DuAtt: A Dual-Layer Attestation Scheme for PLC-Based Industrial Internet of Things

Syed Owais Athar, *Student Member, IEEE*, Muhammad Naveed Aman, *Senior Member, IEEE*, and Biplab Sikdar, *Fellow, IEEE*

***Abstract*—Industrial Internet of Things (IIoT) applications span a wide range of critical infrastructure sectors. Therefore, ensuring efficiency and security is critical in the IIoT given their widespread impact and vulnerability to multifaceted attacks. System integrity can be compromised by malicious modifications in the program/software running on IIoT devices. Although programmable logic controllers (PLCs) are widely used in the IIoT, most of the existing attestation techniques exhibit computational complexity, leading to not only higher demand for resources but also a lack of versatility required to effectively detect and mitigate malicious modifications in PLCs. To resolve this issue, we propose a novel dual-layer attestation technique (DuAtt), that aims to comprehensively validate the integrity of PLC-based IIoT devices at two levels of abstraction. The first layer leverages a physical process-based model to detect abnormal behavior by creating a physical model using linear dynamic state-space (LDS). The physical model then allows us to estimate the system state based on current sensor measurements and PLC output. After estimating the system states, a mismatch in the estimated values and the actual measurements signifies an anomaly. To complement, based on the feedback from the first layer, a second layer runs a software-based attestation check on the PLC's program integrity. Thus, the first layer runs more frequently and requires fewer resources, which reduces the overall computational burden. Whereas, the more complex second layer is responsible for carrying out further investigation when an anomaly is detected. The effectiveness of DuAtt is evaluated using experiments on actual hardware, showing a lower verification delay by up to 37.31% in contrast to the current methods and provides a 100% detection rate against output as well as program modification attacks, with quick restoration to the original state in addition to the added benefits of scalability and adaptability.**

***Index Terms*—Attestation, industrial control systems (ICSs), industrial cyber–physical systems (ICPSs), Industrial Internet of Things (IIoT), linear dynamic state-space (LDS), programmable logic controller (PLC).**

Received 21 April 2025; revised 23 June 2025; accepted 1 July 2025. Date of publication 16 July 2025; date of current version 9 October 2025. This work was supported in part by A*STAR, CISCO Systems Pte. Ltd., USA, and in part by the National University of Singapore under its Cisco-NUS Accelerated Digital Economy Corporate Laboratory under Award I21001E0002. *(Corresponding author: Muhammad Naveed Aman.)*

Syed Owais Athar is with the School of Computing, University of Nebraska-Lincoln, Lincoln, NE 68588 USA, and also with the Department of Electronic Engineering, Balochistan University of Information Technology, Engineering and Management Sciences, Quetta 87300, Pakistan (e-mail: sathar2@huskers.unl.edu).

Muhammad Naveed Aman is with the School of Computing, University of Nebraska-Lincoln, Lincoln, NE 68588 USA (e-mail: naveed.aman@unl.edu).

Biplab Sikdar is with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore 119077 (e-mail: elebisik@nus.edu.sg).

I. Introduction

THE INTEGRATION of digital intelligence with physical processes in industrial cyber–physical systems (ICPSs) has transformed essential infrastructure sectors, including manufacturing, energy, transportation, and healthcare [1]. By coordinating the smooth interplay between digital control systems and tangible components, these systems maximize output and efficiency. But this convergence also brings with it previously unheard-of security risks, since Industrial Internet of Things (IIoT) is vulnerable to many types of cyber attacks that jeopardize public safety, operational continuity, and data integrity [2].

To control and regulate operations across critical infrastructure sectors, the industrial control system (ICS) primarily relies on programmable logic controllers (PLCs) [3]. With advancements in Industry 4.0 and the IIoT, PLCs are still an integral part of ICS [4]. This is due to their adaptability in automating complex processes, increasing efficiency, and providing real-time control [5], [6]. Because of their extensive integration and connection to the Internet, PLCs are more than ever vulnerable to cyberattacks. So, maintaining the integrity of PLC programming is crucial. Adversarial behaviors, which range from unlawful access and manipulation to sophisticated attacks on PLC programming, pose serious risks to the stability and dependability of industrial operations [7]. Once entered, attackers can alter control logic, add malicious code to the PLC's programming, or change set-points. This might lead to the defined processes acting irregularly, which could damage equipment, cause production failures, or interrupt vital operations in critical infrastructure sectors [8].

The Stuxnet attack on Iran's nuclear program [3] and BlackEnergy crimeware targeting Ukraine's power and the train systems [9] show that IIoT systems are increasingly complex. This makes them attractive targets for malicious actors. Significant risks are associated with security breaches in the ICPS and IIoT; these risks go beyond financial losses and can put lives at risk and interfere with vital services [10]. Therefore, strengthening the IIoT against cyber threats is essential to protecting critical infrastructure and upholding social order.

The attestation process of a PLC involves verifying the integrity and authenticity of the PLC's program to detect unauthorized modifications or cyberattacks [11]. Throughout this procedure, various PLC program segments are routinely examined by a reliable authority to make sure they correspond to the authentic version. Both continuous and periodic attestation

are possible; for older systems, periodic attestation is more effective. The attestation maintains high detection accuracy while minimizing computing overhead by randomly checking certain code portions. By doing this, you can guarantee that vital industrial processes that depend on the PLC stay safe without overtaxing the system's capacity.

There are several attestation approaches available to verify the integrity of IIoT. However, their scalability, resource efficiency, and environmental adaptation are frequently limited especially in regard to their application to PLCs. Moreover, these methods ignore any flaws in the underlying physical processes in favor of computationally complex software-based attestation [8]. Normally IIoT functions in dynamic and diverse environments with real-time interactions between physical and digital components [12]. Accordingly, traditional security procedures are frequently insufficient to handle the intricate problems presented by IIoT [13]. Expanding interconnected devices and the widespread use of legacy systems further complicate the security environment, calling for creative solutions to counter new threats.

Emerging threats that target reconfigurable devices like PLCs are particularly difficult to counter using current attestation methodologies because of their complexity and lack of scalability [14], [15]. Furthermore, existing attestation methods for PLC-based IIoT systems often suffer from higher computational complexity due to continuous runtime monitoring. They also lack integration with the physical processes they aim to secure. Also, these methods frequently require the verification of the entire PLC program, leading to excessive computational overhead and delayed threat detection. These limitations highlight the critical demand for an attestation method that not only adapts to real-time threats but also operates efficiently in constrained environments, i.e., minimizes verification workload while maintaining accuracy.

To address the limitations of current PLC attestation techniques, a dual-layer attestation technique (DuAtt) is proposed. DuAtt thoroughly verifies a PLC program's integrity by combining a lightweight physical process-based model for anomaly detection with software-based targeted attestation. This tiered approach improves detection capabilities and lessens computing load by dividing the verification effort across physical and logical layers. Hence, strengthening IIoT against physical attacks as well as protecting against malicious tampering to safeguard critical operations.

The motivation for leveraging a DuAtt stems from the need to balance resource efficiency with comprehensive detection capabilities. Single-layer approaches either focus solely on software verification or physical anomaly detection, which can lead to either high computational costs or incomplete security coverage. The dual-layer approach in DuAtt combines these methods to ensure lightweight but robust validation.

The main contributions of this work are as follows.

1) A novel DuAtt for PLC-based IIoTs called DuAtt, to effectively address the tradeoff between computational complexity and detection accuracy.
2) A lightweight physical process-based model to detect anomalies by using real-time sensor measurements and PLC outputs.

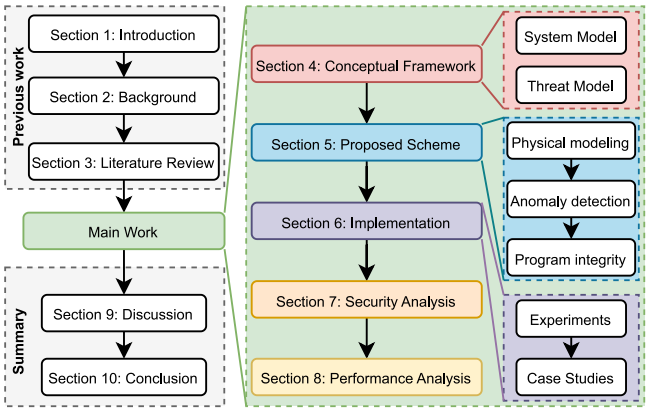

Fig. 1. Structure and organization of this research work.

3) A targeted software-based attestation mechanism for program integrity, activated after an anomaly is detected to further investigate and detect malicious tampering.
4) Real-world hardware-based experiments to validate the effectiveness and efficiency of DuAtt.

Fig. 1 presents the complete organization of this article.

II. Background

The following section will present the essential background required for ICS, PLCs, and physical modeling.

A. Industrial Control System

A wide range of hardware and software components are combined to form ICSs that are used to monitor and manage industrial processes. Transportation, utilities, energy production, manufacturing, and other essential infrastructure sectors all depend on these systems [16]. The following are the components of an ICS.

1) Supervisory Control and Data Acquisition: To facilitate real-time data capture and centralized management across a variety of industries, supervisory control and data acquisition (SCADA) systems are necessary for the remote monitoring and controlling of industrial processes and systems [17].

2) Programmable Logic Controller: PLCs are specialized computing devices that are commonly employed to automate control activities in industrial settings. PLCs provide exact control over industrial processes by executing control logic encoded in proprietary or standardized languages like ladder logic (LAD), functional block diagram (FBD), instruction list (IL), sequential function charts (SFCs), and structured text (ST) [18].

3) Field Devices and Associated Hardware: Field devices serve as the interface between PLCs and the actual environment. These devices include sensors, actuators, and other hardware elements. They gather input from the actual world, transform it into digital signals, and then use the control orders they get from PLCs to start physical processes.

4) IEC 61131 International Standard: The architecture, general principles, and rules for PLCs in industrial automation systems are described in the IEC 61131-1:2003 standard [19]. Likewise, by defining the programming languages and

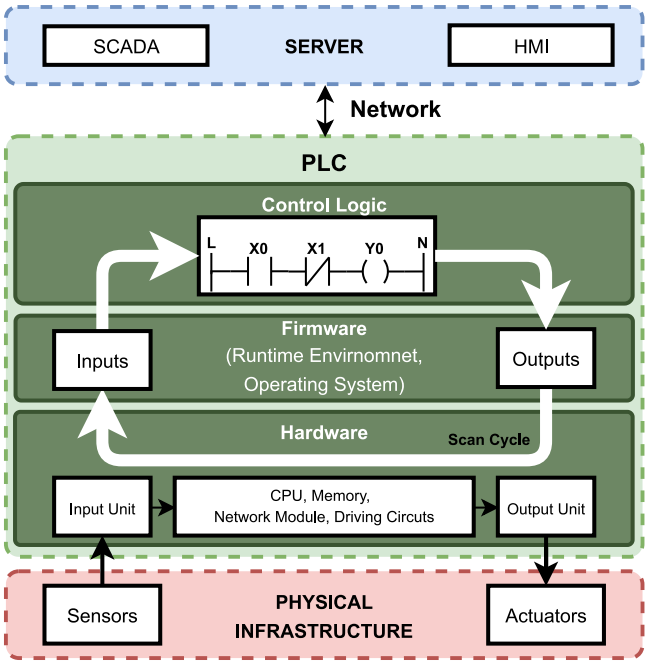

Fig. 2. Architecture of a PLC.

environments, the IEC 61131-3:2013 standard ensures interoperability and consistency between various PLC systems and manufacturers [20].

