IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 12 April 2026, accepted 3 May 2026. Date of publication 00 xxxx 0000, date of current version 00 xxxx 0000.

Digital Object Identifier 10.1109/ACCESS.2026.3691926

Secure Data Bridging in Industry 4.0: An OPC UA Aggregation Approach for Including Insecure Legacy Systems

DALIBOR SAIN[1], THOMAS ROSENSTATTER[1], OLAF SAßNICK[1,2], CHRISTIAN SCHÄFER[3], AND STEFAN HUBER[1]

[1]Josef Ressel Centre for Intelligent and Secure Industrial Automation, Salzburg University of Applied Sciences, 5412 Salzburg, Austria
[2]Paris Lodron University of Salzburg, 5020 Salzburg, Austria
[3]B&R Industrial Automation GmbH, 5142 Eggelsberg, Austria

Corresponding author: Thomas Rosenstatter (thomas.rosenstatter@fh-salzburg.ac.at)

This work was supported by the Austrian Federal Ministry of Economy, Energy and Tourism, the National Foundation for Research, Technology and Development and the Christian Doppler Research Association.

ABSTRACT The increased connectivity of industrial networks has led to a surge in cyberattacks, emphasising the need for cybersecurity measures tailored to the specific requirements of industrial systems. Modern Industry 4.0 technologies, such as OPC UA, offer enhanced resilience against these threats. However, widespread adoption remains limited due to long installation times, proprietary technologies, restricted flexibility, and formal process requirements (e.g. safety certifications). Consequently, many systems do not yet implement these technologies, or only partially. This leads to the challenge of dealing with so-called brownfield systems, which are often placed in isolated security zones to mitigate risks. However, the need for data exchange between secure and insecure zones persists. This paper reviews existing solutions to address this challenge by analysing their approaches, advantages, and limitations. Building on these insights, we identify three key concepts, evaluate their suitability and compatibility, and ultimately introduce the SigmaServer, a novel TCP-level aggregation approach. The developed proof-of-principle implementation is evaluated in an operational technology (OT) testbed, demonstrating its applicability and effectiveness in bridging secure and insecure zones.

INDEX TERMS Industry 4.0, OPC UA, security, aggregation server, brownfield, retrofit.

I. INTRODUCTION

A. MOTIVATION

By applying the design principles of Industry 4.0, modern industrial plants are evolving into highly interconnected Operational Technology (OT) systems, often also linked to the worldwide Internet. While providing benefits such as higher efficiency and greater flexibility, the interconnectedness also increases the overall attack surface, resulting in higher cybersecurity risks [1], [2].

To address these risks, initiatives like the IEC 62443 [3] put focus on cybersecurity for OT, proposing the use of zones and conduits to strengthen the security of industrial networks. As such, a network is separated into multiple zones with different security requirements, following a defence-in-depth concept. Zones are connected by conduits, which are typically firewalls or one-way gateways (data diodes), restricting unauthorised traffic.

A remaining challenge, however, is the long operational lifespan of OT equipment. While IT components like desktop and server systems are typically used for 3-5 years, a production machine represents a far larger investment and is commonly used for 10-15 years and longer [4]. The hardware and software embedded in these machines cannot easily be kept up to date and therefore at some point in their lifespan no longer support up-to-date security standards (such as secure OPC UA communication). They become legacy devices, providing only reduced or no security measures. As such

The associate editor coordinating the review of this manuscript and approving it for publication was Fabrizio Messina.

they pose a risk to the entire industrial network, significantly weakening the overall security. In addition, some protocols may still be popular in the industry, yet do not provide any or sufficient security measures. For instance, the industrial protocol Modbus lacks built-in security features, which led to incidents like FrostyGoop in 2024 [5], where attackers exploited unauthenticated Modbus commands.

In this paper, we use the term *legacy device* to refer to devices that are *not able to fulfil the security requirements* of modern industrial networks – including also insecure protocols such as Modbus.

To mitigate the risk of legacy devices, they can be placed in fully isolated zones. The isolation prevents attacks but at the same time removes the benefits of connectivity, such as precise synchronisation and process optimisation. Therefore, to a certain extent, legacy devices still must be able to communicate beyond their isolated zone, for instance, with Supervisory Control and Data Acquisition (SCADA) systems.

This can be realised with middleware solutions, bridging the gap between isolated zones containing legacy devices and secure zones with systems supporting modern security features. Such solutions must not only be able to translate between different protocols, but also must represent legacy devices beyond their isolated zone with up-to-date security features. Especially the latter requirement is currently not well addressed by existing solutions, as the primary focus is a working, fast and reliable translation between protocols. Therefore, in the following work, our focus is on how to securely integrate legacy brownfield systems in modern secure industrial networks.

B. CONTRIBUTION

Our contribution towards securing industrial legacy systems via aggregation in modern production environments is threefold:

- We provide a structured review of published middleware aggregation solutions and analyse them with regard to their security capabilities.
- As none of the published solutions are found to focus on security, we define a common threat model and introduce three general architectural concepts for embedding insecure legacy systems in a modern, secure industrial network aligned with the Purdue model [6].
- We implement the most practical of these concepts as a proof-of-principle, named SigmaServer, and evaluate its performance characteristics, such as latency and system load, demonstrating that retrofitting can be achieved with acceptable technical effort and low system overhead.

The proposed approach introduces a novel combination of security-aware design and TCP-level namespace separation that has not been addressed in prior work. Existing OPC UA aggregating servers merge the address spaces of multiple devices into a single shared namespace, a pattern we refer to as *namespace pollution*. Furthermore, none of the reviewed aggregation solutions treat security as a primary design concern. The SigmaServer addresses both by instantiating a dedicated OPC UA server per legacy device at the TCP layer. Each server listens on a distinct port, thereby preserving the original namespace structure and eliminating pollution entirely. Communication on the secure side is enforced through OPC UA security policies, ensuring authenticated and encrypted communication on the secure endpoint. The SigmaServer also introduces the capability of simultaneously managing heterogeneous legacy devices running different protocols within a single deployment, exposing each securely as an individual OPC UA endpoint.

The evaluation shows that our proposed SigmaServer achieves an end-to-end latency below 2.6 ms, while its internal processing delay remains in the low microsecond range with an average of 21.15 μs. SigmaServer requires significantly less RAM usage (6–19 MiB) compared with the Open Platform Communications (OPC) Foundation Console Aggregation Server[1] (105–115 MiB). Although CPU usage is slightly higher, it remains in the low single-digit range with 0.75–3.16%. For reproducibility and further research, our implementation is also made publicly available on GitHub.[2]

C. ORGANIZATION OF THE PAPER

The rest of the paper is organised as follows. Section II provides background information on industrial networks and zones, and an overview of OPC UA. Furthermore, this section introduces the reference architecture which is followed throughout this paper. Section III presents the methodology, which is based on two research questions. Section IV reviews the state of the art and analyses existing solutions with regard to their security capabilities. Section V introduces the threat model and three architectural concepts which are evaluated based on their suitability and compatibility with industrial networks. The design of the proposed solution, SigmaServer, is further detailed in Section VI. Section VII and VIII describe the setup of the evaluation experiments and the results respectively. Section IX provides a dedicated security analysis, discussing what SigmaServer achieves against the defined threat model. Finally, Section X concludes the paper and provides an outlook on future work.

II. BACKGROUND

A. INDUSTRIAL NETWORKS

Industrial networks form the technological backbone of modern automation environments. They enable the interaction between sensors, actuators, controllers, and supervisory systems.

At the core of these networks are Programmable Logic Controllers (PLCs), which are located on the field level and execute deterministic control tasks within strict timing

[1] https://github.com/OPCFoundation/UA-.NETStandard-Samples/tree/master/Workshop/Aggregation/ConsoleAggregationServer/
[2] https://github.com/JRC-ISIA/opc-ua-sigmaserver/

constraints. Such real-time operation ensures that industrial processes, which range from manufacturing lines to energy grids, function deterministically and reliably [7].

In contrast to conventional Information Technology (IT) infrastructure, PLCs often rely on specialised industrial communication technologies beyond TCP/IP over Ethernet. They often additionally use fieldbuses or industrial Ethernet-based technology (e.g., POWERLINK, PROFINET, EtherCAT) that better satisfy real-time and reliability requirements essential to process control.

One of these protocols is Modbus, which is open and widely adopted in the industrial domain [8]. This widespread adoption can be attributed to its inherent simplicity allowing interoperability among devices. Originally, Modbus was developed for serial communication through Modbus RTU and Modbus ASCII. The Modbus protocol has since evolved to include Ethernet-based communication via Modbus/TCP as part of IEC 61158 [9], thereby accommodating a broad spectrum of industrial applications and use cases. Given that the Modbus variant including security features based on TLS, known as Modbus/TCP Security, was not introduced until 2018, its adoption remains limited, which is understandable considering the extended operational lifetimes typical of industrial control systems. Moreover, Modbus/TCP Security is only defined by the Modbus Organisation – it is not part of IEC 61158.

Above the level of device control (PLCs, actuators and sensors), SCADA systems offer visualisation, data logging, and high-level management of processes distributed across multiple sites. SCADA systems integrate with PLCs, Remote Terminal Units (RTUs), and Human-Machine Interfaces (HMIs) to provide operators with a comprehensive view of system performance and state. Together, these components create a hierarchical architecture where local, time-critical control functions operate in an environment alongside centralised monitoring and decision-making [7].

B. INDUSTRY 4.0 AND OPC UA

The term *Industry 4.0*, the fourth industrial revolution, is characterised by the convergence of physical, industrial systems with traditional IT systems. The goal is to create production systems, which are capable of self-optimisation and autonomous decision-making [10].

For that purpose, flexibility, adaptability, transparency and interoperability across heterogeneous systems must be met [11]. Schleipen et al. [11] mention that such a compliant system must therefore be able to communicate via open, standardised IT networks, be self-describing and protective about its own information, and also be able to configure and optimise themselves. With the so-called RAMI 4.0 model [12], a reference architecture for Industry 4.0 systems was developed, which specifies the needed semantic technologies.

Furthermore, a strategy paper by the German Federal Government [13], identifies Open Platform Communications Unified Architecture (OPC UA) as a foundational technology for Industry 4.0. This is also strengthened in a report discussing the criteria for Industry 4.0 by ZVEI [14]. OPC UA supports security features such as access control, and provides platform independence through its information model and decoupling from the database model, which makes it a well-suited solution for industrial communication [11].

OPC UA, the successor of the OPC standard, has become a widely adopted standard for industrial communication. It is standardised in IEC 62541 and maintained by the OPC Foundation. The platform-independent, secure-by-design and service-oriented architecture, enables communication for industrial systems [15], either in a client-server fashion, or using a publish-subscriber model.

OPC UA defines not only a communication protocol but also introduces information models to structure variables, methods, and events [16]. The information model consists of nodes (objects, variables, methods, events, etc.) and references between them, organised within the address space of an OPC UA server. OPC UA servers at configured endpoints allow the information model's variables to be read or written and its methods to be invoked. Each OPC UA node can be addressed using its Uniform Resource Identifier (URI) and a unique identifier [15].

Concerning its security, the specification defines a set of security features, which help postulate a secure-by-design architecture. These features include secure session management, transport security, user authentication and access control.

OPC UA aggregating servers connect multiple OPC UA servers and provide a single, centralised, aggregated address space. Großmann et al. [17] propose a reference architecture for such a server, which contains components for providing the aggregated address space, fetching underlying data, discovery and security management for access control.

Figure 1 presents various deployment scenarios of an OPC UA aggregating server showing the aggregation server in different positions in the industrial network. It illustrates an industrial network architecture inspired by the Purdue model [6] and the recommendations in IEC 62443 [3]. This network is separated into three zones, which are connected through firewalls: (i) the shop floor represents the field level (see Section II-A) containing the PLCs, (ii) the operational level contains monitoring and high-level control of the process via a SCADA system and an HMI for user interaction, and (iii) the highest level, the enterprise zone, contains enterprise systems like financing and corporate management. In Figure 1, three possible deployment scenarios are highlighted, with the aggregating server located either on the shop floor or at one of the higher levels, depending on the specific use case. Their positioning depends on the use case at hand, i.e., the clients which should be served by the aggregating server. For instance, if external clients from the enterprise zone or from the internet should be served, the aggregating server should be placed in the operational level.

