

Latency-Aware Security Integration for Operational Technology Networks via Edge Computing: A Comparative Evaluation of TLS-Secured MQTT and Modbus

1st Javier Otero Mosquera
Universidade de Vigo
Vigo, Spain
0000-0002-8552-5871

2nd Pablo Fondo-Ferreiro
atlanTTic
Universidade de Vigo
Vigo, Spain
0000-0002-7732-8288

3rd Cristina López-Bravo
atlanTTic
Universidade de Vigo
Vigo, Spain
0000-0002-9429-8076

4th Pedro Tubío Figueira
Universidade de Vigo
Vigo, Spain
0009-0001-7586-8416

5th Miguel Ramón Díaz-Cacho Medina
Enxeñaría de Sistemas e Automática
Universidade de Vigo
Vigo, Spain
0000-0002-2784-1374

*Corresponding author: Pablo Fondo-Ferreiro (email: pfondo@gti.uvigo.es)

Abstract—The convergence of Information Technology (IT) and Operational Technology (OT) in Industry 4.0 exposes legacy industrial control systems to cybersecurity threats, requiring protective measures that strictly respect real-time and availability constraints. While edge computing offers a promising approach to relocate security computation away from resource-constrained field devices without incurring cloud latency penalties, quantitative evidence evaluating the trade-off between protocol-level security and latency in such architectures remains limited. To address this challenge, an edge-computing architecture is proposed to offload security-related computation and network monitoring away from the operational layer. In this paper, we present a physical-emulated OT testbed that allows for a quantitative comparison of latency overhead and resource utilization across secured industrial protocols, while evaluating the effectiveness of edge-hosted intrusion detection pipelines. This proposed framework provides actionable guidance for deploying efficient security retrofits in brownfield industrial environments.

Index Terms—Industry 4.0, OT Security, Edge Computing, MQTT, Modbus, TLS, Intrusion Detection Systems, Network Latency

I. INTRODUCTION

Cybersecurity is currently ranked among the top global risks by the World Economic Forum [1], and industrial control systems (ICS) are a disproportionately exposed target: many field devices were designed decades ago under an implicit assumption of network isolation, without

authentication or cryptographic protection [2]. As IT and OT networks converge under the Industry 4.0 paradigm, this legacy creates a structural tension. IT security practice prioritizes confidentiality, whereas OT prioritizes availability and deterministic, low-latency operation [3]; any retrofitted security control that measurably increases latency or jitter is, in practice, unusable in a production environment.

Among field devices, PLCs are particularly exposed, as they directly control physical processes and remain a primary target for adversaries (63% of recorded intrusion attempts targeted Modbus TCP/IP ports used by PLCs [4]). This exposure is particularly acute for the communication protocols that carry sensor and control data in OT networks. Modbus, still one of the most widely deployed industrial protocols, transmits in plaintext and offers no native authentication, exposing it to man-in-the-middle and command-injection attacks. MQTT, the dominant publish/subscribe protocol in the wider Industrial Internet of Things (IIoT), suffers from comparable exposure when deployed without transport-layer protection: broker-level scans have repeatedly found hundreds of thousands of unauthenticated, unencrypted MQTT endpoints reachable from the public Internet [5]. Transport Layer Security (TLS) is the natural remedy for both protocols, but encryption, key exchange and certificate validation are not free: they consume CPU cycles and add round-

trip latency on devices that were never dimensioned for cryptographic workloads.

Edge computing offers a way to reconcile these constraints. By relocating computationally expensive tasks, protocol termination, intrusion detection, monitoring and analytics, to a node physically close to the plant floor, edge architectures avoid the latency and bandwidth penalties of cloud offloading while still freeing field devices from carrying the full security burden [6, 7, 8, 9]. Prior work has examined edge-assisted ICS control loops [8], gateway-based protocol protection [10], and Modbus/TLS overhead in isolation [11]. In addition, Snort-based detection has been shown effective for Modbus-specific attack signatures [12]. Despite the growing body of research in this area, the existing literature lacks a direct, controlled, within-architecture comparison of how two of the most widely adopted OT/IIoT protocols, Modbus and MQTT, behave in terms of both latency and resource consumption once secured under an equivalent standard, and once their associated monitoring and detection workload is offloaded to a common edge platform.