B. Architecture of PLC

PLCs are composed of hardware and software elements that facilitate their operation. The input and output (I/O) modules, memory, communication interfaces, programming environment, and central processing unit (CPU) are all included in this [18]. In industrial applications, control system design, programming, and maintenance require an understanding of PLC architecture presented in Fig. 2.

C. Linear Dynamic State-Space Modeling of Physical Systems

A mathematical approach called linear dynamic state-space (LDS) modeling is used to explain how dynamic systems behave in the time domain [21]. The following equations can be used to represent the dynamics of the system, where these sets of equations are usually obtained either using Einstein's or Newton's Laws or using system identification [22]:

$$\dot{x}(k+1) = Ax(k) + Bu(k) + \alpha e(k) \quad (1)$$

$$y(k) = Cx(k) + Du(k) + \beta e(k) \quad (2)$$

where $x(k), u(k), y(k)$, and $e(k)$ are state vectors representing internal states, inputs, outputs, and disturbances at time t, respectively. Likely, A, B, C, and D are matrices defining system dynamics and input–output relationships, while α and β are scaling factors that modulate the influence of the disturbances $e(t)$ on the state evaluation and output, respectively. Some commonly used methods for system identification are listed in Table I.

TABLE I
COMPARISON OF ESTIMATION TECHNIQUES FOR LDS MODELS

Estimation Method	Estimation Accuracy
Prediction Error Methods (PEM) [23]	Very High
Maximum Likelihood Estimation [24]	Very High
Subspace Identification (SIM) [25]	High
Least Squares (LS) Methods [26]	High
Kalman Filter-Based Estimation [27]	Moderate to High
Bayesian Estimation Techniques [28]	Moderate
Machine Learning-based Approaches [29]	Variable

Note: "High" estimation accuracy corresponds to approximately 80-90%, "Very High" refers to accuracy above 90%, and "Moderate" indicates an accuracy range of around 60-80%.

D. Prediction Error Method

The prediction error method (PEM) is a widely used method for estimating model parameters by minimizing the difference between the predicted and actual output of a system. In case of state-space models, it adjust the parameters of the matrices A, B, C, and D, along with the noise factors α and β in (1) and (2), in order for the model to accurately mimic the behavior observed in the actual system. Given the recorded input–output data, the output prediction $\hat{y}(k|\theta)$ is computed as

$$\hat{y}(k|\theta) = C\hat{x}(k|\theta) + Du(k) + \beta e(k) \quad (3)$$

where $\hat{x}(k|\theta)$ represents the predicted states and θ encompasses all the parameters to be estimated. The prediction error $e(k, \theta)$ is the difference between the actual output $y(k)$ and the predicted output $\hat{y}(k|\theta)$

$$e(k, \theta) = y(k) - \hat{y}(k|\theta). \quad (4)$$

By adjusting the model parameters, PEM attempts to minimize the total prediction error over all data points. This can be expressed as the minimization of the following cost function:

$$V(\theta) = \frac{1}{N} \sum_{k=1}^{N} e(k, \theta)^2 \quad (5)$$

where N is the total number of data points. The optimization process iteratively adjusts θ to minimize $V(\theta)$, thereby refining the matrices A, B, C, D, and noise terms α and β in state (1) and (2).

PEM can be used to estimate the system's parameters with a high degree of precision, guaranteeing that the model accurately captures the dynamics of the system as it exists in real time using the data that has been gathered.

E. Dependency Graphs

A dependency graph is a pivotal tool for providing a better understanding of the intricate relationships between various components of a software system. A dependency graph is a directed graph in which components are represented by nodes and dependencies between these components are illustrated by directed edges [34]. In software engineering, this structure is crucial for managing complex processes, optimizing system performance, and ensuring data integrity. These graphs are frequently and systematically explored using depth-first traversal (DFT) for deep searches, dependency resolution, and cycle detection. Dependency graphs and DFT work particularly well

together to help understand dependencies and how they affect system behavior. In this article, we utilize dependency graphs in a novel way to reduce the computational complexity of the proposed attestation technique. Consider a scenario where a traffic light control system operates based on input from pedestrian crossing buttons and vehicle presence sensors. The dependency graph maps the sequence of these dependencies and highlights critical paths that influence the PLC's output decisions. This allows the identification of potential attack points. For instance, if an attacker manipulates the vehicle presence sensor data, the dependency graph would help trace how this impacts the control logic, leading to potential traffic congestion or unsafe pedestrian crossings. Similarly, if an attacker manipulates the pedestrian switch data, the dependency graph would help trace the potential cascading effects on the control logic, making it easier to detect anomalies.

III. Literature Review

A. Security Vulnerabilities in PLCs

PLCs are vulnerable to a variety of possible attacks, which are frequently taken advantage of by weak authentication procedures or network weaknesses [35]. Attackers may be able to modify control logic or insert malicious code into the system by taking advantage of these vulnerabilities to obtain unauthorized access [36]. Furthermore, data integrity may be jeopardized by intercepting and altering information through flaws in PLC communication protocols [37]. Man-in-the-middle attacks are a serious threat because they give adversaries the ability to intercept and change data that is transferred between PLCs, which can result in erroneous decisions and disruptions in industrial operations [38].

Denial of Service (DoS) attacks pose a concern to PLC security as well since they can cause PLCs to become unresponsive and experience downtime, which can interfere with regular industrial operations [39], [40]. Common attack vectors include firmware vulnerabilities [41] and malicious software injection [42], which let hackers disrupt operations and maybe cause harm by inadvertently introducing malicious code into PLC systems [43]. Intentional manipulation of PLC hardware presents further hazards, which may result in harm to equipment or disruptions to industrial operations [44]. Furthermore, dangers might result from intentional or unintended actions, such as illegal modifications, deliberate sabotage, or unintentional errors, by people with access to PLC systems [45]. Physical assaults that pose serious risks to the integrity and functionality of PLCs include rootkit [46], input/output pin control exploitation [47], and different injection attacks [48]. These attacks exacerbate the security issues already encountered by PLCs. An overview of these multidimensional threats and vulnerabilities targeting PLCs is illustrated in Fig. 3.

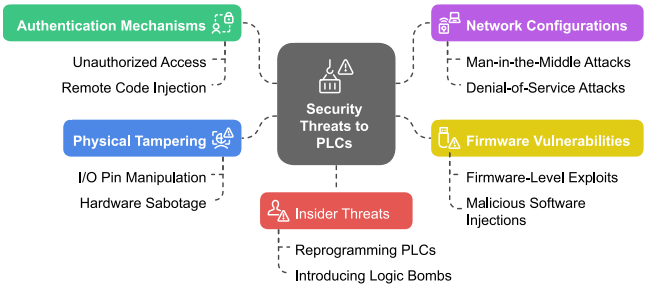

Fig. 3. Overview of security threats targeting PLCs in IIoTs.

B. Existing Attestation Techniques

In recent years researchers have been investigating different approaches to verify the integrity of PLCs within the IIoT. A number of notable strategies have been proposed, where each technique provides a different perspective on PLC attestation.

PLCDefender [30], a hybrid remote attestation technique uses a physics-based model to maintain the integrity of PLC control behavior. Although PLCDefender shows encouraging results in identifying and averting a variety of PLC attack scenarios, it requires more testing and verification in a variety of industrial environments, which is necessary to determine its robustness and scalability.

Similarly, by combining control process validation and remote software attestation, PAtt [31] offers a solution for confirming the integrity of PLCs. Through the integration of memory state integrity measures into control operations, PAtt allows for the remote verification of control logic integrity using the generated sensor traces. Nevertheless, more investigation and verification in industrial settings with diverse operational circumstances and assault scenarios could be necessary to ensure practical viability and real-world application.

Another similar approach by Chen et al. in [32], presents a workable method for attesting the input/output behavior of PLC programs that is based on privacy-preserving black box models. Although this method shows encouraging results in identifying code modifications and predicting actuator states from inputs, its efficacy needs to be evaluated in terms of scalability and adaptation to various industrial situations.

Cao et al. [33] suggested using a software-based remote attestation (SBRA) technique to confirm the reliability of IoT devices without requiring hardware changes. In contrast to physics-based or hybrid approaches, this method uses only software-based verification techniques, such as memory filling at attestation time, a delayed observation mechanism, and a reputation-based strategy to optimize verification overhead. Despite improving detection accuracy, this method is not resilient to physical manipulation or side-channel assaults and is only effective against software integrity violations. Its adaptability in IIoT applications is further limited by the fact that it ignores real-time operational irregularities in industrial settings.

Table II presents a comparison of the key limitations of existing attestation techniques, such as high computational complexity, limited scalability, and poor compatibility with legacy PLCs. These shortcomings can hinder practical deployment in real-world industrial environments. Among the evaluated techniques, PLCDefender [30] and PAtt [31] offer strong accuracy but are hindered by high latency and poor adaptability to legacy systems. Whereas, NN Prediction [32], and SBRA [33] shows better scalability but fall short in

TABLE II
COMPARISON BETWEEN ATTESTATION TECHNIQUES

Technique	Computational Complexity	Time Latency	Detection Accuracy	Scalability	Compatibility
PLCDefender [30]	Very High	Extremely High	98% (for process anomalies)	Very Low	Low
PAtt [31]	High	Very High	95% (for control logic)	Low	Low
NN Prediction [32]	Moderate	Medium	88%	Moderate	Moderate
SBRA [33]	Moderate	High	92% (for software integrity)	Moderate	Moderate
DuAtt [Proposed]	Low	Low	100% for program and output anomalies	High	High

Note: In this table, if ζ denotes the computational complexity, then "Very High" is defined as $\zeta \geq O(N^3)$, "High" as $O(N^2) \leq \zeta < O(N^3)$, "Moderate" as $O(NlogN) \leq \zeta < O(N^2)$, and "Low" as $\zeta < O(NlogN)$. Similarly, if τ represents total time latency, including communication and processing delays, then "Very High" latency refers to $\tau > 4\ ms$, "High" to $2ms < \tau \leq 4\ ms$, "Medium" to $1.5\ ms < \tau \leq 2\ ms$, and "Low" to $\tau \leq 1.5\ ms$. Compatibility describes the effort required to deploy a technique on legacy PLCs without significant hardware modification. A "High" rating implies minimal adaptation is needed, "Moderate" indicates limited configuration or trade-offs, and "Low" suggests substantial changes or overhead are required for deployment.

handling physical attacks and full-layer coordination. DuAtt addresses these shortcomings by providing a balanced and scalable solution with minimal computation overhead. It also ensures compatibility with legacy PLCs while maintaining high detection accuracy.

C. Security Gaps in PLC Attestation and Proposed Solutions

Existing techniques on PLC attestation suffer from one or more of the following problems and limitations.

1) The scalability and adaptability of current PLC attestation techniques are limited and less thoroughly evaluated.
2) Real-time threat detection and response capabilities are constrained by higher computational overhead.
3) ICS vulnerabilities are made worse by a lack of defense against physical assaults, which exposes vital infrastructure to possible disruptions and damage.
4) The efficacy of existing systems is limited because they often require full program analysis. This reduces their capacity to quickly identify and respond to specific attack scenarios.
5) The delayed detection of security breaches frequently compromises real-time monitoring and response allowing malicious activities to remain undetected for prolonged periods of time.
6) Some of the existing attestation techniques such as PLCDefender [30] and PAtt [31] adopt a layered approach by incorporating both physical modeling and software integrity checks. However, these layers operate independently, without dynamic coordination or triggering between them. This lack of interaction limits their ability to efficiently and precisely isolate anomalies.

This shows that adoptable and more proactive security measures are required as vulnerabilities that allow attackers to evade detection mechanisms compromise the authenticity and integrity of IIoT hardware as well as software. The proposed technique addresses the limitations of current PLC attestation methods by the following.

