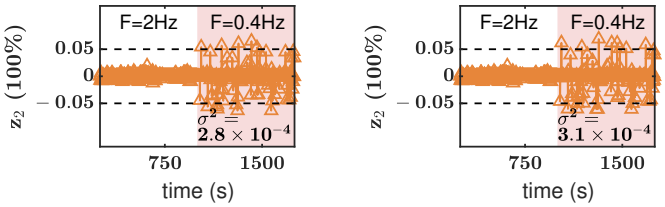
According to Theorem 5.1, different SCADA monitoring frequencies cause different logging errors, leading to different precision losses when validating PLC operation. As shown in Fig. 26, the covariance of prediction error for control command u_2 in EDS decreases with a higher SCADA monitoring frequency.

Inspired by this, we recommend a moving target-based attestation strategy that detects PLC-SAGE by randomly increases the SCADA monitoring frequencies. Since existing SCADA software only allows specifying the monitoring frequency at system startup but not changing it at runtime [26], [65], we developed a customized SCADA using the disclosed PLC communication protocols. Specifically, by acquiring the PLC states at multiple frequencies (e.g., 0.4Hz and 2Hz) offline, the attestation strategy pre-trains the restored control logic and identifies the corresponding prediction errors for each of these frequencies. When ICSs start running, the SCADA will maintain the monitoring frequency at the lowest level and randomly adjust it to other frequencies in real-time for a short time period. The validation model for PLC operation will be updated accordingly, which is invisible to adversaries. As depicted in Fig. 27(a), the covariance of prediction error z_2 in EDS should be decreased from 1.2×10^{-4} to 1.1×10^{-5} when increasing the monitoring frequency from 0.4Hz to 2Hz. But if a stealthy attack is under-going (see Fig. 27(b)), the prediction error z_2 will stay at a covariance of 1.2×10^{-4} even after the monitoring frequency is increased, contradicting the attestation expectation and making the stealthy attack detectable.



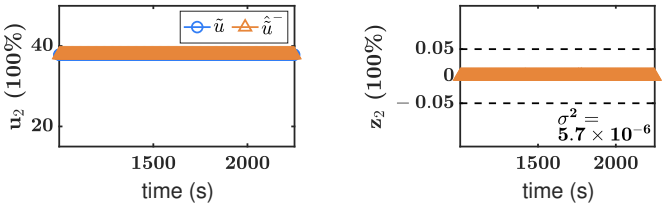
(a) Without Attack (b) Under Attack

Fig. 27. The stealthy attack against the detection model of 0.4Hz is detectable after changing SCADA logging frequency to 2Hz.



(a) Without Attack (b) Under Attack

Fig. 28. The stealthy attack against the detection model of 2Hz is not detectable after changing the SCADA logging frequency to 0.4Hz.



(a) command prediction (b) command prediction error

Fig. 29. Predicting the ethanol gas condensing command (i.e., u_2) in EDS after switching to a high-quality sensor.

Note that stealthy attacks can be detected only by increasing rather than decreasing the monitoring frequency, because the stealthy malicious payloads generated from the control logic validated at a higher frequency satisfy the validation of control logic at a lower frequency, as depicted in Fig. 28.

Note that different from the classic active attack detection strategies, such as watermarking control commands or adapting the dynamics of the physical process, the proposed attestation strategy will not affect the operation of the physical process.

9.2 Reducing Control Error by Increasing Sensing Accuracy

Theorem 5.1 also indicates that the minor control error $e(k)$ during the logging interval of $[\tilde{k}, \tilde{k}+1)$ can lead to a minor model deviation $\omega(\tilde{k})$, which further induces smaller prediction errors. Note that the control error $e(k) = s(k) - y(k)$ highly relies on the precision of measurement $y(k)$ — e.g., the process measurement without noise can induce a constant control error. Again, let us take the ethanol gas condensing in EDS as an example. By upgrading the precision of corresponding sensor for the measurement y_2, the SCADA with 2Hz monitoring frequency validates the logged control command $\tilde{u}_2$ with near-zero prediction errors, as plotted in Fig. 29. Thus, using highly accurate sensors for critical procedures in the physical process helps defend against stealthy attacks.