This paper addresses that shortcoming by providing an empirical, side-by-side assessment of both protocols under comparable security and processing conditions. We built a physical-emulated OT testbed, consisting of a GNSS sensor feeding an industrial PLC, an embedded Raspberry Pi gateway, and an AdvantEDGE-based edge-computing node, and implemented two parallel, functionally equivalent communication architectures, one over MQTT and one over Modbus TCP, both secured end-to-end with TLS certificates, with network intrusion detection (Snort) and latency/monitoring computation fully hosted at the edge. Our contribution is threefold: (i) a reusable, edge-integrated architecture in which OT security processing is decoupled from constrained field devices; (ii) a quantitative, statistically characterized comparison of the latency and resource cost that TLS-secured MQTT and Modbus impose under otherwise identical conditions; and (iii) empirical validation that an edge-hosted Snort instance can detect representative OT attack traffic (port scanning, malformed packets, reverse shells) with negligible propagation delay.

The remainder of the paper is organized as follows: Section II situates the work relative to prior ICS security and edge-computing literature; Section III describes the testbed, protocol implementations and measurement methodology; Section IV reports latency and resource-consumption results; Section V interprets these results and discusses their practical implications; and Section VI present the conclusions.

II. RELATED WORK

The security gap between IT and OT systems has been extensively documented. [3] formalized the inversion

of CIA-triad priorities between the two domains, and [2] cataloged the specific control-system vulnerabilities that arise when legacy OT equipment is connected to IT-adjacent networks. At the protocol level, Modbus TCP's absence of authentication and encryption is well established, and TLS wrapping has been proposed as a remedy, though [11] showed that the TLS handshake introduces a non-negligible connection-establishment delay that must be characterized before deployment in latency-sensitive industrial settings. MQTT faces an analogous exposure problem at scale: large-scale Shodan-based scans found that a substantial fraction of publicly reachable MQTT brokers implement neither authentication nor encryption [5], underscoring that the protocol's lightweight design is frequently deployed without its optional security extensions enabled.

In [13] a secure gateway component with built-in lightweight cryptography is proposed to safeguard OT resources. Although this approach secures part of the communication link, it fails to guarantee end-to-end authentication and confidentiality.

Edge computing has emerged as an architectural response to the latency and bandwidth limitations of cloud-centralized industrial analytics [6]. [8] demonstrated that distributing control computation between local PLCs and edge servers over wireless links improves adaptability to varying network conditions, while [7] showed that dynamic edge/cloud service integration reduces the data-movement and bandwidth requirements of industrial cyber-physical systems. Closer to the security domain, [10] argued that gateway-based protection architectures can secure heterogeneous ICS protocols, including Modbus, while introducing only marginal communication delay, and [14] proposed interlocking IT/OT security mechanisms specifically for edge-cloud-enabled manufacturing. Network-based intrusion detection with Snort has been validated for Modbus-specific attack patterns by [12], who showed that custom detection rules can reliably flag ICS-targeted intrusions.

These contributions establish, individually, that (a) TLS is a viable but non-free security control for OT protocols, and (b) edge computing is a viable venue for offloading ICS security processing. However, none of the reviewed studies directly compares two secured OT protocols against each other, under a shared edge-based security-processing architecture, using a controlled testbed. The present work is, to our knowledge, the first to provide this direct comparison, integrating an AdvantEDGE-based emulated edge platform, TLS-secured MQTT and Modbus, and an edge-hosted Snort deployment within a single reproducible experiment.

III. SYSTEM ARCHITECTURE AND IMPLEMENTATION

This section details the proposed edge-assisted security architecture and its implementation. The overall physical-emulated testbed setup and telemetry definitions are described, followed by the specific TLS-secured MQTT and Modbus TCP protocol paths, the edge-hosted intrusion detection pipeline, and the measurement methodology.

A. Testbed architecture

The testbed described in Fig. 1 emulates an end-to-end industrial data path: one GNSS (Grove-GPS, Air530) sensor streams continuous NMEA position frames over a serial interface to an industrial PLC (Industrial Shields Ardbox Relay WiFi/BT, based on an Arduino Leonardo with an ESP32 WiFi module). The PLC forwards the GNSS telemetry alongside a self-reported device-health packet generated every 500 ms to an embedded gateway (Raspberry Pi 4 Model B, 4 GB RAM, Ubuntu 22.04) over an 802.11 WiFi link. This health packet includes:

- **System Metrics:** Timestamp, temperature, and CPU frequency.
- **Memory & Storage:** Memory usage, minimal memory usage, flash size, and flash usage.
- **Connectivity:** WiFi RSSI.