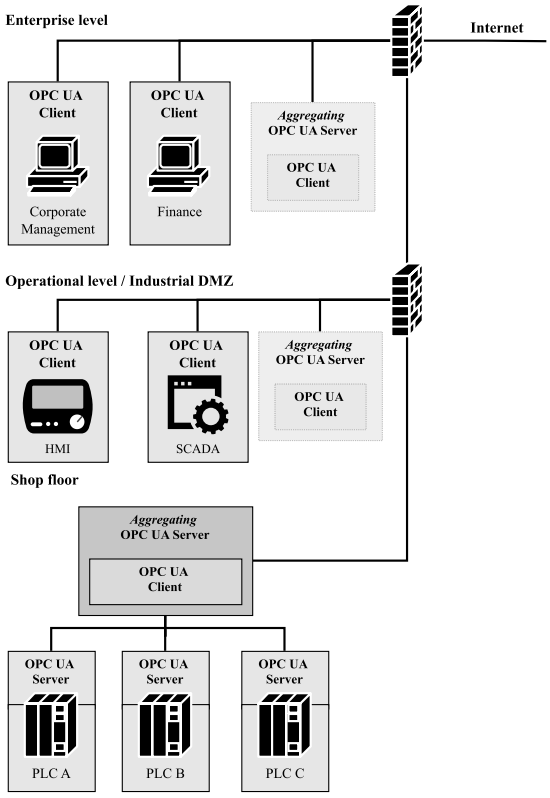


FIGURE 1. Deployment scenarios of an OPC UA aggregating server providing a single, aggregated address space [17], [18].

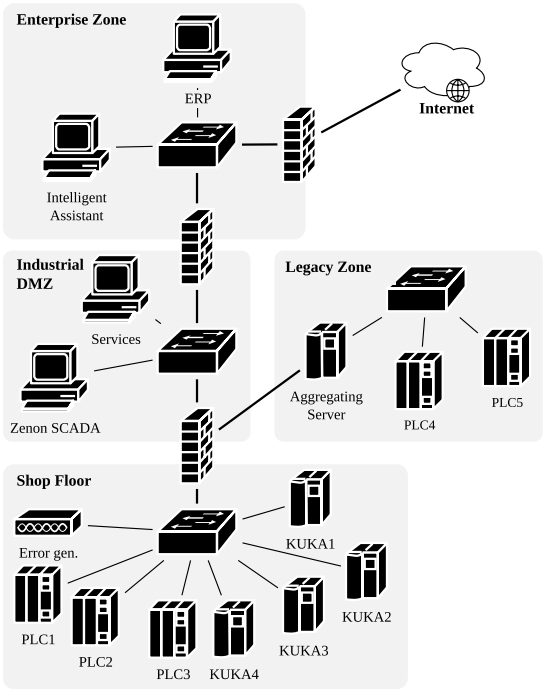


FIGURE 2. Reference architecture of an industrial system. The network setup of the JRC ISIA testbed shows the segmentation into four zones: enterprise zone, industrial Demilitarised Zone (DMZ), legacy zone, and shop floor. This setup is further used in the evaluation of the proposed SigmaServer.

The primary advantage of deploying aggregating servers in each of the three scenarios is that they serve as a centralised platform for external networks, providing access to information collected from internal OPC UA nodes. This advantage becomes clear when considering a scenario where many clients access multiple, often the same, servers: managing multiple connections and users with varying privileges, each requiring the enforcement of individual security rules, demands significant resources [17].

By using an aggregating server, the number of connections and users is also reduced to one per client, which also simplifies security management. Maintaining only a single connection is an additional advantage for resource-constrained devices with real-time tasks, such as PLCs, which may only be capable of handling a limited number of simultaneous connections.

C. REFERENCE ARCHITECTURE OF AN INDUSTRIAL SYSTEM

This section presents a reference architecture to analyse and demonstrate a security-hardened aggregation service that connects legacy devices to an OT network.

The industrial reference architecture shown in Figure 2 is based on the Josef Ressel Centre (JRC) for Intelligent and Secure Industrial Automation (ISIA) testbed, a testbed for industrial research in which all OT devices use the OPC UA protocol. The testbed is structured in accordance with the state of the art, following the IEC 62443 [3] and the Purdue reference architecture [6]. It consists of four security zones that are each connected through a single conduit via firewalls: the *enterprise* zone, the *industrial* DMZ, the *shop floor* zone, and the *legacy* zone.

The process reflected in this testbed is the production of plastic parts with Injection Moulding Machines (IMMs). The three IMMs are each controlled by one PLC (*PLC1-3*) on the shop floor and a SCADA HMI in the operational zone.

All OT devices communicate via the OPC UA protocol. Three industrial computers (*KUKA1–3*) each control a robot arm to move finished parts from the IMMs and place them on a conveyor belt; a fourth robot arm (*KUKA4*) then puts them on pallets at the line's end. Although the IMMs and robot arms are simulated, the control logic executes identically to a real-world deployment. The *enterprise* zone comprises a resource-planning tool and an intelligent assistant (part of separate research on intelligent automation), whereas the *legacy* zone, the focus of this research, connects via a firewall and hosts the proposed SigmaServer server, one PLC (*PLC4*), and a Raspberry Pi emulating an insecure Modbus temperature controller (*PLC5*).

Taken as a whole, the testbed setup models a modern industrial manufacturing process using state-of-the-art protocols (OPC UA) to advance research in cybersecurity and AI-driven automation and anomaly detection.

TABLE 1. Criteria for evaluating the identified concepts for OPC UA aggregation. The criteria range from security properties to operational feasibility as the concepts are evaluated for their applicability in industrial networks.

Criterion	Description
Threat mitigation	Degree to which STRIDE threats are addressed within the threat model (Section V-A).
Namespace integrity	Ability to preserve unique, device-specific OPC UA namespaces, avoiding namespace pollution and conflicts in static node IDs.
Real-time impact	Effect on real-time communication requirements.
Integration effort	Complexity of deploying and configuring the concept, including firewall adjustments, port mappings, and modifications to existing industrial software.
Zones and conduit compatibility	Alignment of the concept with the Purdue reference model and IEC 62443 zone-and-conduit principles.

III. METHODOLOGY

The methodological approach used to investigate the secure integration of legacy systems is guided by two research questions:

RQ.1 *What security and architectural solutions exist for integrating legacy systems into OPC UA-based industrial automation networks?*

RQ.2 *Which integration approaches offer the best balance of security and operational feasibility for legacy system inclusion in industrial networks that follow the Purdue model? And how can such an approach be realised?*

To address **RQ.1**, a structured literature review is conducted with the goal of identifying the state of the art regarding published middleware aggregation solutions. The review aims to provide a broad picture of current approaches presented in research publications for aggregating or retrofitting actuators, sensors, and PLCs. To achieve this, the following search terms are used: *opc ua, security, aggregation, aggregating, server, brownfield, retrofit*. The search is conducted via Google Scholar to minimise potential bias in favour of any single publisher [19]. Consequently, the identified publications are classified according to characteristics relevant to security and architecture, as detailed in Section IV.

With the state of the art established, **RQ.2** puts focus on underlying architectural concepts for integrating legacy systems into modern industrial networks. Based on a derived threat model, three architectural concepts for integrating insecure legacy systems into a secure, Purdue-compliant network are qualitatively assessed for security and operational feasibility (see Table 1).

Following the identification of the most suitable concept, a proof-of-principle is to be realised in an industrial testbed. The proof-of-principle is then to be evaluated using metrics such as latency and system load, to provide insight into the trade-offs between security and operational overhead.

IV. STATE OF THE ART SOLUTIONS

The two topics of OPC UA aggregating servers and retrofitting brownfield systems have already been discussed in depth in the past and are presented in the following sections (Section IV-A and IV-B). The literature summary in Section IV-C shows that when combining both areas, no research publication was found, which evaluates aggregating servers as a retrofitting solution.

The scope of this review is limited to peer-reviewed research publications, as the primary goal is to assess the state of scientific knowledge and identify open research gaps. Commercial products are outside the scope of this review, as their internal security architectures and design are typically not disclosed in a form that permits scientific evaluation and reproducible comparison.

Table 2 provides a summary of the state of the art as found by the literature review in January 2026. The evaluated characteristics indicate whether a solution focused ●, considered ◒, or did not address ○ the topic. Furthermore, for works that identified security, yet did not focus or consider it, ◔ is used. An example for ◔ is when security is only mentioned as future work.

To clarify the security perspective of the existing work, the analysis is divided into two subsections. Section IV-A examines how security is treated in existing OPC UA aggregating server solutions. Section IV-B reviews security aspects across all identified retrofitting approaches for brownfield systems, where insecure protocols and outdated systems introduce different challenges.

A. SECURITY IN OPC UA AGGREGATING SERVER SOLUTIONS

We identified 16 publications for comparison which focus on aggregating services for OPC UA, which are listed in Table 2.

Großmann et al. [17] present the basic concept of an OPC UA aggregating server and propose a reference architecture for such a server. They developed two information model extensions (for the aggregation- and the aggregated server) and implemented a prototype. Besides a *security manager* for access control, no further security aspects are discussed.

Banerjee and Großmann [22] make use of the same prototype as in [17] and use it as aggregation layer to unify data from underlying, heterogeneous services with different information models. Security is not discussed in this publication.

Crespí et al. [16] use an aggregation server as a bridge between cloud services and the shop floor level (edge devices) and provide a single, homogeneous information model. They go into more detail regarding how the aggregated address space is constructed. For future work, they mention the implementation of security mechanisms.

In a research paper by Weskamp and Tikekar [30] from 2021 an Industry 4.0 compliant OPC UA aggregating server is presented. They use the same architecture as in [17] as base and map the underlying information models to the

TABLE 2. Literature overview on state of the art publications (until January 2026) on aggregating servers and brownfield/retrofitting systems ordered by publication year.

Ref.	Year	Authors	Arch.	IM	Sec.	T. crit.	Code	Eval.	Leg.	Bf.	Comment
Aggregating Servers											
[20]	2014	Fernbach *et al.*	○	●	○	○	○	●	○	○	Presents a comprehensive approach to connect buildings and industrial automation systems using a cross-domain information model.
[17]	2014	Großmann *et al.*	●	●	◒	○	○	●	○	○	Present a proof-of-concept implementation and highlight that aggregating servers bring benefits for security.
[21]	2016	Seilonen *et al.*	●	●	○	○	○	●	○	○	Focuses on the transformation of the information model.
[22]	2017	Banerjee and Großmann	●	●	○	●	○	○	○	○	Uses the prototype of [17] and focuses on unifying data from underlying, heterogeneous services.
[23]	2017	Graube *et al.*	○	●	◔	●	○	○	◒	◒	Focuses on the challenges of mirroring several OPC UA servers by using co-simulation.
[24]	2018	Wang *et al.*	●	●	○	○	○	●	○	○	Focuses on the cache management.
[25]	2018	Würger *et al.*	○	●	○	○	○	○	○	○	Uses AutomationML to model the hierarchical structure and the subsequent mapping to an address space for energy data integration.
[26]	2019	Breunig and Schneider	●	●	◔	○	○*	●	○	○	Focuses on multi-protocol functions.
[16]	2019	Crespí *et al.*	●	●	○	●	○	○	○	○	Security is only mentioned in future work.
[27]	2019	Peña *et al.*	○	○	○	●	○	●	●	●	Presents an OPC UA/Modbus gateway solution.
[28]	2020	Li *et al.*	●	●	○	●	○	●	●	○	Focuses on an architecture over TSN.
[29]	2020	Mathias *et al.*	◒	●	○	○	○	○	○	○	Focuses on aggregation dynamics of OPC UA for connecting and accessing multiple database servers.
[30]	2021	Weskamp and Tikekar	●	●	○	●	○	●	○	○	Uses the architecture from [17].
[31]	2022	Pu *et al.*	●	◒	○	●	○	●	○	○	Presents a detailed description of the architecture.
[32]	2024	Busboom	●	●	○	○	○	○	○	○	Reviews approaches for automated OPC UA information model generation.
[33]	2026	Knobloch *et al.*	●	●	◔	◒	○	●	○	○	Presents hash-based type aggregation and reference injection for instance aggregation.
Brownfield and Retrofitting											
[34]	2020	Etz *et al.*	●	●	◔	●	○	○	●	●	Focuses on implementing an OPC UA gateway to bridge telnet with a legacy device using 'python-opcua'.
[35]	2020	Lackorzynski *et al.*	●	○	●	●	○	●	●	●	Proposes MACsec modifications for securing legacy industrial Ethernet at ISO/OSI layer 2.
[36]	2021	Rupprecht *et al.*	●	○	◔	○	○	○	●	○	Evaluates retrofitting legacy PLCs into Industry 4.0 with a hardware gateway.
[37]	2022	Majumder *et al.*	●	●	○	○	○	●	●	●	Uses PLC4X middleware to integrate legacy systems into Industry 4.0.
[38]	2022	Tran *et al.*	●	○	●	○	○	●	●	●	Reviews retrofitting brownfield systems, highlighting security challenges, threats and solutions.
[39]	2024	R *et al.*	●	◒	●	●	○	●	●	●	Presents a three layer architecture for integrating legacy systems with OPC UA.
[40]	2025	Holmes *et al.*	●	●	◔	○	○	○	●	○	Describes a digital retrofitting framework using OPC UA and a layered RAMI 4.0 architecture.