10 Conclusion

This paper unveils a newly identified vulnerability of ICSs resulting from a common practice, i.e., monitors the system operation with a much lower frequency than system execution. Additionally, we demonstrate the security implications of this vulnerability by introducing PLC-SAGE, an autonomous stealthy malicious payload generation attack against PLCs that is capable of manipulating ICS operations without being detected. Guided by the insights gained with PLC-SAGE, we propose two practices to bolster the resistance of PLCs against stealthy attacks. We have verified the stealthiness of PLC-SAGEon two ICS platforms and experimentally validated the effectiveness of the recommended mitigative strategies.

References

[1] 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.

[2] ICS Advisory. Schneider Electric IGSS SCADA Software. https://us-cert.cisa.gov/ics/advisories/icsa-20-084-02, 2020. [Online; Accessed December 2023].

[3] ICS Advisory. SIMATIC WinCC Graphics Designer. https://us-cert.cisa.gov/ics/advisories/icsa-21-040-09, 2021. [Online; Accessed December 2023].

[4] Cybersecurity and Infrastructure Security Agency. Rockwell Automation MicroLogix Controllers and RSLogix 500 Software. https://us-cert.cisa.gov/ics/advisories/icsa-20-070-06, 2020. [Online; Accessed December 2023].

[5] Siemens Security Advisory by Siemens ProductCERT. SSA-381684: Improper Password Protection during Authentication in SIMATIC S7-300 and S7-400 CPUs and Derived Products. https://cert-portal.siemens.com/productcert/pdf/ssa-381684.pdf, 2020. [Online; Accessed December 2023].

[6] Schneider Electric Security Notification. Security Notification - Modicon M100/M200/M221 Programmable Logic Controller (V3.0). https://www.se.com/ww/en/download/document/SEVD-2020-315-05/, 2021. [Online; Accessed December 2023].

[7] Charles Cooper. 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. [Online; Accessed December 2023].

[8] Kevin Collier. In Florida, a near-miss with a cybersecurity worst-case scenario. https://www.nbcnews.com/tech/security/florida-near-miss-cybersecurity-worst-case-scenario-n1257091, 2021. [Online; Accessed December 2023].

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

[10] 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 (CCS)*, pages 1092–1105, 2016.

[11] 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 (CCS)*, pages 817–831, 2018.

[12] Sriharsha Etigowni, Shamina Hossain-McKenzie, Maryam Kazerooni, Katherine Davis, and Saman Zonouz. Crystal (ball): I Look at Physics and Predict Control Flow! Just-Ahead-Of-Time Controller Recovery. In *Proceedings of the 34th Annual Computer Security Applications Conference*, pages 553–565, 2018.

[13] 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.

[14] 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.

[15] Zeyu Yang, Liang He, Peng Cheng, Jiming Chen, David KY Yau, and Linkang Du. PLC-Sleuth: Detecting and Localizing PLC Intrusions Using Control Invariants. In *23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020)*, pages 333–348, 2020.

[16] 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.

[17] Hamid Reza Ghaeini, Daniele Antonioli, Ferdinand Brasser, Ahmad-Reza Sadeghi, and Nils Ole Tippenhauer. State-Aware Anomaly Detection for Industrial Control Systems. In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, pages 1620–1628, 2018.

[18] Alessandro Erba, Riccardo Taormina, Stefano Galelli, Marcello Pogliani, Michele Carminati, Stefano Zanero, and Nils Ole Tippenhauer. Constrained Concealment Attacks against Reconstruction-based Anomaly Detectors in Industrial Control Systems. In *Annual Computer Security Applications Conference*, pages 480–495, 2020.

[19] 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.

[20] 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 (CCS)*, page 21–32, 2009.

[21] Yilin Mo and Bruno Sinopoli. False Data Injection Attacks in Control Systems. In *Preprints of the 1st workshop on Secure Control Systems*, pages 1–6, 2010.

[22] Ziyang Guo, Dawei Shi, Karl Henrik Johansson, and Ling Shi. Worst-case stealthy innovation-based linear attack on remote state estimation. *Automatica*, 89:117–124, 2018.