The gateway relays this traffic to an edge-computing node over Ethernet and additionally emits its own security-posture telemetry every 500 ms. As summarized in Table I, these parameters comprise continuous resource metrics along with binary and ternary indicators that track system state integrity against a boot-time baseline informed by CIS Benchmark recommendations. Every five seconds, the gateway additionally captures a pcap traffic dump on both its WiFi (PLC-facing) and Ethernet (edge-facing) interfaces and forwards it to the edge node for intrusion analysis.

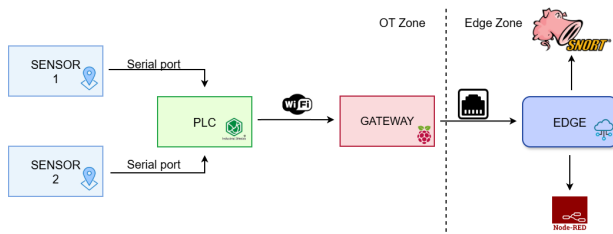


Figure 1: Basic architecture.

The edge-computing tier was implemented using AdvantEDGE, an open-source mobile-edge emulation platform that allows configurable per-link latency, jitter and packet loss (default values were used throughout). Two containerized services run at the edge: (i) Node-RED, which visualizes PLC and gateway telemetry, ingests intrusion alerts, and computes end-to-end and hop-by-hop latency from NTP-synchronized timestamps; and

Table I: Gateway security telemetry variables

Security Parameter	Possible Range
CPU Utilization	[0.0 %, 100.0 %]
RAM Utilization	[0.0 %, 100.0 %]
NTP Config Integrity	(0: Intact, 1: Modified)
SSH Config Integrity	(0: Intact, 1: Modified)
Open Ports	(−1: Decreased, 0: Same, 1: Increased)
Active / Used Ports	(−1: Decreased, 0: Same, 1: Increased)
System Users	(−1: Decreased, 0: Same, 1: Increased)
Environment Variables	(−1: Decreased, 0: Same, 1: Increased)

(ii) Snort, which analyses the periodic traffic captures against a custom rule set and reports detections to Node-RED via an HTTP POST callback. All clocks in the architecture are synchronized via NTP to a common time reference, which is the basis for latency computation as the difference between packet arrival time and the timestamp embedded at the point of origin.

B. Protocol implementations

Two functionally equivalent, TLS-secured communication paths were implemented between the gateway and the edge node, replacing an initial unsecured baseline used only for functional verification.

MQTT architecture (Fig. 2(a)). A Mosquitto broker (custom Docker image, based on eclipse-mosquitto) was deployed at the edge, exposing port 8883 and requiring username/password authentication in addition to TLS encryption. The gateway publishes PLC data (/plc/system) and its own security telemetry (/gateway/data) via the paho-mqtt client library; Node-RED subscribes to both topics using its native MQTT client, configured with the CA certificate for server validation.

Modbus architecture (Fig. 2(b)). The gateway hosts a TLS-wrapped Modbus TCP server (pymodbus), exposing PLC and gateway telemetry through holding registers. Because Node-RED has no native Modbus client with TLS support, a custom Python client, co-located in the same Docker container as Node-RED at the edge, polls the gateway and forwards the decoded register values internally via UDP sockets to Node-RED’s visualization and latency-computation logic.

In both cases, TLS certificates and keys (CA, server, and where applicable client) were generated with OpenSSL following a standard self-signed CA workflow, and the architecture underlying both protocol paths