Arch.: Architecture; **IM:** Information Model; **Sec.:** Security; **T.crit.:** Time Criticality; **Code:** Published Code; **Eval.:** Evaluation; **Leg.:** Legacy devices; **Bf.:** Brownfield devices; *the git project does not exist anymore.

metamodel of the Industry 4.0 Asset Administration Shell (I4AAS) companion specification. Security is not discussed in this publication. Version 1.0 of the OPC UA specification for compliance with I4AAS is defined in OPC 30270 [41] and was published the same year (2021).

Peña et al. [27] presents a Modbus to OPC UA gateway solution implemented on a Raspberry Pi. The proposed gateway enables communication between legacy Modbus based devices and modern OPC UA clients but differs from our work by focusing on protocol translation rather than on securely bridging isolated network zones.

Other research, such as [21], [23], [24], [28], or [31] make also use of aggregating servers, but do not discuss security aspects any further.

The GitHub project of the OPC Foundation [42] provides open-source code for sample OPC UA applications that conform to the standard. In particular, it includes an implementation of an OPC UA aggregating server, which we use as the baseline in our experimental comparison (see Section VIII). As an example of a commercial solution, the UaGateway is offered by Unified Automation [43], providing secure communication between devices using the legacy OPC protocol and those using OPC UA.

Both of these solutions, i.e., the OPC Foundation's sample aggregating server solution and the commercial UaGateway, differ from our proposed aggregating server solution presented in *C.3* (see Section V-D) as we also consider other protocols that are neither OPC nor OPC UA. Furthermore, the proposed SigmaServer provides separate secure OPC UA server instances per device.

B. SECURITY IN RETROFITTING SOLUTIONS

Seven relevant publications on retrofitting of brownfield systems with OPC UA were identified in the review.

Rupprecht et al. [36] evaluate multiple retrofitting concepts to integrate legacy PLCs in a modern, Industry 4.0 compliant environment and provide their data to higher-level systems. One of the approaches places an additional hardware gateway at field level, which connects to the PLC via its proprietary protocol and to the intranet via OPC UA. Security is not discussed in this publication.

Etz et al. [34] implement a similar solution: an OPC UA gateway which bridges a telnet connection of a legacy device to an OPC UA domain. The implementation is done using the information model from the manufacturer and the library *python-opcua*. Security is not discussed in this publication.

Majumder et al. [37] use *PLC4X* as a middleware to integrate legacy systems with different proprietary protocols into an Industry 4.0 environment using modern, standardised protocols such as OPC UA, Message Queuing Telemetry Transport (MQTT), Hypertext Transfer Protocol Secure (HTTPS), etc. However, they conclude, that this tool is unable to handle the different data semantics and data heterogeneity of underlying devices and is therefore not suitable as a standalone tool.

Tran et al. [38] present a review on previous work regarding retrofitting brownfield systems. The authors include typical challenges, with one of them being security. They mention problems with legacy machines, limited security functions and security threats and their possible solutions. They also note, that there are only few studies, which include security aspects. Considering that the use of legacy systems is still very common, they mention that security is an increasingly important factor for the future.

In contrast to other solutions, Lackorzynski et al. [35] make use of a modified MACsec protocol to secure communication on layer 2. This has the advantage that industrial protocols based on Ethernet can be secured on the ISO/OSI layer 2. However, using an Ethernet based security mechanism does not address the problem of the continued use of legacy protocols within the industrial network. Moreover, it requires the non standardised use of the MACsec Ethernet frame.

Busboom [32] presents a review on works that generate OPC UA information models in an automated fashion. The publication includes information in aggregating multiple OPC UA information models, namespace indices, merging types, etc.

In [39], the authors use a three layered architecture for integrating various legacy devices into Industry 4.0. This publication focuses on the security aspect, but differs to the solution presented in this work.

Peña et al. [27] demonstrated an OPC UA/Modbus gateway for energy recovery systems but did not address the secure integration of other proprietary protocols.

Holmes et al. [40] propose a general digital retrofitting framework following a layered RAMI 4.0 architecture, using OPC UA as the standard for modular and distributed communication across heterogeneous legacy machines.

Knobloch et al. [33] propose a hash-based method for type aggregation and an instance aggregation approach using reference injection. While they validate this mapping database design on a robotic assembly station, their focus remains on vertical integration within secure networks.

C. SUMMARY

In summary, of the 23 publications reviewed, only three focus specifically on security, one considers it to some extent, and six identify a need for further security measures in the future. The characteristic *Code* (see Table 2), highlights that none of the identified publication provide their implementation as open-source, thus limiting the reproducibility.

Furthermore, we can summarise our findings as follows:

- OPC UA aggregating servers provide an effective solution for integrating multiple underlying OPC UA servers into a unified, aggregated address space.
- Retrofitting brownfield systems with OPC UA gateways is a common approach for bridging the gap between legacy systems and modern, Industry 4.0-compliant infrastructures.

- Security in retrofitting brownfield systems is often a secondary concern and receives only limited attention.
- None of the proposed works have published a proof-of-principle, hindering the research community's ability to evaluate these solutions or build upon them effectively.

D. POSITIONING AMONG SECURITY-AWARE APPROACHES

The majority of the 23 reviewed publications do not address security at all, four works either focus on or consider it, i.e., [17], [35], [38], and [39]. These are the only publications allowing a more detailed comparison on their security features with SigmaServer. A summary of the findings is presented in Table 3.

Tran et al. [38] provides a comprehensive review of security challenges in brownfield retrofitting. As this work is a review rather than an implementation of an aggregating or brownfield solution, it is not considered in further detail in Table 3.

Großmann et al. [17] introduce the concept of an OPC UA aggregating server and propose a dedicated security manager for access control. However, their architecture merges all device namespaces into a single aggregated address space, which leads to the problem of *namespace pollution*. In the context of security, no explicit threat model is defined and security is limited to the access management on the aggregating server itself. Furthermore, their concept does not support any legacy or insecure fieldbus protocols such as Modbus/TCP.

Lackorzynski et al. [35] propose modifications to the MACsec protocol to enable data link layer authentication and encryption in legacy industrial Ethernet environments. While their work directly addresses security, it operates at ISO/OSI layer 2 and does not address transport- or application-layer protocol security, namespace integrity, or threat modelling. It therefore targets a different layer of the network stack and serves as a complementary security measure rather than an alternative to the OPC UA bridging approach realised by SigmaServer.

Sujith et al. [39] propose a three-tier architecture that uses OPC UA wrappers and adapters to bridge legacy protocols to a central OPC UA server, with encryption and authentication applied on the secure side. Of the three works, this is the most architecturally similar to SigmaServer, and it is the only one to provide both transport security and multi-protocol support. Nevertheless, it employs a single centralised OPC UA server that aggregates all device data into a shared address space, thereby inheriting the problem of namespace pollution. The work also does not define a threat model to support its security design.

In contrast, SigmaServer addresses all of these criteria: it combines strict security mechanisms, per-device secure OPC UA endpoints, namespace integrity, multi-protocol support and a three-level STRIDE-based threat model. These properties show that no prior work in the reviewed literature addresses the secure bridging of insecure legacy systems in a comparably comprehensive manner.

TABLE 3. Comparison of related works addressing security in OPC UA aggregation and brownfield integration against SigmaServer.

Dimension	Assessment
Großmann et al. *[17]*	
Security mechanism	Access control via a dedicated security manager; no transport encryption
Namespace integrity	All device namespaces merged into a single aggregated address space (namespace pollution)
Protocol support	OPC UA only
Threat model	Not defined
Lackorzynski et al. *[35]*	
Security mechanism	Modified MACsec protocol providing layer-2 authentication and encryption
Namespace integrity	Not applicable (operates at ISO/OSI layer 2)
Protocol support	All Ethernet-based industrial protocols
Threat model	Not defined
R et al. *[39]*	
Security mechanism	Encryption and authentication
Namespace integrity	Single centralised OPC UA server; namespace pollution not avoided
Protocol support	OPC UA, Modbus, PROFIBUS or RS-232
Threat model	Not defined
SigmaServer	
Security mechanism	OPC UA SecurityPolicy *Basic256Sha256* with certificate-based authentication*
Namespace integrity	Per-device OPC UA endpoints; namespace pollution eliminated
Protocol support	OPC UA and Modbus/TCP; extensible to further protocols
Threat model	Three-level STRIDE-based threat model (see Section V-A)

*Source code can be modified as all profiles in the open62541 library are supported.

V. CONCEPTS FOR INTEGRATING LEGACY SYSTEMS

In general, aggregating servers are designed to provide a centralised node for managing access from external clients [44]. Legacy device integration typically lies outside the regular scope of aggregating servers and is usually realised via gateways connecting to the shop floor, e.g., [34]. To analyse the different concepts for OPC UA-based aggregation in industrial networks, we first discuss the threat model that is considered (Section V-A). Afterwards we discuss the identified concepts (C.1-3) for aggregation in Sections V-B to V-D. Table 4 summarises the results of the qualitative assessment of each concept presented in Sections V-B to V-D.

A. THREAT MODEL

Attacks are becoming increasingly sophisticated, as evidenced by the rising number of incidents in the OT domain [45], [46]. Thus, we assume a skilled adversary at three different stages. In the first stage, the attacker attempts to penetrate the network remotely via the internet. Next, the attacker gains a foothold in the industrial DMZ, and finally, the attacker compromises one of the legacy systems.

Figure 3 illustrates an industrial automation network inspired by the Purdue model and the location of the attacker, which is highlighted with (1) - (3). The black box between the industrial DMZ and the legacy zone is a placeholder for the identified three concepts for aggregation (Sections V-B to V-D).

Attacker Level 1. The attacker is positioned outside the organisation and can perform Denial of Service (DoS) or privilege escalation attacks. Regarding the latter, an example may be that the attacker manages to exploit a vulnerability in an application, which allows them to get further access into the system (attack level 2).

Attacker Level 2. The attacker has gained access to a system located in the industrial DMZ, allowing them to execute DoS and privilege escalation attacks against the legacy zone. This scenario can be divided into two cases: (i) the attacker obtains physical access to the network through social engineering (e.g., by posing as an engineer or service staff) and compromises a device, or (ii) the attacker exploits a firewall vulnerability to gain network access. In both cases, the attacker can not only perform DoS and privilege escalation attacks, as in attacker level 1, but also conduct Spoofing, Tampering, Repudiation, and Information Disclosure attacks against the legacy zone.

Attacker Level 3. Through social engineering or other actions like infection through USB flash drives (as demonstrated with Stuxnet [47]), the attacker manages to access the legacy zone, hence they are able to perform attacks associated with all six STRIDE threats.