[23] Shaodong Wang, Chao Li, Qinmin Yang, Wenchao Meng, and Yunpeng Li. Detecting Stealthy Attacks in Power Systems based on Switching and Zonotopes. In *2023 International Annual Conference on Complex Systems and Intelligent Science (CSIS-IAC)*, pages 865–872, 2023.

[24] Stuart A Boyer. *SCADA: Supervisory Control and Data Acquisition (4th ed.)*. International Society of Automation, 2009.

[25] Keith Stouffer, Victoria Pillitteri, Suzanne Lightman, Marshall Abrams, and Adam Hahn. Guide to Industrial Control Systems (ICS) Security. *NIST special publication*, 800(82):1–247, 2015.

[26] Siemens AG. SIMATIC WinCC WinCC Professional V14. https://support.industry.siemens.com/cs/document/109742302/simatic-wincc-wincc-professional-v14?dti=0&lc=en-WW, 2016. [Online; Accessed December 2023].

[27] Siemens AG. SIMATIC S7-1500, S7-1500R/H, ET 200SP, ET 200pro Cycle and response times. https://cache.industry.siemens.com/dl/files/558/59193558/att_895996/v2/s71500_cycle_and_reaction_times_function_manual_en-US_en-US.pdf, 2023. [Online; Accessed December 2023].

[28] Jakapan Suaboot, Adil Fahad, Zahir Tari, John Grundy, Abdun Naser Mahmood, Abdulmohsen Almalawi, Albert Y Zomaya, and Khalil Drira. A Taxonomy of Supervised Learning for IDSs in SCADA Environments. *ACM Computing Surveys*, 53(2):1–37, 2020.

[29] Claude Fachkha, Elias Bou-Harb, Anastasis Keliris, Nasir D Memon, and Mustaque Ahamad. Internet-scale Probing of CPS: Inference, Characterization and Orchestration Analysis. In *Network and Distributed Systems Security (NDSS) Symposium*, 2017.

[30] Ruimin Sun, Alejandro Mera, Long Lu, and David Choffnes. SoK: Attacks on Industrial Control Logic and Formal Verification-Based Defenses. In *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 385–402, 2021.

[31] Gauthama Raman MR, Chuadhry Mujeeb Ahmed, and Aditya Mathur. Machine learning for intrusion detection in industrial control systems: challenges and lessons from experimental evaluation. *Cybersecurity*, 4:1–12, 2021.

[32] 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.

[33] 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.

[34] Yuqi Chen, Christopher M Poskitt, and Jun Sun. Code Integrity Attestation for PLCs using Black Box Neural Network Predictions. In *The ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)*, 2021.

[35] Moshe Kravchik and Asaf Shabtai. Efficient Cyber Attack Detection in Industrial Control Systems Using Lightweight Neural Networks and PCA. *IEEE Transactions on Dependable and Secure Computing*, 19(4):2179–2197, 2021.

[36] Bianca Scholten. *The Road to Integration: A Guide to Applying the ISA-95 Standard in Manufacturing*. International Society of Automation, 2007.

[37] Symantec DeepSight Adversary Intelligence Team. Seedworm: Group Compromises Government Agencies, Oil & Gas, NGOs, Telecoms, and IT Firms. https://symantec-enterprise-blogs.security.com/blogs/threat-intelligence/seedworm-espionage-group, 2018. [Online; Accessed December 2023].

[38] Symantec Threat Hunter Team. Shamoon: Destructive Threat Re-Emerges with New Sting in its Tail. https://symantec-enterprise-blogs.security.com/blogs/threat-intelligence/shamoon-destructive-threat-re-emerges-new-sting-its-tail, 2018. [Online; Accessed December 2023].

[39] The Times of Israel. Iran cyberattack on Israel's water supply could have sickened hundreds – report. https://www.timesofisrael.com/iran-cyberattack-on-israels-water-supply-could-have-sickened-hundreds-report/, 2020. [Online; Accessed October 2023].

[40] ICS Advisory. APT Cyber Tools Targeting ICS/SCADA Devices. https://www.cisa.gov/uscert/ncas/alerts/aa22-103a, 2022. [Online; Accessed October 2023].