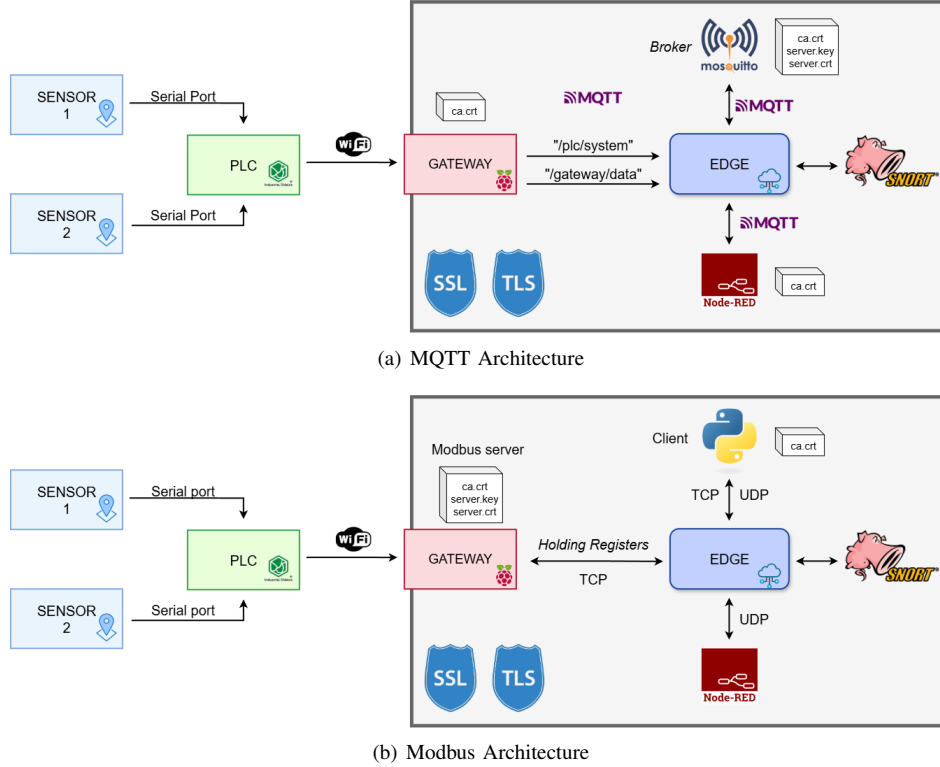


Figure 2: Testbed Architecture for MQTT and Modbus Protocols.

gateway, edge platform and Snort deployment, was held constant to isolate the effect of protocol choice.

C. Attack generation and detection

To validate the intrusion-detection pipeline, a Python *scapy*-based script generated a representative set of OT-relevant attacks against the gateway: malformed UDP and TCP packets (length and checksum errors), malformed ICMP packets, a reverse-shell connection attempt, and port-scanning traffic (SYN, ACK and FIN scans). Corresponding Snort rules, written in Lua, were configured to flag each pattern and forward alerts to Node-RED over HTTP, timestamped both at detection (Snort) and reception (Node-RED).

D. Latency and resource measurement

The system operated continuously for approximately five minutes per protocol configuration, collecting 1012 PLC-to-edge and 319 gateway-to-edge samples for MQTT; and 1225 PLC-to-edge and 313 gateway-to-edge samples for Modbus. For each configuration we report the sample mean, standard deviation, and 95% confidence interval of end-to-end latency (PLC to edge) and gateway-to-edge latency. CPU and RAM utilization at both the gateway and the edge node were sampled concurrently using standard system-monitoring tools (*htop*) for each protocol configuration.

IV. RESULTS

This section presents the empirical evaluation of the proposed edge-assisted security architecture. End-to-end and hop-by-hop latency metrics were evaluated, resource consumption was benchmarked across the gateway and edge computing nodes, and the functional performance of the edge-hosted intrusion detection system was assessed under active attack conditions.

A. Latency

Table II summarizes the latency metrics for both protocols across the two designated measurement points. Over continuous operation, MQTT achieved a mean end-to-end (PLC-to-edge) latency of 37.27 ms ($SD = 9.49$ ms, 95% CI [36.68, 37.85]), whereas Modbus reached 162.68 ms ($SD = 74.21$ ms, 95% CI [158.68, 166.99]), representing a more than fourfold increase.

The gateway-to-edge segment, which isolates the overhead attributable to TLS encryption/processing and the periodic traffic-capture routine at the gateway, reflected an identical pattern: 31.33 ms for MQTT versus 106.62 ms for Modbus.

Beyond the higher average response times, the variance (SD) observed in Modbus latency is notably severe (74.21 ms vs. 9.49 ms in MQTT). This pronounced jitter indicates that securing Modbus with TLS introduces

significant unpredictability into communication delays, whereas MQTT maintains a highly stable and deterministic delivery profile.

Table II: Latency results (ms) for TLS-secured MQTT and Modbus.

Configuration	Mean	SD	95% CI	
			Lower	Upper
MQTT – PLC-to-edge	37.27	9.49	36.68	37.85
Modbus – PLC-to-edge	162.68	74.21	158.68	166.99
MQTT – Gateway-to-edge	31.33	8.40	30.41	32.25
Modbus – Gateway-to-edge	106.62	62.42	99.70	113.54

B. Resource consumption

At the edge node, CPU and RAM utilization remained minimal in both configurations (below approximately 10% mean per-core utilization for Node-RED, Mosquitto and Snort combined), indicating that the intrusion-detection and monitoring workload, the component of the architecture explicitly designed to be edge-resident, does not create a resource bottleneck regardless of the underlying protocol.