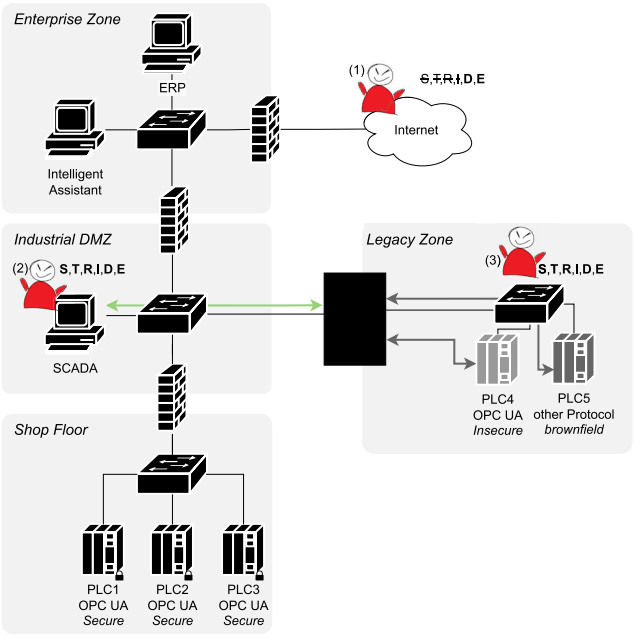


FIGURE 3. **Threat model illustrating three different attacker levels while considering basic security controls enabled.**

B. CONDUIT TO A LEGACY ZONE (C.1)

In the context of integrating a legacy zone into an industrial network, one common approach is to use conduits, such as firewalls or one-way gateways (data diodes), to manage and secure access to and from the legacy devices.

By employing a firewall or data one-way gateway as a conduit, all data exchanged between the legacy zone and other network segments can be monitored, filtered, and restricted based on predefined security policies. This configuration minimises the risk of unauthorised access or malware propagation from the legacy devices to the broader network. The firewall can enforce rules such as allowing only specific protocols, limiting communication to essential systems, or implementing intrusion detection and prevention mechanisms. From a performance perspective, this concept is comparatively time efficient, as the conduit merely forwards or blocks traffic based on its rules without requiring additional processing.

However, these approaches have some significant disadvantages. Clients from the "secure" zones, the industrial network, may still need to communicate using outdated and insecure protocols that are compatible with the legacy devices, exposing them to potential risks. In particular, once an attacker has gained a foothold in the industrial DMZ, they can abuse permitted insecure protocols to carry out spoofing, tampering, and information disclosure attacks. Therefore, Deep Packet Inspection (DPI) would be required to enhance security by thoroughly analysing the data traffic for malicious content, requiring more resources. As data diodes only permit unidirectional communication, they additionally limit the functionality of applications that rely on bidirectional interaction, such as remote control, configuration changes, or acknowledgements. Moreover, information disclosure attacks remain unaddressed, as data diodes do not provide any cryptographic protection.

C. APPLICATION-LEVEL AGGREGATING SERVER (C.2)

The second concept for connecting a legacy zone to an industrial network involves the aggregation on application level, which is commonly implemented using gateways/proxies or specialised solutions like OPC UA aggregating servers. These aggregating servers are specifically designed for scenarios where external entities need to access, both read and write, OPC UA nodes within the industrial automation network.

This approach not only simplifies management and monitoring but also enhances interoperability by presenting data in a standardised format, reducing the need for direct interaction with insecure legacy protocols. As a consequence, only secure protocols are exposed within the secure zones (e.g., industrial DMZ), which already improves the overall security posture of the network. Despite these benefits, the effectiveness of this scenario depends on the capabilities of the aggregation logic and the completeness of the device-specific data mapping. Moreover, aggregating servers were neither designed to be (i) used for handling access to legacy systems within the network, and (ii) used as a security solution.

A challenge of this approach is *namespace pollution*, as the namespace is no longer unique to individual devices. Instead,

it becomes a shared space, which can lead to conflicts, ambiguities, or difficulty in managing and distinguishing data from different devices. Graube et al. [23] further detailed some disadvantages. One notable issue mentioned is that industrial automation software is typically designed with a *static, fixed namespace*, which relies on constant node IDs for access. The *persistence* of these node IDs can pose a challenge as well, as changes to the merged information model may affect the order of nodes represented by the aggregating server during runtime. In practice, this also increases the integration effort, since PLC projects must be adapted to the modified aggregated namespace in order to access the underlying devices via the aggregating server.

Another challenge which needs to be taken into account is that an application-level aggregating server may introduce *timing issues*. Particularly in cases when PLCs are required to communicate with each other in real-time, potentially affecting the overall performance and responsiveness of the industrial system, which often relies on precise and synchronised operations.

D. TCP-LEVEL AGGREGATING SERVER (C.3)

Individual namespaces for each device within the legacy zone can be maintained by distributing them at the transport layer (e.g., TCP) rather than at the application layer, as is done by OPC UA aggregating servers (*C.2*). This solution requires multiple OPC UA servers running on the aggregating server, allowing devices to be distinguished through their respective TCP ports.

C.3 is similar to a protocol gateway that exposes individual device endpoints on dedicated TCP ports (e.g., [48]), with the key difference that *C.3* supports heterogeneous legacy protocols beyond a single protocol family like Modbus, by incorporating protocol-specific client logic behind each secure OPC UA endpoint.

Timing constraints still need to be accounted for, as this solution does not eliminate all the previously mentioned disadvantages of *C.2*. However, it effectively addresses issues such as *namespace pollution* and the reliance of industrial automation software on a *static, fixed namespace*.

Challenges arise from the dynamic deployment of multiple OPC UA servers. One such challenge is the need to map the TCP ports to devices in the legacy zone using static mappings. These static mappings are necessary to avoid re-configuring PLCs whenever changes occur within the legacy zone. Once the ports are configured, however, the PLCs do not require further adjustments. As a result, the integration effort is lower than for *C.1* and *C.2*.

Another challenge stems from the use of separate TCP ports for each legacy device. Specifically, integrating new PLCs – particularly a large number of them – may be more time-consuming and may require more adjustments to the firewall compared to *C.2* in which only one TCP port is used. At the same time, hosting dedicated OPC UA server instances per legacy device improves the robustness against attacks originating from the secure zones, as the OPC UA server instances are isolated, with each running its separate instance of the OPC UA stack.

As for any other concept presented for our use case, a TCP-level aggregating server must reflect the current status of the legacy devices requiring it to communicate via the legacy protocol. The monitoring of the operational state of these devices is essential, due to the otherwise high time delay for accessing the OPC UA server representing the legacy device.

E. ADDITIONAL SECURITY MEASURES

Regardless of the discussed concepts, OPC UA must follow state-of-the-art security measures when used in production environments. Moreover, the legacy zone, as an insecure network, must be additionally protected. Network filters and deep packet inspection may be required within the legacy zone, especially when considering attacker level 3 in the threat model (see Section V-A).

Additional measures to secure the vulnerable legacy zone include the deployment of honeypots. For instance, a network filter facing the secure zone can redirect suspicious traffic to a sandboxed honeypot [49]. The advantage of deploying a honeypot is that it allows one to immediately detect, record and learn from the attacker's behaviour.

VI. SigmaServer

The architecture of SigmaServer is built upon the TCP-level aggregating server design (*C.3*) due to its properties summarised in Table 4. *C.3* is presented in detail in Section V-D. It mitigates *namespace pollution*, and minimises modifications to existing industrial software relying on a fixed namespace [23].

A. ARCHITECTURE

SigmaServer demonstrates a proof-of-principle implementation and is implemented with the open62541[3] (v.1.4.8) library. Its modular design is based on three main components: multiple insecure clients (*InsecClients*), a central thread-safe data store, and multiple secure servers (*SecServers*). The overall architecture is shown in Figure 4.

Each InsecClient is responsible for a single legacy device, such as a PLC or another system that relies on insecure or outdated communication protocols. The InsecClients operate in separate threads and establish connections to the assigned legacy endpoints. They retrieve both structural information and real-time data values, which are stored in the central thread-safe data store. The current version of SigmaServer supports OPC UA and Modbus/TCP protocols. However, the architecture allows the integration of additional insecure communication protocols, such as CAN or PROFINET, by adding further InsecClient variants in the publicly available open-source codebase.

The shared storage functions as the central data buffer of the system, maintaining two synchronised thread-safe maps:

[3]https://github.com/open62541/open62541/

TABLE 4. Decision matrix for the architectural concepts (*C.1 - C.3*). *C.3* performs best by also addressing namespace integrity, and requiring a lower integration effort.

Criterion	Justification	C.1	C.2	C.3
Threat Mitigation	*C.3* provides the strongest isolation by hosting an individual secure OPC UA server instance for each legacy device.	Medium	Medium	High
Namespace Integrity	*C.2* merges address spaces, causing pollution; *C.1* and *C.3* preserve individual namespaces.	High	Low	High
Real-Time Impact	Evaluated by the latency introduced for real-time PLC communication. *C.1* avoids application-layer remapping, keeping the overhead minimal.	Low	Medium	Medium
Integration Effort	*C.1* requires more extensive security configurations of firewalls including DPI. *C.2* and *C.3* have reusable server components, however, *C.2* additionally requires more extensive adaptation of industrial systems.	High	High	Medium
Zones and Conduit Compatibility	*C.1* allows insecure traffic across the zone boundaries. *C.2* and *C.3* act as compliant conduit nodes.	Medium	High	High

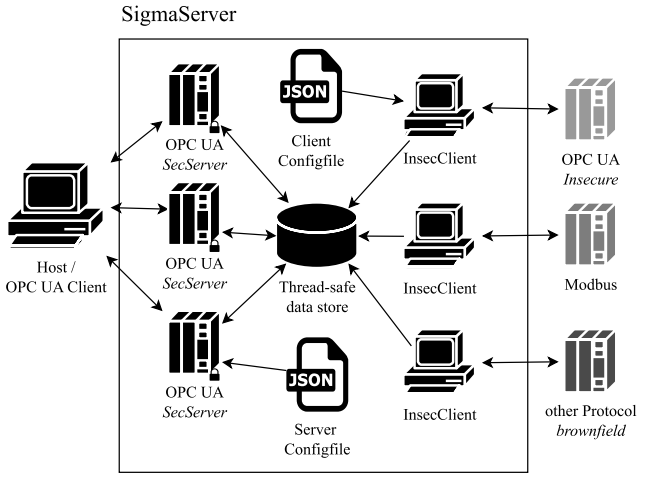


FIGURE 4. Architecture of the proposed SigmaServer illustrating the separation of insecure client interfaces and multiple secure OPC UA endpoints.

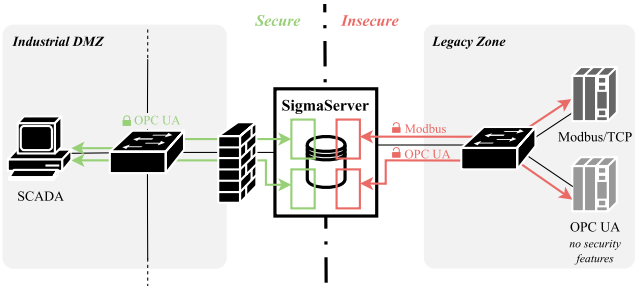


FIGURE 5. Utilisation of SigmaServer as a bridge between the secure zone and the insecure legacy zone.

one for the static node structure and another for current data values. This design enables strict separation between the insecure and secure zones. All structures and data are mirrored to JSON files under *config/<alias>/namespace<N>.json* allowing independent inspection of the running system.

The SecServers host a secure OPC UA server instance per legacy device. Each SecServer runs on a dedicated TCP port (e.g., 4841) and provides a unique OPC UA endpoint to represent a single device from the insecure network. This approach guarantees full namespace separation, thereby eliminating the problem of namespace pollution and ensuring a clear one-to-one mapping between legacy devices and their secure OPC UA representations. Thus, hosts in the secure zone can access legacy devices by reconfiguring only the TCP port used for communication. Communication between clients in the secure zone and the SigmaServer is encrypted and authenticated using OPC UA SecurityPolicy *Basic256Sha256*, supporting both certificate-based and username/password authentication. As the implementation is based on open62541, any other security profile supported by the library can be applied by modifying the source code.

At system startup, the SigmaServer loads two configuration files: the client configuration defines all insecure endpoints, node IDs, and Modbus registers to be queried. The server configuration specifies which secure servers to start, along with their corresponding aliases and TCP ports. After initialisation, each InsecClient retrieves the node structures from its source device and fills the shared storage with values. The SecServers then register these structures into their local OPC UA address space and start providing real-time values to the host systems such as SCADA.

Figure 5, shows the positioning of the TCP-level aggregating server, acting as a bridge between secure and insecure network zones. Both network zones are connected to a network interface of the SigmaServer.