[41] Nathan Brubaker, Keith Lunden, Ken Proska, Muhammad Umair, Daniel Kapellmann Zafra, Corey Hildebrandt, and Rob Caldwell. INCONTROLLER: New State-Sponsored Cyber Attack Tools Target Multiple Industrial Control Systems. https://www.mandiant.com/resources/blog/incontroller-state-sponsored-ics-tool, 2022. [Online; Accessed October 2023].

[42] Dragos Inc. CHERNOVITE's PIPEDREAM Malware Targeting Industrial Control Systems (ICS). https://www.dragos.com/blog/industry-news/chernovite-pipedream-malware-targeting-industrial-control-systems/, 2022. [Online; Accessed October 2023].

[43] Josh Homan, Sean McBride, and Rob Caldwell. IRONGATE ICS Malware: Nothing to See Here...Masking Malicious Activity on SCADA Systems. https://www.fireeye.com/blog/threat-research/2016/06/irongate_ics_malware.html, 2016. [Online; Accessed December 2023].

[44] Marina Krotofil and Chris Sistrunk. "MAN-IN-THE-SCADA": Anatomy of Data Integrity Attacks in Industrial Control Systems. In *BlackHat ASIA*, 2017.

[45] Gabriel Sanchez. Man-In-The-Middle Attack Against Modbus TCP Illustrated with Wireshark. https://www.sans.org/white-papers/38095/, 2017. [Online; Accessed December 2023].

[46] Igor Nai Fovino, Andrea Carcano, Marcelo Masera, and Alberto Trombetta. Design and Implementation of a Secure Modbus Protocol. In *International conference on critical infrastructure protection*, pages 83–96, 2009.

[47] Eli Biham, Sara Bitan, Aviad Carmel, Alon Dankner, Uriel Malin, and Avishai Wool. Rogue7: Rogue Engineering Station Attacks on S7 Simatic PLCs. In *BlackHat USA*, 2019.

[48] Thomas Wiens. S7comm Wireshark dissector plugin. https://sourceforge.net/projects/s7commwireshark/, 2021. [Online; Accessed December 2023].

[49] Saranyan Senthivel, Irfan Ahmed, and Vassil Roussev. SCADA network forensics of the PCCC protocol. *Digital Investigation*, 22:S57–S65, 2017.

[50] Yangyang Geng, Yuqi Chen, Rongkuan Ma, Qiang Wei, Jie Pan, Jingyi Wang, Peng Cheng, and Qingxian Wang. Defending cyber–physical systems through reverse-engineering-based memory sanity check. *IEEE Internet of Things Journal*, 2022.

[51] Defense Use Case. Analysis of the Cyber Attack on the Ukrainian Power Grid. *Electricity Information Sharing and Analysis Center*, 2016.

[52] Anonymous Anonymous. Demo videos of memory manipulation against S7-315 PLC of Siemens. https://youtu.be/Nafn8-6FG4I, 2023. [Online; Accessed December 2023].

[53] Anonymous Anonymous. Demo videos of memory manipulation against MicroLogix 1400 PLC of Rockwell. https://youtu.be/ikRrnUEXIWM, 2023. [Online; Accessed December 2023].

[54] Ravi Kumar. An overview of the global PLC industry and its dynamics. https://medium.com/world-of-iot/95-an-overview-of-the-global-plc-industry-and-its-dynamics-102ae2cfc0b4#:~:text=Summary%20of%20the%20PLC%20market&text=Siemens%20have%20the%20largest%20share,31%25%20in%20the%20world's%20market, 2019. [Online; Accessed December 2023].

[55] David Formby, Milad Rad, and Raheem Beyah. Lowering the Barriers to Industrial Control System Security with GRFICS. In *2018 USENIX Workshop on Advances in Security Education (ASE 18)*, 2018.

[56] Karl Johan Åström, Tore Hägglund, and Karl J Astrom. *Advanced PID control*. 2006.

[57] Bruno Sinopoli, Luca Schenato, Massimo Franceschetti, Kameshwar Poolla, Michael I Jordan, and Shankar S Sastry. Kalman Filtering With Intermittent Observations. *IEEE transactions on Automatic Control*, 49(9):1453–1464, 2004.