1) Integrating both physical process modeling and software-based attestation to enhance the adaptability and scalability issues.
2) Reducing the computation burden and complexity by employing a two-layered framework.
3) Providing comprehensive protection through continuous monitoring from both cyber and physical attacks, ensuring authenticity and integrity, by employing a dual-layered attestation technique that monitors and defends against both program and output attacks.
4) Improving system efficiency through targeted comparison of PLC program sections to detect malicious tampering.
5) Employing real-time threat detection and response capabilities, using anomaly detection in conjunction with PLC program attestation to immediately identify any malicious tampering.
6) Introducing a true dual-layer architecture where the physical anomaly detection layer dynamically triggers the software verification layer. It not only enhances efficiency by using a dependency graph-based targeted attestation but also verifies only the segments of the PLC program mapped to anomalous outputs. This cross-layer coordination reduces the verification scope while maintaining accuracy.

DuAtt introduces a transformative dual-layered attestation design that surpasses conventional methods by combining anomaly detection with software-based integrity verification. Unlike single-layer attestation schemes that either prioritize physical process validation or software attestation, DuAtt integrates both approaches to form a synergistic defense mechanism. This hybrid strategy not only broadens the scope of cyber-physical attack detection but also optimizes resource efficiency, ensuring a more comprehensive and adaptive security solution for industrial IoT environments.

IV. CONCEPTUAL FRAMEWORK AND ASSUMPTIONS

A. System Model

The architecture and components in the system model are shown in Fig. 4. We use the example of PLCs to interconnect and control several traffic signals. These PLCs are connected to a gateway, which performs anomaly detection and transmits data to the associated server. The gateway facilitates data interchange and communication by operating as an interface between the traffic lights and the server. The traffic control SCADA system is housed on the server and is responsible for monitoring and managing the operation of the traffic lights. The proposed anomaly detection unit (ADU) is installed on the gateway and continuously monitors for anomalies. Whereas the proposed program verification unit (PVU) is responsible for PLC program attestation and is installed on the SCADA server.

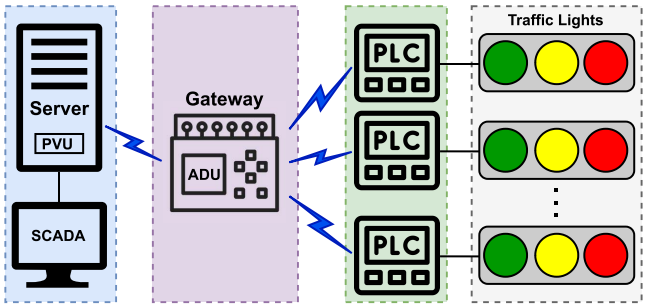

Fig. 4. System model.

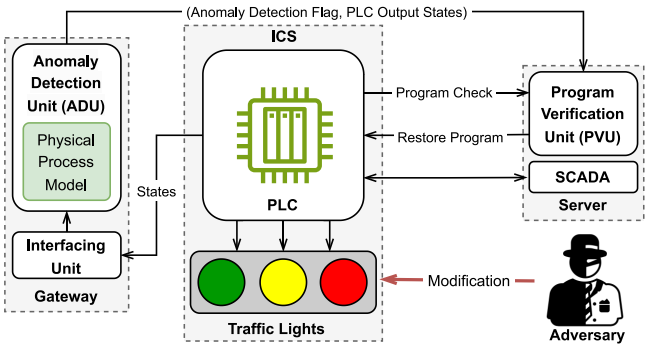

Fig. 5. Overview of DuAtt illustrating the dual-layer attestation process and data flow.

B. Threat Model

The threat model describes possible attackers and attack situations that the suggested attestation plan seeks to prevent. It includes a range of physical and cyber threats, such as malicious code injection, tampering with control logic, illegal access, and any cyber-physical assaults on the controller's output. In this article, we assume that the adversary:

1) can eavesdrop, replay, drop, and modify messages sent/received by the PLC on the Internet [11], [49];
2) has knowledge about the PLC's system architecture;
3) can gain physical access to a PLC, leading to direct manipulation of PLC I/Os;
4) may gain privileged access to embed malicious logic within the PLC ladder program; and
5) cannot tamper with the PVU. The PVU (verifier) is considered a trusted authority and is responsible for verifying the integrity of IIoT components.

DuAtt detects such threats through a combination of anomaly-driven targeted comparisons and selective verification of PLC program segments. For this article, we reference the NIST SP 800-82 standard threat model for ICSs [3], which outlines potential cyber-physical threats and mitigation strategies that align with DuAtt's design.

C. Assumptions

We make the following assumptions.

1) To provide real-time monitoring and validation, the suggested attestation approach presumes access to sensor measurements and control instructions from the IIoT components [31], [50], [51].
2) PLCs are not physically protected and are vulnerable to physical tampering attacks.
3) It is assumed that field devices, PLCs, and other IIoT components adhere to industry standards for communication protocols and data exchange [52], [53].
4) The suggested approach functions on the premise that under normal working conditions, the physical processes under the control of the IIoT display predictable behavior. This allows for the detection of anomalies that may be signs of physical or cyberattacks.

V. Proposed Attestation Scheme

A. Overview of DuAtt

An overview of the proposed (dual-layer attestation scheme) DuAtt with all its elements is illustrated in Fig. 5, where the prover (ADU) and verifier (PVU) are responsible for detecting anomalies and program integrity verification, respectively.

The inputs from the PLC are fed to a physical model that estimates outputs using the ADU, which are then compared with real-time outputs of the PLCs. In case of a mismatch an anomaly is detected and the PVU is notified. The PVU then runs a targeted software-based attestation algorithm to detect malicious modifications in the PLC program. If the PVU detects a change in the PLC program, it restores the PLC to its original state. Otherwise, it halts the system and alarms the operators to a possible physical attack.

The physical process-based model in DuAtt captures PLC-based IIoT dynamics. To ensure accurate monitoring and integrity validation, it uses system identification techniques like subspace identification (SIM) [25], PEMs [23], and maximum likelihood estimation (MLE) [24]. These methods rely on known physical laws and process parameters. Given the popularity of PEM [23] to accurately model and analyze system dynamics, this article uses this method for physical process model estimation. This enables reliable comparison between actual and predicted outputs, where any deviations and mismatches indicate tampering or anomalies.

For real-time data acquisition in model estimation, we logged data in MATLAB workspace through an ATmega2560 microcontroller-based board [54] as a data acquisition interface using the Arduino Explorer Application [55] directly from the PLC outputs. The recorded input/output data is then imported into the MATLAB workspace and after removing the means, the LDS is estimated using the PEM method. For the discrete event traffic light signal estimated model accuracy is validated by comparing it against the actual program output. For this purpose, the SIMULINK PLC Coder [56] was utilized. The outputs of both the PLC coder block and state-space implementation were compared, where both output plots were identical indicating the accuracy of the estimated model.

B. Anomaly Detection Unit (ADU)

The ADU in DuAtt works as a prover and is responsible for detecting any abnormal behavior. It makes use of Algorithm 1 to identify any abnormal behavior. The LDS model, which depicts the anticipated behavior of the IIoT (see Section V-A) is used by the ADU to estimate the output of the system. The

Algorithm 1: Anomaly Detection Algorithm for ADU

```
1  Procedure AnomalyDetection
2      Inputs: Sensor_data, PLC_output;
3      Outputs: Anomaly_flag;
4      Anomaly_flag ← False;
       // Load LDS model for estimation
5      LDS_model ← Load_LDS_model();
       // Run LDS model to estimate the
          expected PLC output
6      Estimated_output ←
        RunLDSModel(Sensor_data, LDS_model);
       // Comparing estimated and PLC
          actual outputs to detect anomaly
7      if Estimated_output ≯ PLC_output then
8          Anomaly_flag ← True;
           // Call Algorithm 2
9          ProgramVerification(PLC_program, PLC_output);
10         return Anomaly_flag;
11     end if
12 end Procedure
```

Algorithm 2: Program Verification Algorithm for PVU

```
1  Procedure ProgramVerification
2      Inputs: PLC_program, PLC_output;
3      Authentic_program ← Load_authentic_program();
       // Call Algorithm 3
4      Mod_flag ←
        TargetedComparison(PLC_program, PLC_output);
5      if (Mod_flag) then
           // Replace tampered program with
              authentic PLC program
6          Update_PLC_memory(Authentic_program);
7      else
           // Halt PLC due to output attack
8          Halt_PLC();
9      end if
10 end Procedure
```

ADU runs the LDS model using incoming sensor data, and a possible anomaly is indicated if the PLC's real output deviates from the LDS model's expected output. In these situations, the PVU is notified by the ADU to verify the program's integrity by raising a flag.

C. Program Verification Unit (PVU)

The PVU is employed as a verifier and is responsible for validating PLC program integrity. Upon receiving a signal from the ADU (signaling the discovery of an anomaly), the PVU verifies the integrity of the PLC's program. Algorithm 2 is employed to carry out this verification procedure.

First, the PVU determines whether an abnormality has been found and waits for a signal from the ADU. If that is the case, it loads the real program and compares it with the one that is now operating on the PLC. To identify malicious tampering and pinpoint its location, the legitimate program is compared with the current running program on PLC, and the status of PLC outputs is forwarded to Algorithm 3. The PVU restores the PLC's program memory with the genuine program if any differences are discovered between any section of the two programs, suggesting a possible compromise of the PLC's program. The PVU halts the PLC's operation to avoid any additional damage if it finds that the programs are identical, indicating a possible output assault.

The role of PVU is to identify malicious tampering. Current methods involve comparing a legitimate reference of the PLC program with the active program running on the PLC. However, this approach is not only time-consuming but also lacks precision, because it fails to pinpoint specific ladder logic networks or sections of the PLC program that have been maliciously modified. The challenge is to identify the dependencies of these networks in the PLC ladder diagram that correspond to a change in the physical output of the controller.

To overcome this challenge and selectively compare and identify the malicious section of PLC code, Algorithm 3 is utilized by the PVU. A dependency graph that maps different parts of the PLC code to specific output states based on Ladder diagram networks is pre-established. An example of a dependency graph is shown in Fig. 6. The portion of a PLC program code responsible for emulating a single ladder rung is considered a part that may or may not logically control a physical output. In many cases multiple rungs make up the logic to control a single PLC output. Whereas, a single output can depend on multiple rungs, hence, modifying any such section can change the output status.

Now, when an anomaly is detected by the ADU it sends the output states along with the anomaly detection flag to the PVU. The anomaly-driven targeted comparison algorithm is designed to selectively compare selective parts of the PLC program on the basis of detected anomalies to enhance efficiency and resolve the scalability problem for large-scale systems with multiple subsystems. The flow of events is illustrated in Fig. 7, and the steps of the algorithm are as follows.

1) The dependency graph G is created offline to establish a mapping between the PLC outputs and the corresponding parts of the PLC program that control them.
2) Upon detecting an anomaly in any output, the relevant part(s) of the program associated with the affected output are identified as anomaly nodes.
3) A DFT is applied starting from anomaly nodes and gathering connected parts of the program to see which portion might be impacted.
4) The traversal results are intersected to isolate the parts of the program that are responsible
5) To identify the specific areas of the program that might be causing the anomaly, the outcomes of these traversals are intersected.
6) The isolated parts are then compared with parts of the legitimate program to detect any modification.
7) The PVU Algorithm 2 is notified whether any modifications were found to take appropriate action.