B. OPERATION PRINCIPLE

When the SigmaServer starts, the main process loads the configuration files and creates a separate thread for each client and server defined in the setup. Each InsecClient thread connects to its assigned device. For OPC UA devices, the client first reads the NamespaceArray (standard node i=2255) to map each namespace index to its URI. This information is stored in memory as `<Alias>:<NsIndex>` → `<URI>` and later used by the secure servers to reconstruct the original namespace table. The client subsequently reads the configured node attributes such as *DisplayName*, *Description*, *DataType* and *BrowseName*. These attributes are queried

using the open62541 read service and saved together with the nodes' numeric identifiers (*NamespaceIndex*, *NodeId*). The complete browse path is constructed by recursively traversing parent-child relationships up to the root folder. Each path is prefixed with the device alias (for example `Objects/PLC21/...`) so that every device has its own subtree in the global structure.

This structure is stored in a thread-safe map and also written to JSON files under *config/<Alias>/Namespace<N>.json*. These files contain a minimal list of all node identifiers for the given namespace and allow dynamic adjustments to the currently needed nodes. Once reading the node structure is finished, the client switches into a cyclic read loop. It continuously reads the values of all configured nodes or, in the case of Modbus devices, the defined registers. The values are normalised into basic OPC UA types: Boolean, Int16/32/64, Float, Double, String, or DateTime. For timestamps, the current system time is stored as an OPC UA DateTime value or as Unix time, depending on the source. Modbus registers are converted according to the configuration, for example scaling a raw integer by 0.1 to represent a temperature value. The resulting data is written atomically to the global value map, replacing the previous entry. Each entry is identified by a deterministic key of the form `<Alias>:<NamespaceIndex>:<NodeId>`.

The shared storage acts as the only communication interface between insecure clients and secure servers. It is implemented as a thread-safe key-value map protected by internal mutexes. This ensures consistent access even when multiple threads write simultaneously. Clients only write new values, while the secure servers only read them. No direct network communication exists between these two domains, which ensures full isolation between the insecure and secure network zones. The current implementation focuses exclusively on read operations from legacy devices to secure clients.

On the secure side, each SecServer thread starts once the structure data is available in memory. During its initialisation, it reads all stored namespaces for its assigned alias and adds them to its own OPC UA server configuration. Each variable node is created with the same numeric NodeId(ns, id) that was read from the original device. Folder hierarchies are recreated from the stored browse paths. A read callback is registered for every variable. When a secure OPC UA client requests a value, the callback retrieves the corresponding JSON object from the shared data map, converts the stored value to the correct data type based on the saved DataTypeId, and returns it as a UA_Variant. This conversion supports all common scalar types and ensures that data are presented exactly as they were read from the original device. The server's namespace table is rebuilt from the stored URIs, and the OPC Foundation standard namespace (`http://opcfoundation.org/UA/`) is not mirrored to avoid collisions.

Each client and server operates in its own thread and is designed with a stoppable control mechanism that enables proper shutdown. In the case of a connection loss or device failure, the affected client continuously attempts to re-establish communication with its assigned endpoint. The reconnection mechanism is handled within the regular polling loop and does not require manual actions.

In addition to the conventional client–server mode, the implementation also supports a publish–subscribe mode. In this configuration, the secure OPC UA servers act as publishers, periodically transmitting aggregated data to one or more subscribers within the secure zone. The operational mode can be switched between the two modes directly through the server configuration file *server_configuration.json*.

The implemented SigmaServer is fully platform independent, which is achieved through containerisation using Docker. This approach allows the entire architecture to be deployed on both Windows and Linux hosts without any code modifications.

C. CONFIGURATION

SigmaServer is configurable through two JSON files. The client configuration file (*client_configuration.json*) defines all insecure legacy endpoints and the server configuration file (*server_configuration.json*) specifies the secure OPC UA server instances.

The client configuration declares one entry per legacy device, with the structure depending on the underlying protocol. For OPC UA devices, each entry specifies the device `alias` and the primary `endpoint` URL of the form `opc.tcp://<host>:<port>`. An optional fallback endpoint can be provided, which is used upon connection loss. The `pollIntervalSec` value defines the cyclic read interval, and a list of `nodes` identified by their `namespaceIndex` and `nodeId` specifies which OPC UA nodes are to be polled. For Modbus/TCP devices, the entry instead provides the `ip` address and `port` of the Modbus server, along with the polling interval. The `registers` list defines which holding registers are read, with each register specifying its `address`, a descriptive `name`, and a `datatype`.

The server configuration declares one entry per secure OPC UA endpoint. Each entry specifies a `port` number on which the SecServer will listen, the `plcAlias` referencing the corresponding client entry, and the operational `mode`. The mode can be set to either client-server (`CS`) or publish-subscribe (`PubSub`). The one-to-one mapping between a client entry and its server entry enforces the TCP-level separation described in Section V-D. Each legacy device is exposed on its own dedicated port and never shares a namespace with another device.

With this configuration, a secure OPC UA client can connect to a specific port (e.g. 4841) accessing exclusively the insecure OPC UA PLC, while another port (e.g. 4842) maps solely to the Modbus cooling system. No further reconfiguration on the secure OPC UA host is required beyond updating the target TCP port, which demonstrates

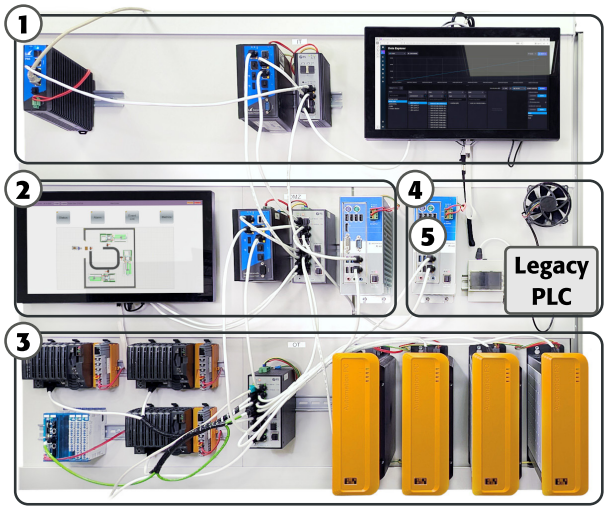


FIGURE 6. The testbed setup for the deployment is segmented into four zones, with the insecure legacy zone being integrated into the DMZ through the proposed SigmaServer.

how SigmaServer integrates into an existing industrial infrastructure with minimal operational overhead. A complete reference for all configuration parameters, including advanced options such as Docker deployment, latency test instrumentation and an example configuration, is provided in the project repository's `CONFIG.md` file.[4]

VII. EXPERIMENTAL SETUP

A. TEST ENVIRONMENT

To evaluate the operational behaviour and performance of the SigmaServer, the legacy zone implemented in the testbed of the JRC ISIA (see Section II-C) was utilised. The setup reflects a realistic replicable industrial environment, allowing data flow, latency and potential attack vectors to be evaluated under controlled conditions. The focus of the experiment lies on the connection between the secure network zone and the legacy zone which is connected through the SigmaServer. The physical setup is illustrated in Figure 6.

The SigmaServer is deployed on a Sigmatek IPC 511 running Windows 10 IoT Enterprise. This IPC is equipped with an Intel Celeron G1820 CPU, 8 GB RAM and two Ethernet interfaces. The interfaces are configured to provide the connection to the secure and legacy networks respectively. This hardware was selected to reflect a typical industrial production environment.

In the legacy zone, two legacy device types are used. One being a B&R X20CP1686X PLC acting as the IMM. For this experiment, this PLC is configured with an insecure OPC UA interface and access to multiple nodes. The PLC is polled by an InsecClient which first retrieves the namespace array and node attributes, then periodically reads configured values. The other legacy device is a Raspberry Pi 4 acting as a Modbus/TCP server representing a cooling system. The Modbus server is implemented using the pymodbus library[5] and provides two holding registers containing: the temperature in degrees Celsius (°C) from a DS18B20 sensor, and fan speed in revolutions per minute (RPM). The SigmaServer's Modbus InsecClient polls these registers at a configurable interval and writes normalised values into the central thread-safe data store.

On the secure side, two SecServer instances mirror the collected data and present it via secure OPC UA endpoints (TCP ports 4841 and 4842). A host system, in this experiment UaExpert, connects to these endpoints using the security policy *Basic256Sha256* with certificate-based authentication enabled.

The network topology is realised using an unmanaged switch in the legacy zone connecting the OPC UA insecure and Modbus server to the SigmaServer. A Barracuda F183R firewall was used to enforce strict TCP routing between the zones. Therefore, only required ports are allowed (i.e., TCP port 4841-4842).

For the evaluation described in Section VII-C, an external test client was used, running on an HP Z2 Tower G9 Workstation (Intel Core i7-12700, 32GB RAM). The client operated on a Linux system with a generic kernel (6.14.0-generic).

It is important to note that neither the SigmaServer Windows 10 IoT PC, nor the test client utilised a hard real-time operating system kernel. This setup was chosen to represent a typical industrial monitoring scenario where servers and clients operate on typical IT infrastructure.

B. ATTACK SCENARIO

Based on the threat model introduced in Section V-A, the following attack scenario illustrates the current security posture of the SigmaServer. The implemented architecture effectively mitigates attacks associated with attacker level 1 and level 2 adversaries. External attackers or compromised systems within the industrial DMZ cannot perform spoofing, tampering, repudiation or information disclosure attacks. Although the SigmaServer provides strong isolation through strict network segmentation as well as authenticated and encrypted communication, it cannot fully prevent DoS or privilege escalation attempts. Its multi-port design inherently distributes the processing load across independent secure server instances, limiting the potential impact of targeted DoS attacks to individual endpoints rather than the entire system. Comprehensive defence against such attacks would still require additional protective measures.

The level 3 attacker, who gains access to the legacy zone itself, remains a realistic and critical threat. In the considered attack scenario, the adversary infiltrates the

[4] https://github.com/JRC-ISIA/opc-ua-sigmaserver/blob/main/docs/CONFIG.md

[5] https://pymodbus.readthedocs.io/en/latest/

production network via social engineering by impersonating a maintenance technician. Network access is obtained by connecting a computing device like a Raspberry Pi Zero to an unused Ethernet port on a switch linking critical production systems. The legacy zone includes both an insecure OPC UA server running on a PLC representing an IMM, and a Modbus/TCP cooling system as described in Section VII-A. Subsequently, the attack focuses on the Modbus protocol.

Using this position, the attacker carries out a spoofing attack, more specifically, a targeted ARP poisoning attack on the unsecured communication channel between the aggregation server and the Modbus server. This attack enables the adversary to manipulate the data transmitted from the Modbus server to the aggregation server. By utilising Ettercap and custom Etterfilter rules, the TCP payload of Modbus responses is continuously modified during transmission, causing the aggregation server to provide falsified data to the user.

For example, the attacker can present operators with stable and risk-free temperature readings from the cooling system, while in reality the device is overheating due to the insufficient thermal management. As a result, workers remain unaware of the serious condition, potentially leading to severe damage. In a real-world production environment, this could result into equipment failure, production downtime, and therefore significant financial losses.

Besides the demonstrated ARP spoofing attack, the analysis highlights that the attack surface of insecure field-bus protocols is diverse. Replay attacks, denial-of-service attempts on the Modbus port, or the deployment of rogue Modbus servers represent further realistic threat scenarios. In industrial environments, this does not only pose a security risk but also directly impacts safety, as manipulated process values can cause physical damage to equipment, production lines and endanger human life.

The presented proof-of-principle of SigmaServer cannot mitigate these weaknesses on its own but must be complemented by additional mechanisms such as Intrusion Detection Systems (IDSs). While the design and evaluation of such countermeasures are beyond the scope of this article, they represent a promising direction for future research.

C. EVALUATION METRICS

To evaluate the operational efficiency of the SigmaServer, three experiments were conducted focusing on (i) end-to-end latency between the insecure and secure interfaces under realistic operating conditions, (ii) internal software latency of the SigmaServer between InsecClient and SecServer, and (iii) average CPU and RAM utilisation compared to the OPC Foundation Console Aggregation Server for different aggregation scenarios.

End-to-end SigmaServer latency. The first experiment evaluates the end-to-end latency of the SigmaServer using the testing setup shown in Figure 7. This measurement includes the internal processing time of the SigmaServer and the network latency between the participating systems.