[58] Guoshen Yu and Guillermo Sapiro. Statistical Compressed Sensing of Gaussian Mixture Models. *IEEE Transactions on Signal Processing*, 59(12):5842–5858, 2011.

[59] Brian DO Anderson and John B Moore. *Optimal Filtering*. Courier Corporation, 2012.

[60] SIMATIC. Programming Guideline for S7-1200/S7-1500. https://support.industry.siemens.com/cs/document/90885040/programming-guideline-for-s7-1200-s7-1500?dti=0&lc=en-WW, 2018. [Online; Accessed December 2023].

[61] Karl-Heinz John and Michael Tiegelkamp. *IEC 61131-3: Programming Industrial Automation Systems*. 2010.

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

[63] Schneider Electric. User Manual for PLC Programming with CoDeSys 2.3. https://www.ee.pw.edu.pl/~purap/PLC/manuals/m07590333_00000000_1en.pdf, 2010. [Online; Accessed December 2023].

[64] Randal E Bryant, O'Hallaron David Richard, and O'Hallaron David Richard. *Computer systems: a programmer's perspective*, volume 2. 2003.

[65] Schneider Electric. How To Download and Upload Your Latest Source Code User Guide. https://download.schneider-electric.com/files?p_enDocType=User+guide&p_File_Name=EIO0000002536.00.pdf&p_Doc_Ref=EIO0000002536, 2017. [Online; Accessed December 2023].

[66] Lennart Ljung. System identification. In *Signal analysis and prediction*, pages 163–173. Springer, 1998.

[67] Peter S Maybeck. *Stochastic Models, Estimation, and Control*. Academic press, 1982.

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

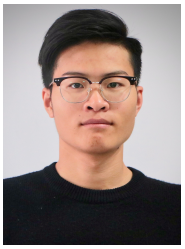
[69] Anastasis Keliris and Michail Maniatakos. ICSREF: A Framework for Automated Reverse Engineering of Industrial Control Systems Binaries. In *Symposium on Network and Distributed System Security (NDSS)*, 2019.

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

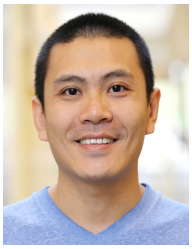
[71] Siemens AG. Programming with STEP 7. https://cache.industry.siemens.com/dl/files/825/109751825/att_933142/v1/STEP_7_-_Programming_with_STEP_7.pdf, 2017. [Online; Accessed December 2023].

[72] Rockwell Automation. MicroLogix 1400 Programmable Controllers Reference Manual. https://literature.rockwellautomation.com/idc/groups/literature/documents/rm/1766-rm001_-en-p.pdf, 2023. [Online; Accessed December 2023].

[73] Schneider Electric. Modicon M221 Logic Controller User Guide. https://download.schneider-electric.com/files?p_Doc_Ref=EIO0000000976&p_enDocType=User+guide&p_File_Name=M221_UserGuide_EN_EIO0000000976.03.pdf, 2023. [Online; Accessed December 2023].



Zeyu Yang received the Ph.D. degree in Control Science and Engineering in 2022, from Zhejiang University, Hangzhou, China. He is currently a postdoctoral researcher at the College of Control Science and Engineering, Zhejiang University. His research interests include cyber-physical system security and industrial control system security.



Liang He (S'09-M'12-SM'17) is currently an Assistant Professor at the University of Colorado, Denver, CO, USA. He worked as a Research Fellow with the University of Michigan, Ann Arbor, during 2015 – 2017. His research interests include CPS, IoT, and mobile computing.

Peng Cheng (M'10) received the B.E. degree in Automation, and the Ph.D. degree in Control Science and Engineering in 2004 and 2009 respectively, from Zhejiang University, Hangzhou, China. Currently he is Professor with College of Control Science and Engineering, Zhejiang University. His research interests include networked sensing and control, cyber-physical systems, and control system security.

Jiming Chen (M'08-SM'11-F'18) received B.Sc degree and Ph.D degree both in Control Science and Engineering from Zhejiang University. He was a visiting researcher at University of Waterloo from 2008 to 2010. Currently he is Professor with College of Control Science and Engineering, Zhejiang University. His research interests include IoT, networked control, and wireless networks.