At the gateway, however, utilization differed markedly between configurations: MQTT operation kept CPU usage in the 24–42% range per core with approximately 1.13/3.70 GB RAM in use, whereas Modbus operation raised CPU usage to 58–74% per core with a comparable RAM footprint. Given that the gateway is the most resource-constrained device in the architecture (an embedded single-board computer, in contrast with the more capable emulated edge node), this asymmetry is operationally significant: Modbus imposes roughly twice the CPU load of MQTT on precisely the tier of the architecture least able to absorb it.

C. Intrusion detection

The edge-hosted Snort NIDS successfully detected and flagged all six injected attack vectors (malformed UDP, TCP, ICMP, reverse-shell attempts, and TCP SYN/ACK/FIN port scans). The propagation delay between Snort threat detection and alert rendering within Node-RED was minimal, confirming that edge-level alert routing introduces negligible operational delay.

V. DISCUSSION

The central empirical finding of this study is that, once security processing (TLS encryption, intrusion detection, monitoring) is held constant across two OT protocols within the same edge-integrated architecture, protocol choice rather than not the presence of security itself is the dominant determinant of end-to-end latency. Both configurations remain, in absolute terms, within latency ranges generally considered acceptable for supervisory monitoring and non-safety-critical industrial telemetry.

However, the relative gap between them (a mean difference exceeding 125 ms end-to-end) is large enough to be decisive in any deployment where multiple sequential hops, additional processing stages, or tighter closed-loop control requirements would compound this overhead.

Two architectural factors explain the Modbus penalty. First, the register-based data model of Modbus requires multiple discrete read operations to reconstruct a single logical value. Notably, three separate register reads to reconstruct the timestamp alone, which multiplies the number of TLS-protected round trips required per logical message, in contrast with MQTT’s single-message publish/subscribe exchange. Second, the absence of a native, TLS-capable Modbus client in Node-RED required a custom polling client, introducing a client-initiated request/response pattern inherently less efficient than MQTT’s broker-mediated, subscriber-driven delivery, in which the edge-side subscriber is continuously listening rather than polling. This latter point also has an important secondary implication: because Modbus values are updated at the gateway without any push notification to the edge-side client, there is a structural risk that reads may occasionally retrieve stale register values between gateway-side updates. Consequently, a synchronization hazard absent from the MQTT architecture, where message delivery is inherently event-driven.

From a resource-provisioning standpoint, the disparity in gateway-side CPU utilization (roughly double for Modbus) is arguably of greater practical consequence than the latency values themselves, precisely because the gateway is the tier of the architecture most likely to be resource-constrained in a real brownfield deployment (an embedded device retrofitted onto legacy equipment, rather than a purpose-provisioned edge server). A protocol choice that halves gateway CPU load directly extends the range of legacy hardware onto which this security architecture can be retrofitted without a corresponding hardware upgrade, a practically relevant consideration given that under-investment in OT cybersecurity is frequently driven by budget and hardware-replacement constraints rather than by an absence of awareness of the risk.

Taken together with the successful, low-latency detection of all injected attack types, these results support a specific, actionable recommendation for practitioners retrofitting security into brownfield OT environments under similar architectural constraints: where protocol choice is not already fixed by legacy equipment, MQTT-over-TLS combined with edge-hosted intrusion detection offers a more favorable latency/resource trade-off than Modbus-over-TLS for equivalent security guarantees. Where Modbus is mandated by existing infrastructure, its latency and CPU cost should be explicitly budgeted for in the design of any control loop or alerting mechanism

that depends on timely data delivery.

VI. CONCLUSIONS

This paper presented and empirically evaluated an edge-computing-integrated architecture for securing OT communications without relocating security processing onto resource-constrained field devices. By implementing MQTT and Modbus TCP, each secured end-to-end with TLS, within an otherwise identical testbed, and offloading intrusion detection and monitoring entirely to an edge node, we isolated and quantified the latency and resource-consumption cost attributable to protocol choice under a shared security posture. MQTT-over-TLS exhibited a mean end-to-end latency of 37.27 ms compared to 162.68 ms for Modbus-over-TLS, together with substantially lower gateway-side CPU utilization, while an edge-hosted Snort deployment correctly and promptly detected all injected attack traffic in both configurations.