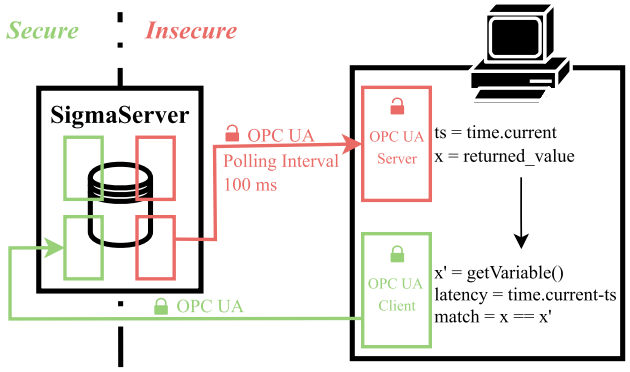


FIGURE 7. Testing setup for measuring the latency of SigmaServer and other solutions.

As illustrated in Algorithm 1, the external latency test client generates a random value each time it is polled by the SigmaServer's InsecClient. Immediately after this value is returned, the SecTestClient reads the corresponding value from the secure OPC UA endpoint of the SigmaServer. The latency is defined as the time difference between the moment a value is generated at the InsecTestServer and the moment the same value is successfully retrieved from the SecServer. This reflects a realistic scenario where data from an insecure legacy PLC flows through the SigmaServer to a secure OPC UA client.

To verify whether the values are matching, the test compares the received value from the secure interface with the returned value from the insecure interface. If both values match, the result is marked as *true*, otherwise, it is marked as *false*. This true/false validation ensures that only matching values are considered in the evaluation but also shows if the test client is fast enough for a proper validation.

Time measurements for the end-to-end latency were performed using Python's `time.monotonic()`, which utilises the underlying Linux CLOCK_MONOTONIC to ensure high-resolution monotonic timestamps.

Internal software SigmaServer latency. The second experiment focuses on the internal software latency of the SigmaServer. For this purpose, the implementation provides built-in software latency measurements around the shared data store. Two timestamps are recorded:

- Δt_1 measures the time between the arrival of a new value at the InsecClient and the completed write operation into the thread-safe data store.
- Δt_2 measures the time between the SecServer receiving a read request from a secure OPC UA client and the moment the requested value is fetched from the data store and is available on the SecServer for transmission.
- T_{proc} is therefore defined as the overall internal software latency, $\Delta t_1 + \Delta t_2$.

These measurements capture the internal processing behaviour of the SigmaServer between the insecure and secure sides, while explicitly excluding any polling interval, as well as any network latency. The

Algorithm 1 End-to-End Latency Measurement

Participants: InsecTestServer, SecTestClient, InsecClient, SecServer
Note: InsecTestServer and SecTestClient reside on the same external test device

InsecTestServer generates random value when read by SigmaServer
returnedValue ← InsecTestServer.getGeneratedValue()
ts ← current_time()
receivedValue ← SecTestClient.readFromSecServer()
latency ← current_time() − *ts*
if *returnedValue* = *receivedValue* **then**
    match ← true
else
    match ← false
end if
save(latency, match, returnedValue, receivedValue)

internal software latency T_{proc} was measured using the C++ `std::chrono::high_resolution_clock`, configured to record timestamps with microsecond resolution. Cryptographic operations required for secure OPC UA communication are not included in the internal software latency measurement.

Server resource utilisation. The third experiment evaluates the runtime efficiency of SigmaServer and compares it to the OPC Foundation Console Aggregation Server. To analyse scalability and resource behaviour under different workloads, four test scenarios were conducted, ranging from a single active OPC UA server to multiple concurrent OPC UA servers. Specifically, the measurements included configurations with one, two, and three simultaneously running OPC UA servers, as well as a fourth setup combining three OPC UA servers with one Modbus client to evaluate the mixed-protocol performance of the SigmaServer. Since the fourth setup requires Modbus handling, it can only be run on the SigmaServer.

All experiments were performed on the JRC ISIA testbed (see Section II-C) using the configuration described in Section VII-A.

VIII. RESULTS AND DISCUSSION

The performance evaluation of the SigmaServer was carried out according to the methodology defined in Section VII-C. All data were recorded under the same conditions.

A. END-TO-END SigmaServer LATENCY

A total of 14,287 samples were recorded for this experiment, with every single measurement resulting in a *true* match between the value generated on the insecure interface and the value read from the secure OPC UA endpoint. The measured latency values range from a minimum of 2.6 ms to a maximum of 24.7 ms with a standard deviation of 3.0 ms.

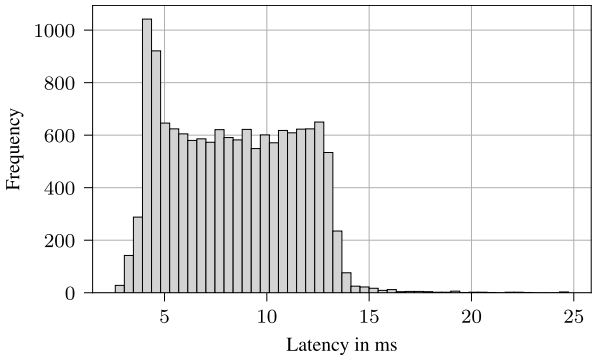


FIGURE 8. Distribution of the end-to-end SigmaServer latency measured with the test client between the insecure and secure interfaces.

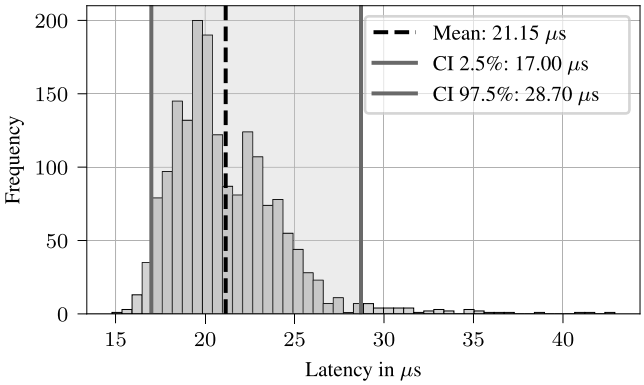


FIGURE 9. Internal software latency of the SigmaServer, showing the distribution of processing delay between InsecClient and SecServer.

Since no mismatches occurred, the results indicate that the SigmaServer consistently processes and forwards values faster than the external latency test client can resolve. Although the exact minimum latency of the SigmaServer cannot be determined due to the measurement limits of the test client, the results show that the complete data path, including the network transmission, is less than 2.6 ms. Figure 8 shows the latency distribution of this experiment.

B. INTERNAL SOFTWARE SigmaServer LATENCY

A total of 1,788 samples were collected to determine the internal software delay of the SigmaServer. The measured latencies range from a minimum of 14.80 μs to a maximum of 42.90 μs with a standard deviation of 3.11 μs. The overall internal software latency T_{proc} was measured at 21.15 μs. Figure 9 summarises the latency distribution and shows the corresponding 95% confidence interval. As this measurement excludes polling intervals, network latency and cryptographic operations, it isolates the pure computational overhead of the thread-safe data store and the data transfer between InsecClient and SecServer.

C. SERVER RESOURCE UTILIZATION

CPU and RAM utilisation were recorded over a one-hour period using the `docker stats` monitoring tool, with

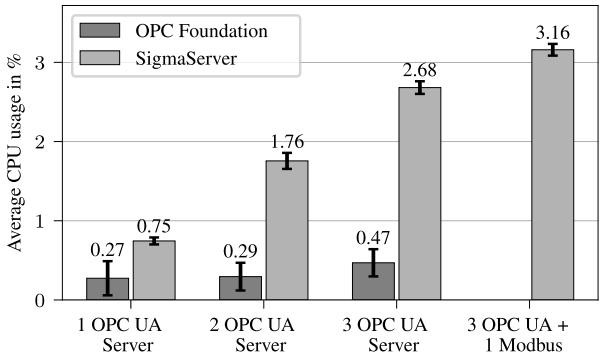


FIGURE 10. Average CPU usage of the SigmaServer compared to the OPC Foundation Console Aggregation Server.

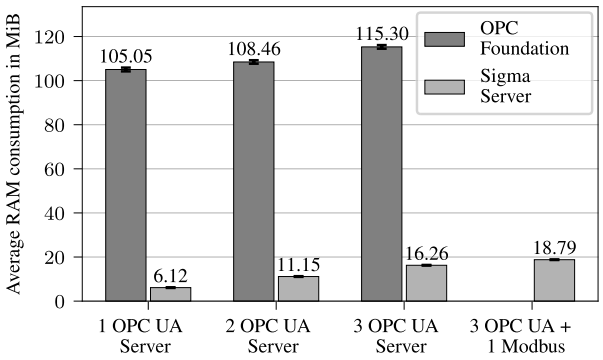


FIGURE 11. Average memory consumption of the SigmaServer compared to the OPC Foundation Console Aggregation Server.

sampling intervals of one second. The polling interval of the InsecClient was configured to 100 ms.

The average CPU load of the SigmaServer increased nearly linearly with the number of concurrently running server instances, ranging from 0.75% for a single OPC UA server to 3.16% when three OPC UA servers and one Modbus client were active. The RAM usage showed similar predictable behaviour, increasing from 6.12 MiB at lowest to 18.79 MiB at maximum.

Figure 10 presents the average CPU utilisation for all scenarios, while Figure 11 compares the RAM utilisation of the SigmaServer with the OPC Foundation Console Aggregation Server. Both implementations were configured with an identical OPC UA address space to ensure a fair comparison. As noted earlier, the mixed protocol scenario (OPC UA and Modbus) is only possible with SigmaServer, hence Figure 10 only includes the results for our solution.

D. DISCUSSION

The evaluation demonstrates that the SigmaServer requires only minimal computing resources while maintaining stable and predictable behaviour in all scenarios examined. The overall internal software latency, measured in the low

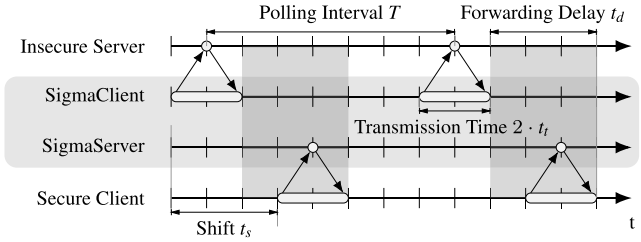


FIGURE 12. Timing diagram illustrating the realistic forwarding delay t_d introduced by the SigmaServer. A common polling interval of T is assumed, with each polling operation requiring $2t_t$. The polling activities are shifted by t_s, resulting in a forwarding delay $t_d \in [t_t, t_t + T)$.

microsecond range, confirms that the thread-safe data store and the modular client–server design operate efficiently and no significant delays occur.

The time complexity is determined by the operations on the shared data store: both Δt_1 and Δt_2 consist of a single key lookup and update on a `std::unordered_map`, which provides amortised $O(1)$ access by design. Since the map is fully populated during initialisation and no structural changes occur thereafter, rehashing is avoided entirely at runtime. The per-operation complexity is therefore strictly $O(1)$ in practice.

The resource utilisation results highlight clear architectural differences compared to the OPC Foundation Console Aggregation Server. SigmaServer uses multiple independent OPC UA server instances with TCP-level separation, while the OPC Foundation Console Aggregation Server provides a single, centrally aggregated address space. As a result, SigmaServer shows slightly higher CPU usage, it remains within a low and predictable single-digit range and scales linearly with the number of aggregated devices. At the same time, SigmaServer requires significantly less RAM and shows a noticeably lower variance in both CPU and RAM usage.

This observed linear scaling is consistent with the $O(n \cdot m)$ time complexity of the system, where n is the number of concurrently active InsecClient-SecServer pairs and m the number of nodes per device. Each pair operates in its own dedicated thread, with each InsecClient executing an independent polling loop over its m assigned nodes. Under the evaluated configurations, where m was held constant across all scenarios, $O(n \cdot m)$ reduces to the observed linear scaling $O(n)$.

Overall, the results confirm that the SigmaServer offers a reliable and resource efficient mechanism for bridging insecure field devices with secure OPC UA infrastructures, while keeping software overhead minimal and ensuring the elimination of namespace pollution. Together with the predictable performance characteristics, the results confirm that the SigmaServer meets the practical requirements of brownfield integration by providing secure OPC UA endpoints with low overhead.