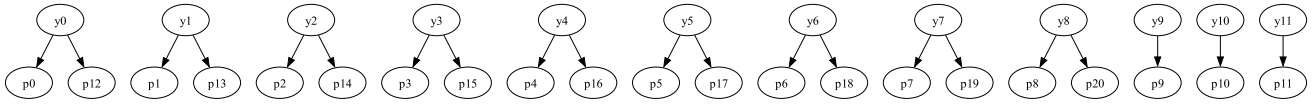

Fig. 6. Dependency graph of a 4-way traffic light controller mapping PLC program parts to output states.

Algorithm 3: Anomaly-Driven Targeted Comparison

Procedure *TargetedComparision*
  Inputs: PLC_Program, PLC_output;
  Outputs: Mod_flag;
  Load dependency graph G;
  Map all parts of PLC program $\{p_0, p_1, \ldots\}$ to corresponding PLC outputs $\{y_0, y_1, \ldots\}$; Use PLC_output to identify affected output status $\{u_0, u_1, \ldots\}$;
  `// Find affected nodes`
  foreach p_i *in* G **do**
    if $G[p_i] = u_i$
      Add p_i to anomaly_nodes;
    end if
  end
  `// DFT from each anomaly node`
  foreach *anomaly node* u *in anomaly_nodes* **do**
    Initialize a list τ;
    Perform DFT starting from u and add visited nodes to τ;
    Add τ to traversal_results;
  end
  Initialize $\zeta = \tau_1$ (first traversal result);
  `// Find common nodes`
  foreach *traversal result* τ_i *in traversal_results* **do**
    Compute $\zeta = \zeta \cap \tau_i$;
  `// Find isolated parts`
  foreach *traversal result* τ_i **do**
    Compute isolated_parts as $\zeta = \zeta \setminus \tau_i$;
  Mod_flag $\leftarrow$ False;
  `// Compare legitimate and running program parts`
  foreach *program part* p_i *in isolated_parts* **do**
    if $F_{\text{legit}}[p_i] \neq F_{\text{running}}[p_i]$
      `// Modification detected`
      Mod_flag $\leftarrow$ True ;
    end if
  end
  Return Mod_flag;
end Procedure

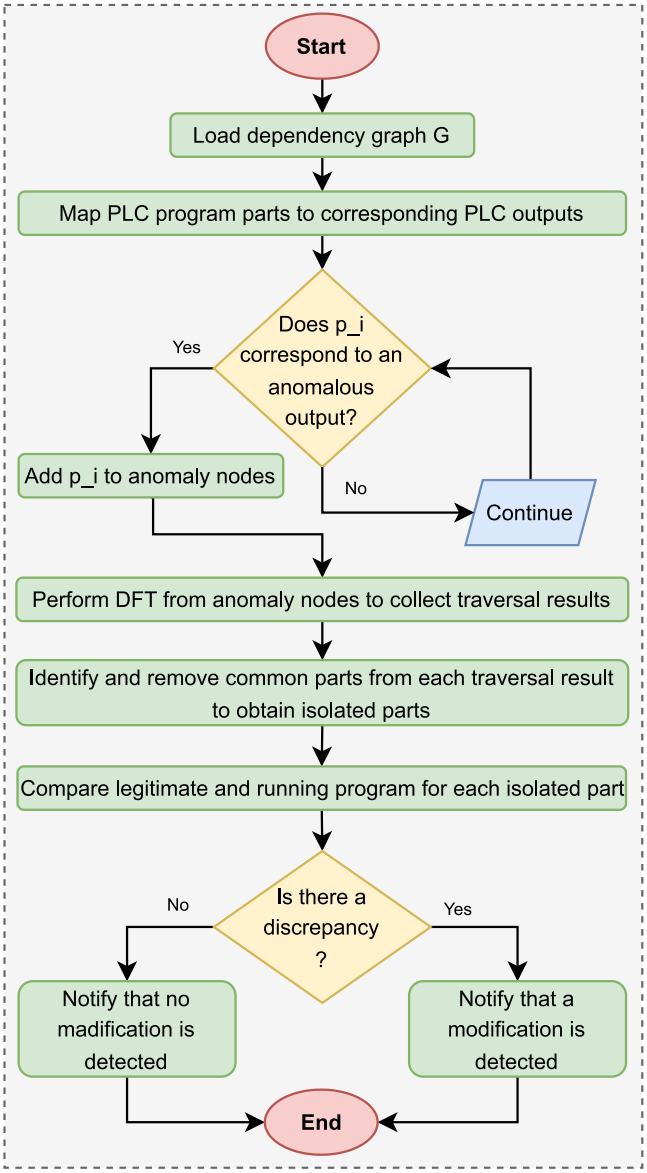

Fig. 7. Flowchart of Algorithm 3, illustrating the process from creating a dependency graph to detecting anomalies, performing DFT, and notifying the calling algorithm of any modifications.

D. Anomaly Detection and Attestation for Program Integrity

A systematic dual-layer approach is adapted to ensure integrity checking and responding to potential threats. A flowchart in Fig. 8 represents the complete process, where to simulate the expected behavior of the IIoT, a physical process-based model is first used. This model is then used to compare the expected output of the system with the actual output. Upon detecting any anomalies that indicate a departure from the anticipated behavior, the PLC's program integrity is immediately verified. Potential security risks are alerted when malicious tampering is detected. The system is promptly shut down, and the program is reset to its initial condition in order to reduce risks and prevent any further damage.

VI. Implementation and Experimentation

Two major components make up the implementation and testing part of this research, describing the actual experimental

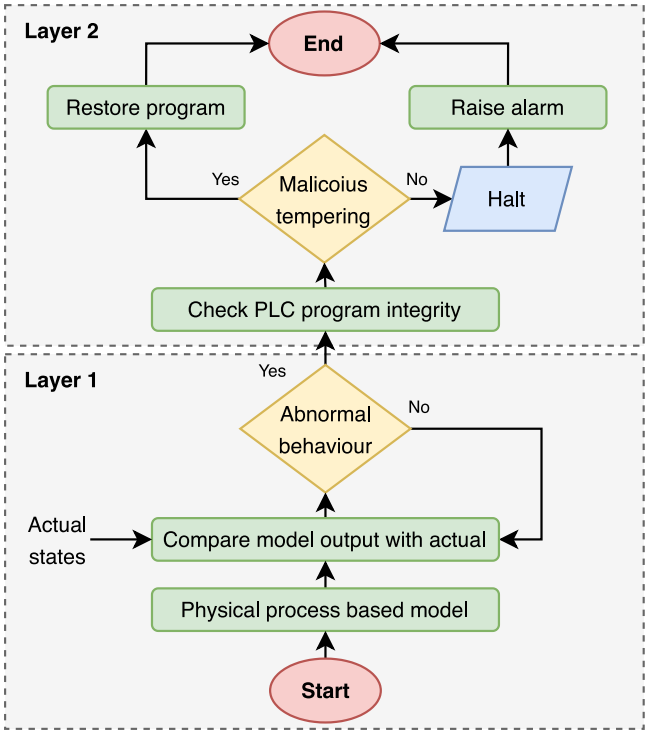

Fig. 8. Flowchart illustrating the dual-layered attestation technique and the program restoration process.

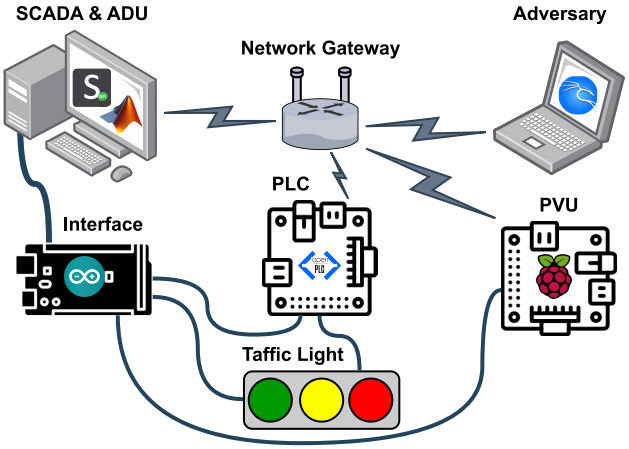

Fig. 9. Block diagram demonstrating all components used in the practical implementation and their connectivity.

setting and investigating case studies to evaluate the effectiveness of the proposed DuAtt. Fig. 9 demonstrates the interconnection of all components for practical implementation, and Fig. 10 shows the actual experimental setup.

A. Description of Experimental Setup

1) Physical Model Estimation: Referring to the set of (1) and (2), the matrices A, B, C, D and E for the LDS model are obtained by using the PEM [23] in MATLAB from the real-time recorded data from the PLC. A graphical representation of the estimated and predicted responses for one cycle of one pole of a traffic light signal is shown in Fig. 11, where we

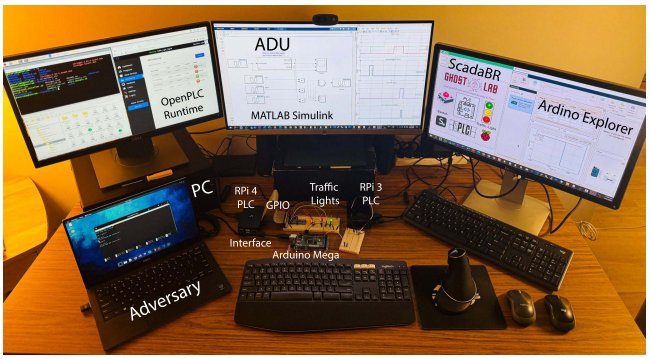

Fig. 10. Experimental setup.

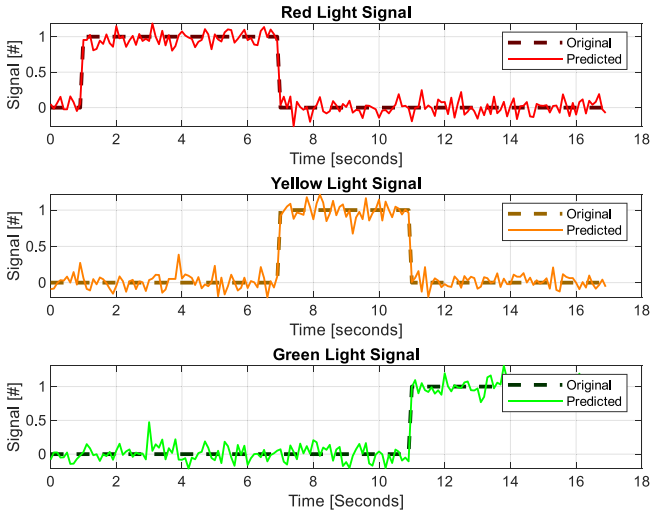

Fig. 11. Comparison of the original and predicted traffic light signals for red, yellow, and green traffic lights. The Y-axis label indicates binary logic levels.

observe a close fit between the estimated physical model and the real-time data

$$A = \begin{bmatrix} 1.419 & 0.5084 & 1.833 & -0.9943 \\ -0.1926 & 1.384 & 0.3368 & 0.2844 \\ -0.6795 & -1.06 & -1.931 & 0.8908 \\ 0.4585 & -1.154 & -0.9614 & 0.4267 \end{bmatrix} \tag{6}$$

$$B = \begin{bmatrix} -18.75 & -18.86 & -9.853 \\ -12.07 & -15.23 & -9.378 \\ 36.83 & 41.15 & 24.98 \\ 27.07 & 29.69 & 18.59 \end{bmatrix} \tag{7}$$

$$C = \begin{bmatrix} 0.09592 & 0.1166 & 0.01334 & 0.09403 \end{bmatrix} \tag{8}$$

$$D = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} \tag{9}$$

$$K = \begin{bmatrix} 2.594 \\ 3.055 \\ -5.588 \\ -6.018 \end{bmatrix}. \tag{10}$$

2) Raspberry Pi as PLC: The Raspberry Pi 4b is used as a PLC for experimental verification of the proposed DuAtt attestation scheme. This is done by utilizing the OpenPLC project [57], a well-known open-source platform. The OpenPLC project comes with an OpenPLC editor that helps in writing and compiling PLC codes with the

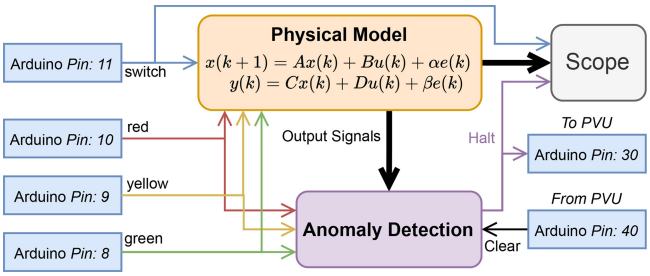

Fig. 12. Implementation of ADU in MATLAB/SIMULINK along with Arduino support package [58] and Arduino explorer [55].