As future work we plan to integrate the original testbed into a real 5G network deployment.

ACKNOWLEDGMENT

This work was supported in part by Xunta de Galicia (Spain) under grant ED481D-2025/007; in part by the Ministerio de Ciencia, Innovación y Universidades, Spain, under grants PID2024-156563OA-I00 and PID2023-148214OB-C22; and in part by the R&D&I project “Creation of a network of laboratories and demonstration centers for research and innovation in cybersecurity solutions,” funded by the European Union - NextGenerationEU, the Recovery, Transformation and Resilience Plan, and INCIBE.

REFERENCES

- [1] World Economic Forum, “Informe sobre riesgos globales 2024: Los riesgos aumentan, pero también nuestra capacidad de respuesta,” <https://es.weforum.org/agenda/2024/01/informe-sobre-riesgos-globales-2024-los-riesgos-aumentan-pero-tambien-nuestra-capacidad-de-respuesta/>, 2024, accedido: Julio 2024.
- [2] S. Sriram, G. Rajeshkumar, S. Sadesh, E. Saranya, K. Saranya, and K. Venu, “Cyber security control systems for operational technology,” *2023 Second International Conference on Electronics and Renewable Systems (ICEARS)*, pp. 1–8, 2023.
- [3] W. A. Conklin, “It vs. ot security: A time to consider a change in CIA to include resilience,” in *2016 49th Hawaii International Conference on System Sciences (HICSS)*, 2016, pp. 2642–2647.
- [4] K. Ramdass and N. K. Balaraman, “Securing industrial control systems: PLC, SCADA, and HMI protection,” in *2025 IEEE 13th Conference on Systems, Process & Control (ICSPC)*. IEEE, 2025, pp. 252–257.
- [5] D. Kant, A. Johannsen, and R. Creutzburg, “Analysis of IoT security risks based on the exposure of the MQTT protocol,” *Electronic Imaging*, vol. 2021, no. 3, pp. 96–1–96–8, 2021.
- [6] S. Douch, M. R. Abid, K. Zine-Dine, D. Bouzidi, and D. Benhaddou, “Edge computing technology enablers: A systematic literature study,” *IEEE Access*, vol. 10, pp. 69 264–69 302, 2022.
- [7] D. Hästbacka, J. Halme, L. Barna, H. Hoikka, H. Pettinen, M. Larrañaga, M. Björkbohm, H. Mesiä, A. Jaatinen, and M. Elo, “Dynamic edge and cloud service integration for industrial IoT and production monitoring applications of industrial cyber-physical systems,” *IEEE Transactions on Industrial Informatics*, vol. 18, no. 1, pp. 498–508, 2022.
- [8] Y. Ma, C. Lu, B. Sinopoli, and S. Zeng, “Exploring edge computing for multitier industrial control,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 11, pp. 3506–3518, 2020.
- [9] M. Alqahtani, A. Amin, K. Kim, and S. Lee, “Enhancing scada security in critical infrastructure: A multi-layered architecture using IoT-based monitoring and AI-driven anomaly detection,” *International Journal of Advanced Computer Science and Applications*, vol. 17, no. 1, 2026. [Online]. Available: <https://doi.org/10.14569/IJACSA.2026.0170187>
- [10] M. Slunjski and D. Sumina, “Investigation of Internet-of-Things gateway-based solution for protection of industrial control systems,” in *2023 3rd International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)*, 2023, pp. 1–6.
- [11] M. K. Ferst, H. F. M. de Figueiredo, and G. W. Denardin, “Connection time in Modbus/TLS for secure communications on photovoltaic systems,” in *2019 IEEE 15th Brazilian Power Electronics Conference and 5th IEEE Southern Power Electronics Conference (COBEP/SPEC)*, 2019, pp. 1–6.
- [12] C. Kim and D. Robinson, “Modbus monitoring for networked control systems of cyber-defensive architecture,” in *2017 Annual IEEE International Systems Conference (SysCon)*, 2017, pp. 1–6.
- [13] D. Upadhyay *et al.*, “Securing industrial control systems: Developing a SCADA/IoT test bench and evaluating lightweight cipher performance on hardware simulator,” *International Journal of Critical Infrastructure Protection*, vol. 47, p. 100705, 2024. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1874548224000465>
- [14] T. Kampa *et al.*, “Interlocking IT/OT security for edge cloud-enabled manufacturing,” *Ad Hoc Networks*, vol. 154, p. 103384, 2024.