While the performed latency measurements show that the SigmaServer introduces only minimal additional latency, we also need to consider typical deployment scenarios.

In these scenarios the resulting latency still can be significant, yet they primarily depend on the timing of the polling or publish-subscriber mechanisms employed by the clients.

To illustrate this behaviour in more detail, we discuss the forwarding latency that may appear when both the SigmaServer client and the secure client use polling to retrieve data updates. Figure 12 shows a timing diagram of this scenario. For this analysis, we assume a common polling interval T for both participants. The processing time of each participant is neglected, as it is at least an order of magnitude smaller than the polling interval T. Similarly, any internal data sampling performed on the insecure server is ignored, since it occurs regardless of whether the SigmaServer is present. The expected additional forwarding delay for new data arriving via the SigmaServer depends on the time shift t_s between the two polling activities. As shown in Figure 12, this delay lies within the interval $t_d \in [t_t, t_t + T)$, where the lower bound represents the best case and the upper bound (excluded) corresponds to the worst-case delay.

When considering the overall data age, with a client directly polling the insecure server at interval T, this results in a maximum data age of $t_t + T$, which is the sum of the transmission time and the polling interval. Introducing the SigmaServer in between, with an equal polling rate T, leads to a best-case data age of $2 \cdot t_t + T$ and a worst-case data age of $2 \cdot (t_t + T)$.

IX. SECURITY CONSIDERATIONS

SigmaServer is designed with security as a primary concern, in contrast to the majority of existing aggregation and retrofitting solutions identified in our literature review (see Section IV), where security is mostly absent or mentioned as future work.

SigmaServer's core security property stems from the strict separation between the insecure legacy zone and the secure industrial network. No direct network path exists between the two zones: insecure clients (*InsecClients*) write to the thread-safe data store, while secure servers (*SecServers*) only read from it, ensuring full network isolation at the software level.

All communication exposed to the secure zone is encrypted and authenticated using the OPC UA SecurityPolicy *Basic256Sha256*, supporting both certificate-based and username/password authentication. This means that clients operating in the industrial DMZ or on the shop floor are never exposed to insecure legacy protocols, such as Modbus/TCP, which carry no authentication or integrity protection.

With respect to the threat model (Section V-A), the architecture effectively mitigates attacks associated with *attacker levels 1 and 2*. Specifically, external attackers who have gained a foothold in the industrial DMZ cannot perform spoofing, tampering, repudiation, or information disclosure attacks against the legacy zone. The multi-port TCP-level design (TCP-level aggregation) further limits the impact of DoS attacks to individual endpoints, as each legacy device is represented by an independent *SecServer* instance, thereby limiting the impact of potential disruptions.

An additional security benefit arises from the one-to-one device mapping introduced by the TCP-level aggregation concept (*C.3*). By assigning each legacy device a dedicated TCP port and namespace, the architecture avoids *namespace pollution* inherent in application-level aggregating servers (*C.2*), which could potentially introduce ambiguities in access control and complicate the enforcement of per-device security policies.

A. REMAINING VULNERABILITIES

Despite the guarantees described above, SigmaServer cannot fully protect the legacy zone itself from an *attacker level 3* adversary who has gained direct access to the insecure network. As demonstrated in Section VII-B, an adversary connected to the legacy zone can carry out a targeted ARP poisoning attack on the unsecured Modbus/TCP channel between the aggregation server and a legacy device. Using off-the-shelf libraries, the attacker can continuously manipulate the TCP payload of Modbus responses, causing the SigmaServer to forward falsified process values, for instance, stable temperature readings even though the device is overheating.

This limitation is a direct consequence of the inherent insecurity of legacy respectively insecure protocols. Modbus/TCP, for instance, provides no authentication, integrity protection, or replay prevention. Since the proposed solution forwards the data it receives from the insecure network, it is unable to distinguish legitimate from manipulated values on the insecure side (as are all other concepts). Beyond the demonstrated ARP spoofing, the attack surface of insecure protocols is broad: replay attacks, DoS attempts targeting protocol port, and the deployment of rogue Modbus servers are all realistic threat vectors within the legacy zone [50].

The current implementation of SigmaServer focuses exclusively on read operations from legacy devices. Any future extension to support write access and method calls will require careful security analysis, as the forwarding of write requests from the secure zone to insecure legacy devices introduces additional attack vectors, such as the risk of malicious commands propagating into safety-critical process control systems. However, OPC UA already provides a fine-grained access control and with SigmaServer it is even possible to hide non-required nodes completely, which otherwise could pose a security risk.

A further limitation concerns the central data store, which represents a single point of failure in the current implementation. While its thread-safe design ensures consistent access under concurrent load, a crash or corruption would result in all *SecServers* being unable to provide up-to-date values, which would effectively interrupt the entire bridge functionality. Possible mitigations, such as automatic process restart on failure, or a redundant standby instance, are not

yet implemented but represent a possible extension for future work.

B. COMPLEMENTARY SECURITY MECHANISMS

Comprehensive protection of the legacy zone requires complementary security measures that go beyond the scope of SigmaServer itself. In particular, the following measures are identified as necessary extensions:

- **Intrusion Detection Systems (IDSs):** Monitoring the insecure network for anomalous network traffic patterns, e.g., unexpected register writes (Modbus), abnormal polling frequencies, or ARP table inconsistencies, can enable timely detection of level 3 attacks.
- **Honeypot deployment:** The network filter facing the secure zone can redirect suspicious traffic to a sandboxed honeypot [49]. This allows the immediate recording, and analysis of attack behaviour without exposing critical legacy devices.
- **Deep Packet Inspection (DPI):** Network filters with DPI capabilities [51] at the boundary between the insecure and the secure zone can be used to analyse legacy protocol traffic for malicious content, complementing the zone separation enforced by SigmaServer. Moreover, commands (i.e., write access and method calls) can be blocked to prevent manipulation of legacy devices.
- **Layer-2 security:** As discussed in Section IV, solutions based on a modified MACsec protocol can secure Ethernet-based industrial protocols at the ISO/OSI layer 2, partially addressing the absence of link-layer security in legacy networks [35]. However, such mechanisms do not eliminate the use of insecure application-layer protocols within the legacy zone and therefore serve only as a complementary measure.

X. CONCLUSION

In this work, we address the challenge of securely integrating insecure legacy devices into OPC UA-based industrial automation networks. While OPC UA aggregating servers and brownfield retrofitting solutions have been widely discussed in prior work, security aspects are rarely considered. To close this gap, we analysed existing aggregation and retrofitting approaches from a security perspective. Furthermore, we developed a threat model for a typical industrial network setup to identify potential attack vectors. Based on these analyses, three architectural concepts for bridging secure and insecure network zones were derived and evaluated against threat mitigation, namespace integrity, real-time impact, integration effort, and zone compatibility criteria.

We identified TCP-level aggregation as the most suitable concept for the secure integration of legacy systems. It prevents namespace pollution and establishes a clear one-to-one mapping between legacy devices and their secure representations, minimising the integration effort required. Based on this concept, we introduced SigmaServer as a proof-of-principle implementation. Its architecture enforces strict separation between insecure client interfaces and secure OPC UA endpoints by employing a central thread-safe data store and multiple secure servers listening on different ports.

The experimental evaluation on an industrial testbed demonstrated that SigmaServer introduces negligible overhead. End-to-end latency remained below 2.6 ms, with an average internal processing delay of 21.15 μs. Resource utilisation is compared to the performance of the OPC Foundation Console Aggregation Server, which focuses exclusively on OPC UA. In contrast, SigmaServer supports bridging of insecure legacy protocols such as Modbus in addition to OPC UA. The comparison further revealed that SigmaServer consumes less RAM than the OPC Foundation Console Aggregation while maintaining predictable CPU usage. These findings confirm that secure legacy system integration without namespace pollution can be achieved with minimal technical complexity and without compromising performance.

A security analysis demonstrated that SigmaServer effectively mitigates spoofing, tampering, repudiation, and information disclosure attacks at attacker levels 1 and 2. Attacks originating within the legacy zone (level 3) cannot be fully addressed by SigmaServer alone due to the insecurity of legacy protocols, and require additional countermeasures such as IDSs, DPIs or honeypots.

Overall, SigmaServer offers a practical, security-focused solution for bridging insecure legacy devices into modern secure industrial networks. By enabling secure OPC UA communication and efficient runtime performance, it demonstrates that TCP-level aggregation can be effectively realised.

A. FUTURE WORK

While the presented SigmaServer already provides a robust foundation for secure data bridging, several extensions are planned for future work. The protection of the legacy zone itself remains an open challenge. As demonstrated in Section VII-B, the aggregation server cannot prevent attacks that originate within the insecure network. Future work will therefore focus on additional security mechanisms such as Intrusion Detection Systems and the integration of honeypots. These measures aim to detect and analyse malicious activities and to direct suspicious traffic away from critical legacy devices (sandboxing).

Additionally, the current implementation focuses on read access to legacy devices. An extension of SigmaServer can be to support write access and method calls. This includes the secure forwarding of OPC UA write requests and method calls from secure clients to legacy systems.

Finally, future work will further focus on the behaviour discussed in Figure 12, which highlights the forwarding delay introduced by asynchronous polling between insecure and secure interfaces. While the internal processing latency of the SigmaServer is negligible, the effective latency strongly depends on the interaction between polling activities. This latency however could be minimised when

using a publish-subscribe pattern on the secure side and synchronising the publishing activities with the polling activities on the insecure side. Implementing such a synchronisation mechanism would nearly remove the remaining performance-related drawbacks of SigmaServer.

ACKNOWLEDGMENT

The authors would like to thank Barracuda Networks for providing complimentary firewall licenses, which supported the experimental infrastructure used in this research.

REFERENCES

[1] B. Briggs. (2019). *Hackers Hit Norsk Hydro With Ransomware. The Company Responded With Transparency*. [Online]. Available: https://news.microsoft.com/transform/hackers-hit-norsk-hydro-ransomware-company-responded-transparency/

[2] T. Miller, A. Staves, S. Maesschalck, M. Sturdee, and B. Green, "Looking back to look forward: Lessons learnt from cyber-attacks on industrial control systems," *Int. J. Crit. Infrastruct. Protection*, vol. 35, Dec. 2021, Art. no. 100464.

[3] *Industrial Communication Networks-Network and System Security-Part 3-2: System Security Requirements and Security Assurance Levels*, Standard IEC 62443-3-2, 2020.

[4] National Institute of Standards and Technology, *Guide to Operational Technology (OT) Security*, Standard NIST SP 800-82, Gaithersburg, MD, USA, 2023.

[5] M. Graham, C. Ahlers, and K. O'Meara. (Jul. 2024). *Impact of FrostyGoop ICS Malware on Connected OT Systems*. [Online]. Available: https://hub.dragos.com/hubfs/Reports/Dragos-FrostyGoop-ICS-Malware-Intel-Brief-0724r2.pdf

[6] T. J. Williams, *A Reference Model for Computer Integrated Manufacturing (CIM): A Description From the Viewpoint of Industrial Automation*, Research Triangle Park, North Carolina, USA: Instrum. Soc. Amer., 1989.

[7] E. D. Knapp and J. T. Langill, *Industrial Network Security: Securing Critical Infrastructure Networks for Smart Grid, SCADA, and Other Industrial Control Systems*, Massachusetts, United States: Syngress, 2014.

[8] C. Urrea, C. Morales, and J. Kern, "Implementation of error detection and correction in the modbus-RTU serial protocol," *Int. J. Crit. Infrastructure Protection*, vol. 15, pp. 27–37, Dec. 2016.

[9] *Industrial Communication Networks-Fieldbus Specifications—Part 6: Application Layer Service Specification, International Electrotechnical Commission Std. IEC 61158-6, 2023.*, Standard IEC 61158-6, 2023.

[10] H. Lasi, P. Fettke, H.-G. Kemper, T. Feld, and M. Hoffmann, "Industry 4.0," *Bus. & Inf. Syst. Eng.*, vol. 6, no. 4, pp. 239–242, Aug. 2014.

[11] M. Schleipen, S.-S. Gilani, T. Bischoff, and J. Pfrommer, "OPC UA & industrie 4.0—Enabling technology with high diversity and variability," *Proc. CIRP*, vol. 57, pp. 315–320, Jan. 2016.