IEC 61131-3 standard programming techniques. The resulting code runs on the Raspberry Pi's local host in the OpenPLC runtime. The compiled program is saved in a ST ".st" file format. Through the use of the OpenPLC runtime, real-time control of input–output sensors and actuators is possible by utilizing the Raspberry Pi's GPIO pins. Augmented with four LED lights with a bunch of current-limiting resistors and a spring-loaded push switch, the Raspberry Pi is configured to control a single-pole traffic light signal.

The Raspberry Pi was chosen as the PLC because of its unmatched open-source capabilities, which allow for flexible functionality and the implementation of many attack scenarios that are required for this research.

3) Anomaly Detection: The ADU in DuAtt was implemented in MATLAB SIMULINK. MATLAB was selected because of its strong multidomain analysis capabilities. The MATLAB SIMULINK ADU implementation is shown in Fig. 12, and with the help of the Arduino support package for Arduino hardware [58], this configuration enables the real-time integration of the state-space model and the physical signals from the PLC. When a physical switch is pressed, the state model and the physical process both start running simultaneously. This mechanism makes sure that the physical process and the ADU are synchronized. In case of an anomaly, the ADU generates a halt signal, sent directly to an output pin of the microcontroller board through which this signal is then sent directly to both the PLC and PVU, respectively.

4) Program Attestation: After an abnormal behavior is detected by the ADU, the PLC goes into halt mode and the PVU is activated. The PVU was implemented on another Raspberry Pi 3 that was already connected to the PLC (Raspberry Pi 4 running the OpenPLC runtime). The PVU executes a Python code with reference to Algorithm 2 and waits for the halt signal it receives through its GPIO pin. The PVU then copies the current program under execution from the PLC and compares it with a legitimate copy. In case malicious tampering is detected, the PVU restores to the original program and signals the ADU to remove the halt signal, and as soon as the halt signal is removed the PLC starts working normally again. Otherwise, if the PVU does not detect any malicious tampering in the PLCs program, we can infer that this is a physical output attack. In such a situation the PVU informs the ADU that no program anomaly has been detected, where the restoration of normal operation necessitates a physical examination of the PLC to verify hardware integrity.

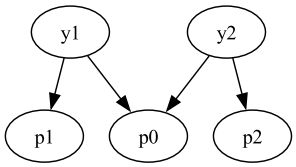

Fig. 13. Dependency graph of a single pole traffic light signal.

B. Case Studies

To validate the operation and effectiveness of DuAtt, the following two case studies were conducted.

1) Program Modification Attack: In this case study, the attacker gains unauthorized access to the PLC and maliciously alters its code. This study aims to evaluate the system's effectiveness in identifying and counteracting unauthorized modifications while maintaining the PLC program's integrity. To launch a program modification attack, a Linux-based computer system was connected to the network. This system was then used to replace the instruction code file under execution with a modified program in the PLC leading to an altered output. The purpose of DuAtt is to detect this attack and restore the PLC to its original state.

Considering a single side traffic light signal for proof of concept, where both yellow and red lights corresponding to PLC outputs $y1$ and $y2$, respectively, are affected. The input, outputs, auxiliary elements, and timers associated with the PLC register for the traffic light signal are detailed in Table III. The preconstructed dependency graph illustrating the dependency of each PLC output on a specific part of the PLC program is shown in Fig. 13. In the dependency graph, the nodes $p1$ and $p2$ are identified as anomalies after performing a DFT

$$\tau_1 = p1, p0 \tag{11}$$

$$\tau_2 = p2, p0. \tag{12}$$

The intersection of the traversal paths then isolates the common code section, which may be responsible for the detected anomaly, i.e.,

$$\zeta = \tau 1 \cap \tau 2 = p0. \tag{13}$$

Further isolation reveals

$$\tau 1 - \zeta = p1 \tag{14}$$

$$\tau 2 - \zeta = p2. \tag{15}$$

Now that the potential malicious sections have been identified, the PVU compares the isolated sections of the legitimate code with the running program on the PLC, i.e., comparing the executable binaries

$$p1_{\text{ligitimate}} \neq p1_{\text{vunerable}} \rightarrow \text{modification detected} \tag{16}$$

$$p2_{\text{ligitimate}} \neq p2_{\text{vunerable}} \rightarrow \text{modification detected} \tag{17}$$

where $p_{\text{ligitimate}}$ represents the part of the legitimate PLC program and $p_{\text{vulnerable}}$ denotes the section of the malicious

TABLE III
ANNOTATION OF PLC REGISTER ELEMENTS FOR A SINGLE TRAFFIC LIGHT SIGNAL

Inputs	Label	Outputs	Labels
X0	Switch	Y0	Green light
X1	Halt signal	Y1	Yellow light
		Y2	Red light
Auxiliary	**Label**	**Timers**	**Labels**
M10	Start process	T0	6 s ON-Timer
M0, M1, M2	Timer status	T1	4 s ON-Timer
M20, M21, M22	SCADA control	T2	6 s ON-Timer

program running on the PLC. Thus, DuAtt successfully isolates and identifies the modified sections of the code. This approach significantly reduces the verification time and computational complexity by focusing only on the relevant parts of the program, unlike existing attestation techniques that require verification of the entire program, leading to increased computational overhead.

2) Output Modification Attack: This case study focuses on an attack in which the attacker tries to alter the PLC's output signals to cause incorrect control actions or deceptive feedback to the system. To ensure the accuracy and dependability of control operations, the system's ability to recognize and react to these output adjustments must be assessed. To mimic this attack, a Python script to directly control the Raspberry Pi's GPIO pin is executed on a Linux-based computer. A map of annotated PLC inputs, outputs, and other internal elements for a single traffic light signal is tabulated in Table III. DuATT aims to identify any unusual activity by thoroughly scanning the system and flagging potential security threats. If an output modification attack is detected, the operator physically inspects the PLC to confirm that the hardware is secure. Afterward, the GPIO pins are reset, and the PLC is restarted to bring it back to normal operation smoothly.

3) Backup Mechanism: In the event that both layers of the attestation system detect an anomaly and the program cannot be restored to a known safe state, DuAtt initiates a safety shutdown protocol that isolates the PLC from the network and alerts the supervisory control system. Additionally, the system triggers an automated rollback to the last verified safe program state where feasible, ensuring minimal operational disruption. This prevents further propagation of compromised signals or data while enabling rapid recovery.

VII. SECURITY ANALYSIS

In this section, we analyze the security of DuAtt against possible attacks, compliance with the fundamental security principles, and the capability to function normally in adversarial environments.

A. Security Objectives

The primary security objective of DuAtt is to maintain the integrity of a PLC program and its output signals. The dual-layer approach improves the integrity check by utilizing.

1) Physical Process-Based Anomaly Detection: A PLC's outputs are continuously compared to an LDS model by the ADU to identify any deviations that might indicate program tampering or adversarial influence.

2) Targeted Software Attestation: After an anomaly is detected, the PVU selectively verifies the relevant PLC program segments rather than the entire program. This reduces the computational overhead and also enhances efficiency.

Furthermore, DuAtt ensures integrity in two ways.

1) Any unauthorized modifications in the control logic are identified through dependency graph-based targeted comparison. This results in the immediate detection of program manipulation attacks.
2) Malicious changes to PLC output states that attempt to evade software-based attestation are caught via real-time physical model validation. This minimizes the risk of undetected tampering and reinforces the attestation process.

Additionally, to protect the anomaly detection mechanism from tampering, DuAtt relies on the inherent security of the ADU and PVU, as stated in the system model and threat model. Since both units are trusted entities, any modifications to the anomaly detection logic are inherently prevented from unauthorized interference. Before execution, the ADU's operational state is verified by the PVU, ensuring consistency and reliability in detecting anomalies. Furthermore, redundant verification mechanisms involving multiple ADUs provide an added layer of security by cross-checking and confirming integrity, reducing the risk of a single point of failure.

B. Resilience Against Attacks

DuAtt is designed to detect the most common cyber and physical attacks that threaten the system integrity of PLC-based IIoT environments [8], [11]. The following attacks are explicitly considered.

1) Program Manipulation Attacks: An attacker may attempt to alter PLC ladder logic or ST programs to modify process behavior covertly. To successfully carry out such an attack, the attacker must either inject malicious instructions into the PLC's control logic or modify existing program sequences without being detected. However, DuAtt can detect these types of attacks by comparing execution states with the reference PLC program in the PVU. Any deviations between the expected and actual execution states trigger an anomaly detection response, ensuring that unauthorized modifications are identified. In turn, these types of malicious modifications can be isolated, and the legitimate PLC program may be restored.

Attackers might inject malicious code into the PLC runtime environment to execute unauthorized instructions or attempt control-flow hijacking. By modifying execution pathways or injecting malicious routines, attackers may attempt to override normal PLC operations and execute unauthorized actions. DuAtt employs anomaly-driven targeted verification, ensuring that any unauthorized changes to code segments are detected and reversed before they impact the system.

Second, an attacker may attempt to bypass the anomaly detection mechanism by altering the PLC's input–output dependencies or manipulating sensor data to mask the changes

TABLE IV
COMPARISON OF SECURITY FEATURES AMONG DIFFERENT TECHNIQUES

Feature	DuAtt	PLCDefender [30]	PAtt [31]	NN Prediction [32]	SBRA [33]
Anomaly Detection	Physical Process Modeling + Software Attestation	Physical Model	Control Logic Integrity	NN-based Prediction	Software-based checksum
Tamper detection	Yes	No	Partial	No	No
Replay Attack Detection	Yes	No	No	No	Partial
Proxy Attack Detection	Yes	No	Partial	No	No
Physical Output Signal Attack Detection	Yes	No	No	No	No

Note: The term "Partial" indicates that the technique provides limited or conditional support for the feature. It might identify particular instances, specific cases, or types of attacks, but it does not provide comprehensive protection in all pertinent situations.

in control logic. However, DuAtt's physical process-based anomaly detection ensures that any unexpected system behavior is flagged, regardless of whether the attacker manipulates direct program execution or attempts to spoof sensor feedback. By continuously monitoring the expected versus actual system responses, the ADU DuAtt effectively detects and isolates inconsistencies that arise from such tampering attempts.

2) Output Signal Modification Attacks: An attacker may intercept and alter PLC output signals to cause physical process disruptions. To successfully execute such an attack, the attacker must manipulate the PLC's output signals by injecting false control commands [1] or interfering with the communication between the PLC and actuators. This can lead to unsafe system behavior, equipment malfunctions, or even operational failures. However, DuAtt detects such attacks by continuously monitoring the expected versus actual physical process states using the ADU. Any deviation between the LDS model's predicted outputs and real-time sensor readings triggers an anomaly alert, prompting the PVU to verify the PLC's execution state. This ensures that unauthorized modifications to output signals are promptly detected before they cause widespread system disruptions.

3) Replay Attacks: Attackers may attempt to replay previously valid attestation responses to evade detection by feeding a legitimate historical response to the verifier. This makes the system falsely believe that the PLC is in an untampered state. To execute such an attack, the adversary must capture and reuse a past attestation response that was accepted by the verifier. However, DuAtt detects replay attempts by employing a randomized selection of attested program segments rather than relying on static integrity checks. Each attestation request dynamically selects different portions of the PLC program for verification. This ensures that repeated use of old responses is ineffective. Also, the randomized attestation mechanism ensures that attackers cannot rely on previously captured responses to bypass integrity verification.

4) Proxy Attacks: An attacker may attempt to impersonate a legitimate PLC through a proxy device to deceive the attestation mechanism and bypass detection. To successfully execute such an attack, the adversary must intercept and relay attestation queries between the verifier and a compromised or rogue device while modifying responses to appear legitimate. However, DuAtt detects such proxy-based deception through anomaly-driven state verification. Since the ADU cross-verifies the real-time sensor data with the expected process behavior, any discrepancies arising from proxy-based manipulation will be detected as an anomaly. Additionally, since attestation responses are tied to dynamic execution states rather than static program checks, proxy attacks that attempt to spoof legitimate execution states fail when the verifier cross-checks the PLC's responses against real-time operational data.

Furthermore, an adversary may attempt to physically replace the legitimate PLC hardware with a rogue device, executing a proxy-based attack at the hardware level. In this scenario, the attacker would swap out the original PLC and connect a manipulated proxy device that relays attestation requests while executing unauthorized operations in the background. However, such an attack is infeasible in DuAtt's security model because the underlying industrial environment adheres to strict backend protocols and device authentication mechanisms [59], which ensure that only authorized hardware can function within the system. In addition to this, since DuAtt relies on real-time process verification rather than solely software attestation, any discrepancies caused by an unauthorized hardware swap would be immediately detected through inconsistencies in the physical process model and sensor feedback. This ensures that runtime PLC replacement attempts fail as they cannot replicate the expected real-time system behavior [60].

To contextualize the security strengths of DuAtt, we compare its security properties with other prominent attestation techniques, including PLCDefender [30], PAtt [31], NN-based Prediction [32], and SBRA [33]. The comparison focuses on key security properties such as anomaly detection mechanisms, temper detection mechanisms, replay attack prevention, proxy attack resistance, and physical output signal attack resistance (see Table IV).

The comparison highlights how DuAtt integrates multiple security properties, ensuring both computational efficiency and strong attack resistance. Unlike PLCDefender and PAtt, which primarily rely on physical process models or control logic verification, DuAtt introduces a hybrid approach that enhances security without significantly increasing computational complexity.

C. Limitations and Future Enhancements

While DuAtt provides strong integrity guarantees by integrating both physical process-based anomaly detection and software attestation, it assumes that:

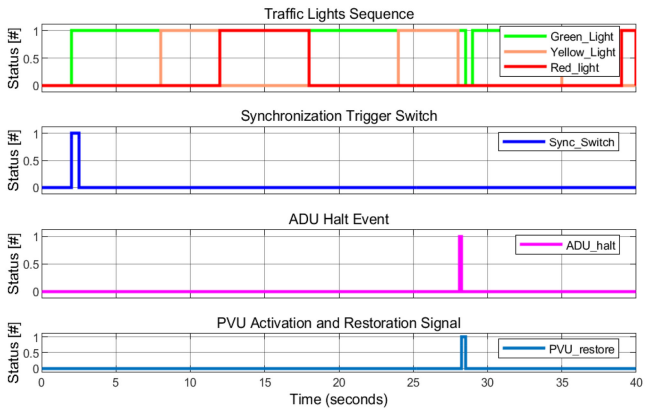

Fig. 14. SIMULINK scope illustrating the sequence of events including the hardwired traffic light signals, synchronization signal from the physical switch, ADU halt signal, and PVU attestation and restoration signal. The *Y*-axis label indicates binary logic levels.

1) the PVU remains trusted and is not compromised by insider threats;
2) the LDS model accurately captures physical process dynamics and does not require frequent reconfiguration; and
3) attackers cannot gain persistent low-level access to hardware interfaces without detection.

Future developments may focus on refining DuAtt's attestation mechanism to increase the precision of anomaly detection and improve adaptability to various industrial environments.

1) Enhancing anomaly detection techniques by incorporating adaptive thresholds based on real-time operational data.
2) Extending attestation capabilities to heterogeneous PLC architectures to support a wider range of industrial systems.
3) Optimizing system scalability and response efficiency to accommodate large-scale deployments with minimal overhead.

DuAtt through its dual-layered architecture establishes a robust, scalable, and efficient attestation framework tailored for IIoT environments, ensuring real-time integrity verification while maintaining low computational costs.

VIII. Performance Analysis

The effectiveness and efficiency of DuAtt is assessed in this section.

A. Accuracy

Fig. 14 illustrates a program tempering attack, where, by using the trigger indicators in the SIMULINK scope, we can analyze the events. At time $t = 4.5$ s the synchronization switch is pressed and the green light is turned on. After completing its cycle of 6 s, the yellow light turned on for 4 s. However, when the red light turned on, it must have completed 6 s, but after completing 4 s, it turned on and the green light was immediately turned on. Right at this moment, the ADU detects an anomaly and signals a halt. Meanwhile, the PVU checks the program for integrity and also raises a flag after a second of the halt signal. Soon after restoring the PLC program, both the ADU halt and PVU flag go down, and the green light starts operating from the start and completes in a 6-s time duration, followed by the yellow light.

Likewise, in case of an output modification attack, the PLC operation is suspended after the PVU cannot find any malicious program modification. This continues until the operator secures the PLC and removes any malicious tampering with the output of the PLC. For both case studies, DuAtt achieves 100% detection accuracy in identifying tempering with the output of the program. Because anomalies are deterministically captured at the physical layer and validated through targeted attestation, the system reports no false positives or false negatives in the experimental evaluation.

TABLE V
Comparison of Execution Times and Improvement Across Techniques

Technique	DuAtt	[32]	[33]	[31]	[30]
File Size (KB)	32	32	32	32	32
Transfer Time (ms)	0.64	0.64	0.64	0.64	0.64
Processing Time (ms)	0.7	0.75	1.1	1.2	1.7
Total Time (ms)	1.34	1.39	1.74	1.84	2.34
Improvement (%)	-	3.73%	22.99%	37.31%	74.63%

TABLE VI
Comparison of Computation Complexity

Technique	Parameters	Computational Complexity
DuAtt	N: size of input data, p: parts of PLC program	$O(N+p)$
SBRA [33]	N: size of input data	$O(NlogN)$
NN Prediction [32]	N: size of input data, m: number of important inputs	$O(N\log N+m)$
PAtt [31]	N: size of input data (including nonce)	$O(N^2)$
PLCDefender [30]	N : size or dimensionality of the input data	$O(N^3)$

B. Computational Complexity

Comparing DuAtt to its predecessors, there are noticeable performance gains. The technique continually monitors system integrity while operating with a low processing overhead by utilizing a lightweight physical model-based anomaly detection algorithm.

To analyze DuAtt's response time we used a ST file for the single traffic light of 4 KB. Hence, the File Size $= 4$ KB $\times$ $8 = 32$ KB, where considering the typical Raspberry Pi Wi-Fi speed as 100 Mb/s. Therefore, the file transfer time can be computed as 0.32 ms. After a few milliseconds, the PVU copies the file from the PLC, does a comparison, and then replaces the running program on the PLC with the legitimate program. Hence, this is equivalent to two transactions totaling 0.64 ms. For a 4 KB structure text PLC instruction, the estimated processing times and propagation delays range from 0.65 to 0.7 ms. So the total time is the sum of transfer time and processing time, i.e., 0.64 ms + 0.7 ms $=$ 1.34 ms. Considering a Raspberry Pi using Wi-Fi and a 32 kb ST file, a comparison between DuAtt and existing state-of-the-art techniques is presented in Table V. We observe that DuAtt results in a verification time that is 3.73%, 22.99%, 37.31%, and 74.63% faster as compared to the techniques in [30], [31], [32], and [33], respectively.

TABLE VII
COMPUTATIONAL COMPLEXITY AND SCALABILITY ANALYSIS OF ATTESTATION TECHNIQUES

Technique	Complexity	Scalability Trend	Potential Bottlenecks	Load Balancing Feasibility
DuAtt	$O(N+p)$	Linear (Efficient)	Network congestion at >2000 devices	Easily distributed across multiple verifiers
SBRA [33]	$O(N \log N)$	Near-Linear	Increased processing time at high N	Partial, requires memory-efficient optimizations
NN Prediction [32]	$O(N \log N+m)$	Near-Linear	Extra complexity due to m inputs	Moderate, neural network optimization possible
PAtt [31]	$O(N^2)$	Quadratic (Inefficient)	Heavy processing demands at high N	Limited due to centralized verification
PLCDefender [30]	$O(N^3)$	Exponential	Severe bottlenecks beyond 1000 devices	Not scalable, requires significant resources

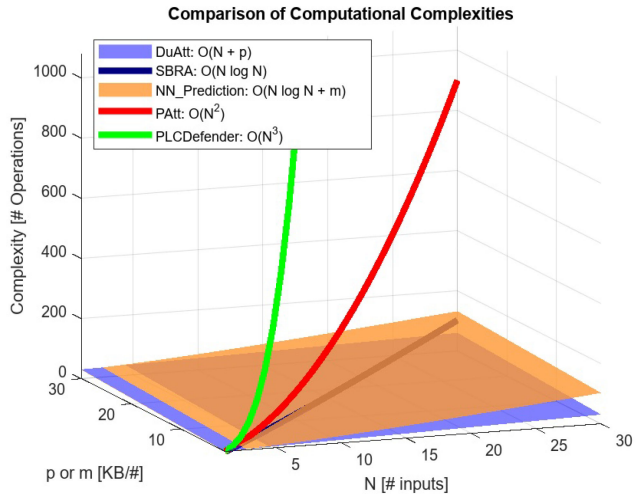

Fig. 15. Comparison of Computational Complexities: The plot illustrates the computational complexity of five techniques. The light blue plane represents the DuAtt method with $O(N+p)$, the dark blue line shows the SBRA with $O(N \log N)$, the orange plane corresponds to the NN-based prediction method with $O(N \log N+m)$, while the red and green lines depict the physics-based attestation (PAtt) and PLCDefender with complexities $O(N^2)$ and $O(N^3)$, respectively.

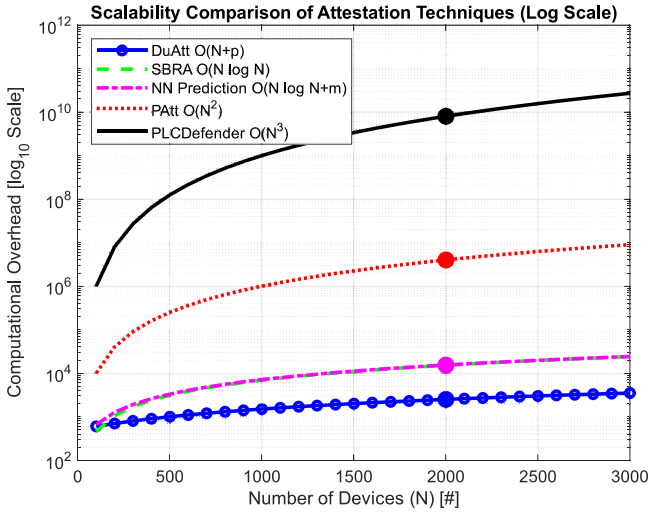

Fig. 16. Scalability comparison of attestation techniques on a logarithmic scale, where the computational overhead of DuAtt and alternative attestation methods (SBRA, NN-Based prediction, PAtt, and PLCDefender) are plotted against the number of devices N. DuAtt $O(N+p)$ shows the lowest overhead while maintaining efficient scalability, whereas PLCDefender $O(N^3)$ exhibits exponential overhead growth. Marked bottleneck points (at 2000 devices) highlight theoretical scalability limits, confirming DuAtt's advantage for large-scale IIoT applications.

The analysis of computational complexity shown in Table VI sheds light on the efficiency of different attestation methods. DuAtt has a linear complexity ($O(N+p)$), where p is the PLC program size and N is the size of the input data. On the other hand, current techniques like PLCDefender [30], physics-based attestation (PAtt) [31], code integrity attestation using NN-based prediction [32], and SBRA [33] show complexities of $O(N \log N+m)$, $O(N^2)$, $O(N^3)$, and $O(N \log N)$, respectively. We observe that DuAtt has significantly lower computational complexity as compared to conventional methods, particularly when dealing with situations involving big sensor datasets or intricate PLC codes, which makes it an attractive option for real-world applications. The plot in Fig. 15 demonstrates the increasing computational demands as the input size grows. We observe that DuAtt scales effectively to accommodate large-scale industrial systems with extensive sensor data and complex PLC programs.

C. Scalability

To study the scalability of DuAtt, we note that verification involves both a physical process-based anomaly detection followed by software-based attestation. Denoting the complexity of anomaly detection and software-based attestation by ζ_α and ζ_S, respectively, the computational burden to verify n devices ζ_T is given by

$$\zeta_T = N \cdot (\zeta_\alpha + \zeta_S) \tag{18}$$

$$= n\big[O(N+p)\big]. \tag{19}$$

In contrast, as the computational complexity of existing techniques is significantly higher than DuAtt, their scalability is proportionally impacted negatively. Fig. 16 shows the scalability of various techniques as N increases. We observe that DuAtt is highly scalable as compared to other techniques.

Moreover, the comparison of various methods in terms of computing complexity, scalability trends, and possible bottlenecks is also summarized in Table VII. This summary provides a broader context for interpreting the performance limitations of existing techniques as the device count increases, where DuAtt offers efficient load balancing across multiple verifiers for larger-scale IIoTs.

IX. DISCUSSION

A. System Modeling and Robustness

DuAtt relies on LDS models for physical process-based anomaly detection. This dependency can pose limitations when applied to highly nonlinear or complex systems where LDS models may fail to capture the full system dynamics. Extending the approach to nonlinear systems requires the

incorporation of machine-learning-based predictive models or hybrid estimation techniques, which may introduce additional computational costs. Future work aims to explore hybrid approaches that balance model accuracy with computational efficiency. However, these limitations will not affect the running time and computational complexity of the proposed model because LDS development is a one-time process and will continue to be the same until or unless there are physical changes to the system.

The accuracy of the anomaly detection mechanism is directly linked to the fidelity of the underlying system model. Since LDS modeling is a one-time identification process and remains valid unless physical modifications, like changes to sensors, actuators, or I/O configurations occur. This makes DuAtt robust and stable in long-term deployments. Especially in systems where the control structure remains unchanged. For more complex discrete-event industrial processes, petri net (PN) and interpreted PN (IPN) models may be considered to enhance modeling accuracy [61]. Note that ladder diagram PNs [62] and IPNs offer superior expressiveness in capturing event-driven logic and can aid in generating accurate LDS representations during the initial setup. Thus, expanding DuAtt's applicability to a broader class of PLC-based IIoT systems.

B. Hardware Platforms and Software Portability

The experimental setup utilized a Raspberry Pi 4B as the PLC emulator due to its open-source capabilities and GPIO support. However, the Raspberry Pi has certain limitations, such as lower processing power compared to industrial-grade PLCs, limited real-time performance , and robustness. Despite these limitations, the proof-of-concept demonstrates the feasibility of the proposed scheme in low-power environments. Scalability to real-world industrial settings involves the integration of heterogeneous PLCs with varying architectures. In practice, DuAtt can be ported to hardware platforms like Siemens S7 series and Allen-Bradley MicroLogix series by adapting to their memory access interfaces (e.g., address mapping and data block structure) and ensuring compatibility with their respective execution environments and programming standards [63], [64]. Future iterations can incorporate FPGA-based PLCs to address higher data loads and achieve faster anomaly detection with minimal latency.

The prototype implementation of DuAtt uses MATLAB to facilitate rapid development and testing of the proposed framework. However, for industrial deployment, the MATLAB implementation will be replaced with optimized Python and C++ modules that reduce computational overhead and eliminate the need for a MATLAB runtime environment. Empirical studies have shown that C++ implementations can be approximately ten times faster than MATLAB and nearly one hundred times faster than Python for computationally intensive tasks [65]. Preliminary tests with Python-based modules indicate similar performance with reduced resource consumption, but further optimization in C++ is expected to enhance real-time processing capabilities.

C. Compatibility With Legacy Systems

The practical deployment of DuAtt in industrial environments requires minimal modifications to existing legacy systems. Specifically, the addition of ADUs and PVUs can be integrated with SCADA systems via network interfaces without requiring major reconfiguration of the PLC program. This approach ensures compatibility with older PLC models while enhancing security. However, integrating DuAtt with proprietary PLC systems may require additional compatibility testing and validation.

D. Performance Benefits Over Existing Techniques

As compared to baseline algorithms such as PLCDefender and PAtt, DuAtt scales more efficiently due to its selective validation approach, which avoids full program verification unless anomalies are detected. This design choice allows it to handle larger networks of PLCs with minimal computational overhead.

E. Model Validation and Industrial Viability

The experimental results demonstrate DuAtt's superior detection accuracy and reduced latency. The high accuracy stems from the precise LDS model used for anomaly detection and the integrity verification mechanism. Lower latency is attributed to the selective program checks in the second layer, which only activate when anomalies are detected. Compared to existing works, this selective approach significantly reduces unnecessary computations, making DuAtt more efficient for real-time IIoT deployments.

Moreover, DuAtt has been evaluated using real-time data streams logged from IIoT devices (Field PLCs) in our experimental setup. This ensures that the framework can handle realistic workloads and industrial scenarios. While DuAtt provides comprehensive detection capabilities, it relies on accurate physical models for process-based verification, which may require updates for nonstationary processes. Additionally, initial calibration may introduce slight overhead during deployment.

X. Conclusion

This article presents DuAtt, a novel dual-layer attestation framework intended to improve the security and reliability of PLC-based IIoT. The proposed method addresses critical challenges including computational efficiency, adaptability, and scalability in resource-constrained industrial situations by integrating a lightweight physical model-based anomaly detection mechanism in combination with a software-based targeted attestation approach. This dual-layer approach ensures comprehensive protection against both cyber and physical threats, thereby protecting critical infrastructure sectors. DuAtt outperforms existing techniques by achieving faster response times and reduced computational overhead. As shown through its ability to promptly restore PLC integrity while maintaining 100% detection rate for both output and program manipulation attacks. Additionally, because of its scalability and versatility, the suggested technique can be implemented in a variety of industrial settings.

Future research directions involves advancing the anomaly detection technique with the inclusion of machine-learning-based algorithms. Moreover, utilizing digital twins in combination with physical model-based anomaly detection may further enhance the effectiveness and efficiency.

REFERENCES

[1] M. Usama and M. N. Aman, "Command injection attacks in smart grids: A survey," *IEEE Open J. Ind. Appl.*, vol. 5, pp. 75–85, 2024.

[2] H. Kayan, M. Nunes, O. Rana, P. Burnap, and C. Perera, "Cybersecurity of industrial cyber-physical systems: A review," *ACM Comput. Surveys*, vol. 54, no. 11S, pp. 1–35, 2022.

[3] K. Stouffer et al., "Guide to industrial control systems (ICS) security," Inf. Technol. Lab., Nat. Inst. Stand. Technol., Gaithersburg, MD, USA, Rep. NIST SP-800-82, 2011.

[4] M. A. Sehr et al., "Programmable logic controllers in the context of industry 4.0," *IEEE Trans. Ind. Informat.*, vol. 17, no. 5, pp. 3523–3533, May 2021.

[5] Y. Peng, P. Liu, and T. Fu, "Performance analysis of edge-PLCs enabled Industrial Internet of Things," *Peer-to-Peer Netw. Appl.*, vol. 13, pp. 1830–1838, Jun. 2020.

[6] R. Langmann and M. Stiller, "The PLC as a smart service in industry 4.0 production systems," *Appl. Sci.*, vol. 9, no. 18, p. 3815, 2019.

[7] A. Serhane, M. Raad, R. Raad, and W. Susilo, "Programmable logic controllers based systems (PLC-BS): Vulnerabilities and threats," *SN Appl. Sci.*, vol. 1, pp. 1–12, Jul. 2019.

[8] Z. Wang, Y. Zhang, Y. Chen, H. Liu, B. Wang, and C. Wang, "A survey on programmable logic controller vulnerabilities, attacks, detections, and forensics," *Processes*, vol. 11, no. 3, p. 918, 2023.

[9] "Blackenergy and Quedagh: The convergence of crimeware and APT attacks," White Paper, F-Secure Labs, London, U.K., 2016.

[10] G. Gori, L. Rinieri, A. Melis, A. Al Sadi, F. Callegati, and M. Prandini, "A systematic analysis of security metrics for industrial cyber–physical systems," *Electronics*, vol. 13, no. 7, p. 1208, 2024.

[11] W. Alsabbagh and P. Langendörfer, "Security of programmable logic controllers and related systems: Today and tomorrow," *IEEE Open J. Ind. Electron. Soc.*, vol. 4, pp. 659–693, 2023.

[12] I. Behnke and H. Austad, "Real-time performance of industrial IoT communication technologies: A review," *IEEE Internet Things J.*, vol. 11, no. 5, pp. 7399–7410, Mar. 2024.

[13] M. Wang, Y. Sun, H. Sun, and B. Zhang, "Security issues on Industrial Internet of Things: Overview and challenges," *Computers*, vol. 12, no. 12, p. 256, 2023.

[14] A. S. Banks, M. Kisiel, and P. Korsholm, "Remote attestation: A literature review," 2021, *arXiv:2105.02466*.

[15] M. Usama, M. N. Aman, and B. Sikdar, "Run-time self attestation of FPGA based IoT devices," *IEEE Internet Things J.*, vol. 11, no. 20, pp. 33406–33417, Oct. 2024.

[16] D. Sullivan, E. Luiijf, and E. J. Colbert, *Components of Industrial Control Systems*. Cham, Switzerland: Springer, 2016.

[17] R. L. Krutz, *Securing SCADA Systems*. Hoboken, NJ, USA: Wiley, 2005.

[18] W. Bolton, *Programmable Logic Controllers*. London, U.K.: Newnes, 2015.

[19] *IEC Webstore—Water Automation, Water Management, Smart City*, IEC Standard 61131-1:2003, 2003. [Online]. Available: https://webstore.iec.ch/publication/4550

[20] *IEC Webstore—Water Automation, Water Management, Smart City*, IEC 61131-3:2013, 2013. [Online]. Available: https://webstore.iec.ch/publication/4552

[21] Y. Li, T. C. Green, and Y. Gu, "Descriptor state space modeling of power systems," *IEEE Trans. Power Syst.*, vol. 39, no. 4, pp. 5495–5508, Jul. 2024.

[22] K. J. Åström and P. Eykhoff, "System identification—A survey," *Automatica*, vol. 7, no. 2, pp. 123–162, 1971.

[23] T. B. Schön, A. Wills, and B. Ninness, "System identification of nonlinear state-space models," *Automatica*, vol. 47, no. 1, pp. 39–49, 2011.

[24] J. Vlček, "Maximum likelihood estimation of parameters in state-space models," *IFAC Proc. Vol.*, vol. 34, no. 20, pp. 105–109, 2001.

[25] W. Favoreel, B. De Moor, and P. Van Overschee, "Subspace state space system identification for industrial processes," *J. Process Control*, vol. 10, nos. 2–3, pp. 149–155, 2000.

[26] M. Verhaegen and V. Verdult, *Filtering and System Identification: A Least Squares Approach*. Cambridge, U.K.: Cambridge Univ. Press, 2007.

[27] L. Simpson, A. Ghezzi, J. Asprion, and M. Diehl, "An efficient method for the joint estimation of system parameters and noise covariances for linear time-variant systems," in *Proc. 62nd IEEE Conf. Decis. Control (CDC)*, 2023, pp. 4524–4529.

[28] C. P. Robert et al., *The Bayesian Choice: From Decision-Theoretic Foundations to Computational Implementation*, vol. 2. Cham, Switzerland: Springer, 2007.

[29] A. Chiuso and G. Pillonetto, "System identification: A machine learning perspective," *Annu. Rev. Control, Robot., Auton. Syst.*, vol. 2, no. 1, pp. 281–304, 2019.

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

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

[32] Y. Chen, C. M. Poskitt, and J. Sun, "Code integrity attestation for PLCs using black box neural network predictions," in *Proc. 29th ACM Joint Meeting Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng.*, 2021, pp. 32–44.

[33] J. Cao, T. Zhu, R. Ma, Z. Guo, Y. Zhang, and H. Li, "A software-based remote attestation scheme for Internet of Things devices," *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 2, pp. 1422–1434, Mar./Apr. 2023.

[34] M. Albanese, *Dependency Graphs*. Berlin, Heidelberg: Springer, 2019, pp. 1–3. [Online]. Available: https://doi.org/10.1007/978-3-642-27739-9_1771-1

[35] C. Ju, G. Yang, Y.-W. Chen, and C. Pan, "Dynamic optimization of data packet-based communication for PLC visual monitoring," *Appl. Sci.*, vol. 9, no. 8, p. 1721, 2019.

[36] H. Yoo and I. Ahmed, "Control logic injection attacks on industrial control systems," in *Proc. 34th IFIP TC 11 Int. Conf. ICT Syst. Security Privacy Protect.*, 2019, pp. 33–48.

[37] J. Seong, R. Ranjan, J. Kye, S. Lee, and S. Lee, "Enhancing industrial communication with Ethernet/Internet protocol: A study and analysis of real-time cooperative robot communication and automation via transmission control protocol/internet protocol," *Sensors*, vol. 23, no. 20, p. 8580, 2023.

[38] S. Banik, T. Banik, S. M. Hossain, and S. K. Saha, "Implementing man-in-the-middle attack to investigate network vulnerabilities in smart grid test-bed," in *Proc. IEEE World AI IoT Congr. (AIIoT)*, 2023, pp. 0345–0351.

[39] F. Tacliad, T. D. Nguyen, and M. Gondree, "DoS exploitation of Allen-Bradley's legacy protocol through fuzz testing," in *Proc. 3rd Annu. Ind. Control Syst. Security Workshop*, 2017, pp. 24–31.

[40] E. N. Ylmaz, B. Ciylan, S. Gönen, E. Sindiren, and G. Karacayılmaz, "Cyber security in industrial control systems: Analysis of DoS attacks against PLCs and the insider effect," in *Proc. 6th Int. Istanbul Smart Grids Cities Congr. Fair (ICSG)*, 2018, pp. 81–85.

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

[42] C. Aguayo Gonzalez and A. Hinton, "Detecting malicious software execution in programmable logic controllers using power fingerprinting," in *Proc. 8th IFIP WG 11.10 Int. Conf. Crit. Infrastruct. Protect.*, 2014, pp. 15–27.

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

[44] H. Yang, L. Cheng, and M. C. Chuah, "Detecting payload attacks on programmable logic controllers (PLCS)," in *Proc. IEEE Conf. Commun. Netw. Security (CNS)*, 2018, pp. 1–9.

[45] H. Yoo, S. Kalle, J. Smith, and I. Ahmed, "Overshadow PLC to detect remote control-logic injection attacks," in *Proc. 16th Int. Conf. Detection Intrusions Malware, Vulnerability Assessment*, 2019, pp. 109–132.

[46] L. Garcia, F. Brasser, M. H. Cintuglu, A.-R. Sadeghi, O. A. Mohammed, and S. A. Zonouz, "Hey, my malware knows physics! Attacking PLCs with physical model aware rootkit," in *Proc. NDSS*, 2017, pp. 1–15.

[47] A. Abbasi, M. Hashemi, E. Zambon, and S. Etalle, "Stealth low-level manipulation of programmable logic controllers I/O by pin control exploitation," in *Proc. 11th Int. Conf. Crit. Inf. Infrastruct. Security*, 2017, pp. 1–12.

[48] W. Alsabbagh and P. Langendörfer, "A flashback on control logic injection attacks against programmable logic controllers," *Automation*, vol. 3, no. 4, pp. 596–621, 2022.
[49] A. C.-C. Yao, "How to generate and exchange secrets," in *Proc. 27th Annu. Symp. Found. Comput. Sci. (SFCS)*, 1986, pp. 162–167.
[50] Z. Sun, B. Feng, L. Lu, and S. Jha, "OAT: Attesting operation integrity of embedded devices," in *Proc. IEEE Symp. Security Privacy (SP)*, 2020, pp. 1433–1449.
[51] "IIoT sensors and valve control," Westlockcontrols, 2024. Accessed: Nov. 2, 2024. [Online]. Available: https://www.westlockcontrols.com/iiot-sensors-industrial-automation/
[52] *Industrial Communication Networks—Fieldbus Specifications*, IEC Standard 61158, 2019, [Online]. Available: https://webstore.iec.ch/publication/6028
[53] *OPC Unified Architecture*, IEC Standard 62541, 2020. [Online]. Available: https://webstore.iec.ch/publication/6023
[54] "ATmega2560 microcontroller board Arduino mega 2560 Rev3," Arduino, 2024. [Online]. Available: https://store-usa.arduino.cc/products/arduino-mega-2560-rev3
[55] "Arduino explorer application." 2024. [Online]. Available: https://www.mathworks.com/help/matlab/supportpkg/using-arduino-explorer.html
[56] "Simulink PLC coder." Mathworks, 2025. [Online]. Available: https://www.mathworks.com/products/simulink-plc-coder.html
[57] "OpenPLC project an open source PLC software." 2024. [Online]. Available: https://autonomylogic.com/
[58] "Simulink support package for Arduino hardware." 2024. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/40312-simulink-support-package-for-arduino-hardware
[59] B. Alotaibi, "A survey on Industrial Internet of Things security: Requirements, attacks, AI-based solutions, and edge computing opportunities," *Sensors*, vol. 23, no. 17, p. 7470, 2023.
[60] V. Varadharajan, U. Tupakula, and K. K. Karmakar, "Techniques for enhancing security in industrial control systems," *ACM Trans. Cyber-Phys. Syst.*, vol. 8, no. 1, pp. 1–36, 2024.
[61] A. P. Estrada-Vargas, E. López-Mellado, and J.-J. Lesage, "An identification method for PLC-based automated discrete event systems," in *Proc. 49th IEEE Conf. Decis. Control (CDC)*, 2010, pp. 6740–6746.
[62] J. C. Q. Quezada, E. F. García, J. M. Marín, J. B. López, and V. Q. Aguilar, "Ladder diagram Petri nets: Discrete event systems," in *Petri Nets in Science and Engineering*. London, U.K.: IntechOpen, 2018.
[63] J.-R. Jiang and Y.-T. Chen, "Industrial control system anomaly detection and classification based on network traffic," *IEEE Access*, vol. 10, pp. 41874–41888, 2022.
[64] J. Cho and S. Gong, "Dynamic data abstraction-based anomaly detection for industrial control systems," *Electronics*, vol. 13, no. 1, p. 158, 2023.
[65] C. Chang, W. Xu, H. Yan, L. Li, Y. S. Chu, and D. Yu, "Accelerating differential phase contrast imaging for NSLS-II data analysis," in *Proc. 10th Int. Conf. Expo Emerg. Technol. Smarter World (CEWIT)*, 2013, pp. 1–4.

Syed Owais Athar (Student Member, IEEE) received the B.S. degree in electronic engineering and the M.S. degree in electrical engineering from BUITEMS, Quetta, Pakistan, in 2012 and 2018, respectively.

He is a Fulbright Ph.D. Scholar with the University of Nebraska-Lincoln, Lincoln, NE, USA. He worked in industry and later joined BUITMES, as a Lecturer with the Department of Electronic Engineering. His research interests include industrial IoT, industrial cyber–physical systems, hardware security, machine learning, quantum computing, industrial electronics, power electronics, energy harvesting, renewable energy, and STEM education.

Muhammad Naveed Aman (Senior Member, IEEE) received the B.Sc. degree in computer systems engineering from KPK UET, Peshawar, Pakistan, the M.Sc. degree in computer engineering from the Center for Advanced Studies in Engineering, Islamabad, Pakistan, and the M.Engg. degree in industrial and management engineering and the Ph.D. degree in electrical engineering from Rensselaer Polytechnic Institute, Troy, NY, USA, in 2006, 2008, 2012, and 2012, respectively.

He is an Assistant Professor with the University of Nebraska-Lincoln, Lincoln, NE, USA. His research interests include IoT and network security, hardware systems security and privacy, wireless and mobile networks, and stochastic modeling.

Biplab Sikdar (Fellow, IEEE) received the B.Tech. degree in electronics and communication engineering from North Eastern Hill University, Shillong, India, in 1996, the M.Tech. degree in electrical engineering from Indian Institute of Technology Kanpur, Kanpur, India, in 1998, and the Ph.D. degree in electrical engineering from Rensselaer Polytechnic Institute, Troy, NY, USA, in 2001.

He is a Professor with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore, where he also serves as the Head of the Department of Electrical and Computer Engineering and the Director of the Cisco-NUS Corporate Research Laboratory. He was an Assistant Professor from 2001 to 2007 and an Associate Professor from 2007 to 2013 with the Department of Electrical, Computer, and Systems Engineering, Rensselaer Polytechnic Institute. His research interests include IoT and cyber–physical system security, network security, and network performance evaluation.

Prof. Sikdar is a recipient of the NSF CAREER award, the Tan Chin Tuan Fellowship from NTU Singapore, the Japan Society for Promotion of Science Fellowship, and the Leiv Eiriksson Fellowship from the Research Council of Norway. He has served as an Associate Editor for the IEEE TRANSACTIONS ON COMMUNICATIONS, IEEE TRANSACTIONS ON MOBILE COMPUTING, and IEEE INTERNET OF THINGS JOURNAL, and is an IEEE COMSOC and a VTS Distinguished Lecturer and an ACM Distinguished Speaker. He is a member of Eta Kappa Nu and Tau Beta Pi.