[12] M. Hankel and B. Rexroth. (Apr. 2015). *The Reference Architectural Model Industrie 4.0 (RAMI 4.0)*. [Online]. Available: https://www.zvei.org/fileadmin/user upload/PresseundMedien/Publikationen/2015/april/DasReferenzarchitekturmodellIndustrie4.0RAMI4.0/ZVEI-Industrie-40-RAMI-40-English.pdf

[13] (Apr. 2015). *Umsetzungsstrategie Industrie 4.0*. [Online]. Available: https://www.plattform-i40.de/IP/Redaktion/DE/Downloads/Publikation/umsetzungsstrategie-2015.html

[14] (Mar. 2018). *Welche Kriterien Müssen Industrie-4.0-Produkte Erfüllen*. [Online]. Available: https://www.zvei.org/fileadmin/user upload/PresseundMedien/Publikationen/2018/Maerz/ZVEIWelcheKriterienmuessenI-4.0-Produkteerfuellen2018/ZVEIWelcheKriterienmuessenI-4.0-Produkteerfuellen2018.pdf

[15] *OPC 10000-1 Unified Architecture Part 1: Overview and Concepts*, Scottsdale, AZ, US, OPC, 2022.

[16] R. S. Crespí, A. Armentia, I. Sarachaga, O. Casquero, F. Pérez, and M. Marcos, "OPC UA aggregation server in the fog," in *Proc. 24th IEEE Int. Conf. Emerg. Technol. Factory Autom. (ETFA)*, Sep. 2019, pp. 1256–1260.

[17] D. Großmann, M. Bregulla, S. Banerjee, D. Schulz, and R. Braun, "OPC UA server aggregation—The foundation for an Internet of Portals," in *Proc. IEEE Emerg. Technol. Factory Autom. (ETFA)*, Sep. 2014, pp. 1–6, doi: 10.1109/ETFA.2014.7005354.

[18] M. Schleipen, *Praxishandbuch OPC UA: Grundlagen, Implementierung, Nachrüstung, Praxisbeispiele*, Würzburg, Germany: Vogel Communications Group, 2018.

[19] C. Wohlin, "Guidelines for snowballing in systematic literature studies and a replication in software engineering," in *Proc. 18th Int. Conf. Eval. Assessment Softw. Eng.*, May 2014, pp. 1–10.

[20] A. Fernbach, W. Kästner, S. Mätzler, and M. Wollschlaeger, "An OPC UA information model for cross-domain vertical integration in automation systems," in *Proc. IEEE Emerg. Technol. Factory Autom.*, Sep. 2014, pp. 1–8.

[21] I. Seilonen, T. Tuovinen, J. Elovaara, I. Tuomi, and T. Oksanen, "Aggregating OPC UA servers for monitoring manufacturing systems and mobile work machines," in *Proc. IEEE Int. Conf. Emerg. Technol. Factory Autom.*, Sep. 2016, pp. 1–4.

[22] S. Banerjee and D. Grossmann, "Aggregation of information models—An OPC UA based approach to a holistic model of models," in *Proc. 4th Int. Conf. Ind. Eng. Appl. (ICIEA)*, Apr. 2017, pp. 296–299.

[23] M. Graube, S. Hensel, C. Iatrou, and L. Urbas, "Information models in OPC UA and their advantages and disadvantages," in *Proc. 22nd IEEE Int. Conf. Emerg. Technol. Factory Autom. (ETFA)*, Sep. 2017, pp. 1–8.

[24] H. Wang, Y. Ma, and F. Yu, "An OPC UA multi-server aggregator with cache management," in *Proc. Chin. Autom. Congr. (CAC)*, Nov. 2018, pp. 68–73.

[25] A. Würger, K.-H. Niemann, and A. Fay, "Concept for an energy data aggregation layer for production sites a combination of AutomationML and OPC UA," in *Proc. IEEE 23rd Int. Conf. Emerg. Technol. Factory Autom. (ETFA)*, vol. 1, Sep. 2018, pp. 1051–1055.

[26] D. A. Breunig and M. Schneider, "Multi-protocol data aggregation and acquisition for distributed control systems," *Proc. CIRP*, vol. 81, pp. 310–315, Jun. 2019.

[27] R. R. Peña, R. D. Fernández, M. Lorenc, and A. Cadiboni, "Gateway OPC UA/modbus applied to an energy recovery system identification," in *Proc. Workshop Inf. Process. Control (RPIC)*, Sep. 2019, pp. 235–240.

[28] Y. Li, J. Jiang, C. Lee, and S. H. Hong, "Practical implementation of an OPC UA TSN communication architecture for a manufacturing system," *IEEE Access*, vol. 8, pp. 200100–200111, 2020.

[29] S. G. Mathias, S. Schmied, and D. Grossmann, "An investigation on database connections in OPC UA applications," *Proc. Comput. Sci.*, vol. 170, pp. 602–609, Mar. 2020.

[30] J. N. Weskamp and J. Tikekar, "An industrie 4.0 compliant and self-managing OPC UA aggregation server," in *Proc. 26th IEEE Int. Conf.*, Sep. 2021, pp. 1–8.

[31] C. Pu, X. Ding, P. Wang, and Y. Yang, "Practical implementation of an OPC UA multi-server aggregation and management architecture for IIoT," in *Proc. IEEE Int. Conf. Internet Things (iThings) IEEE Green Comput. Commun. (GreenCom) IEEE Cyber, Phys. Social Comput. (CPSCom) IEEE Smart Data (SmartData) IEEE Congr. Cybermatics (Cybermatics)*, Aug. 2022, pp. 476–481.

[32] A. Busboom, "Automated generation of OPC UA information models—A review and outlook," *J. Ind. Inf. Integr.*, vol. 39, May 2024, Art. no. 100602.

[33] R. Knobloch, C. Plesker, A. Reuther, T. Dasbach, and B. Schleich, "OPC UA aggregation servers–enabling vertical integration of production machines for digital manufacturing," *Digit. Eng.*, vol. 8, Mar. 2026, Art. no. 100080.

[34] D. Etz, H. Brantner, and W. Kastner, "Smart manufacturing retrofit for brownfield systems," *Proc. Manuf.*, vol. 42, pp. 327–332, Jan. 2020.

[35] T. Lackorzynski, G. Garten, J. S. Huster, S. Köpsell, and H. Härtig, "Enabling and optimizing MACsec for industrial environments (extended abstract)," in *Proc. 16th IEEE Int. Conf. Factory Commun. Syst. (WFCS)*, vol. 8282, Apr. 2020, pp. 1–4.

[36] B. Rupprecht, E. Trunzer, S. König, and B. Vogel-Heuser, "Concepts for retrofitting industrial programmable logic controllers for industrie 4.0 scenarios," in *Proc. 22nd IEEE Int. Conf. Ind. Technol. (ICIT)*, vol. 1, Mar. 2021, pp. 1034–1041.

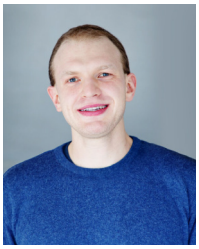
[37] M. Majumder, M. Shakil, and A. Zoitl, "Evaluation of PLC4X based middleware as integrator of brownfield systems into industrial cyber-physical systems," *IFAC-PapersOnLine*, vol. 55, no. 2, pp. 427–432, Jan. 2022.

[38] T.-A. Tran, T. Ruppert, G. Eigner, and J. Abonyi, "Retrofitting-based development of brownfield industry 4.0 and industry 5.0 solutions," *IEEE Access*, vol. 10, pp. 64348–64374, 2022.

[39] R. Sujith, V. S. Ganapathy, R. Ilayaraja, K. Durgadevi, N. Balaji, and A. Vignesh, "Seamless integration of legacy industrial systems with OPC UA for enhanced digital transformation," in *Proc. Int. Conf. Power, Energy, Control Transmiss. Syst. (ICPECTS)*, Oct. 2024, pp. 1–6.
[40] N. Holmes, L. Katavich, and X. Xu, "Retrofitting legacy systems for industry 4.0 via OPC UA and distributed control," *Manuf. Lett.*, vol. 44, pp. 1337–1348, Aug. 2025.
[41] OPC Foundation. (2021). *OPC 30270-OPC UA for Asset Administration Shell (Industry 4.0 AAS)*. [Online]. Available: https://reference.opcfoundation.org/I4AAS/v100/docs/
[42] *GitHub-OPCFoundation/UA-.NETStandard-Samples*, Accessed: Sep. 01, 2025. [Online]. Available: https://github.com/OPCFoundation/UA-.NETStandard-Samples
[43] *UaGateway-Unified Automation*, Accessed: Sep. 10, 2025. [Online]. Available: https://www.unified-automation.com/products/ua-runtime-software/uagateway.html
[44] *Sichere Unternehmensübergreifende Kommunikation Mit OPC UA*, Bundesministerium Für Wirtschaft und Energie (BMWi), Berlin, Germany : Bundesministerium für Wirtschaft und Energie (BMWi), 2019, p. 28.
[45] S. D. D. Anton, D. Fraunholz, D. Krohmer, D. Reti, D. Schneider, and H. D. Schotten, "The global state of security in industrial control systems: An empirical analysis of vulnerabilities around the world," *IEEE Internet Things J.*, vol. 8, no. 24, pp. 17525–17540, Dec. 2021.
[46] Fortinet. (2024). *Fortinet Report: Threat Actors Increasingly Targeting Ot Organizations*. [Online]. Available: https://www.fortinet.com/corporate/about-us/newsroom/press-releases/2024/fortinet-report-threat-actors-increasingly-targeting-ot-organizations
[47] P. Mueller and B. Yadegari. (2012). *The Stuxnet Worm*. [Online]. Available: https://www2.cs.arizona.edu/{~}collberg/Teaching/466-566/2012/Resources/presentations/topic9-final/report.pdf
[48] C. Chiculita, E. Liviu, and M. L. Cristea, "Towards multi-port modbus gateway," in *Proc. 4th Int. Symp. Electr. Electron. Eng. (ISEEE)*, Oct. 2013, pp. 1–4.
[49] O. Saßnick, G. Schäfer, T. Rosenstatter, and S. Huber, "A generative model based honeypot for industrial OPC UA communication," in *Proc. Comput. Aided Systems Theory–EUROCAST*, Apr. 2024, pp. 320–334.
[50] M. R. U. Rashid, Y. Singh, and S. I. Manzoor, "Methodology for assessing existing vulnerabilities in modbus protocol," in *Proc. Comput. Sci.*, vol. 259, 2025, pp. 1983–1993.
[51] O. N. Nyasore, P. Zavarsky, B. Swar, R. Naiyeju, and S. Dabra, "Deep packet inspection in industrial automation control system to mitigate attacks exploiting modbus/TCP vulnerabilities," in *Proc. IEEE IEEE 6th Int. Conf. Big Data Secur. Cloud (BigDataSecurity) Int. Conf. High Perform. Smart Comput., (HPSC) IEEE Int. Conf. Intell. Data Secur. (IDS)*, May 2020, pp. 241–245.

DALIBOR SAIN received the B.Sc. degree in information technology and systems management from Salzburg University of Applied Sciences, Austria, in 2024. He is currently pursuing the M.Sc. degree in cybersecurity with the Josef Ressel Centre for Intelligent and Secure Industrial Automation. He is a Junior Researcher with the Josef Ressel Centre for Intelligent and Secure Industrial Automation, focusing on OT-security and the secure integration of legacy devices in Industry 4.0.

THOMAS ROSENSTATTER received the B.Sc. degree in information technology and systems management from Salzburg University of Applied Sciences, Austria, in 2014, the M.Sc. degree in embedded and intelligent systems from Halmstad University, Sweden, in 2016, and the Ph.D. degree in computer science and engineering with a focus on automotive cybersecurity from the Chalmers University of Technology, Sweden, in 2021. He was a Researcher with the RISE Research Institutes of Sweden, concentrating on cybersecurity in mobility solutions. He is currently a Senior Lecturer and a Researcher with the Josef Ressel Centre for Intelligent and Secure Industrial Automation, Salzburg University of Applied Sciences. His research interests include cybersecurity and resilience in cyber-physical systems, with a focus on the automotive and industrial domains.



OLAF SAßNICK received the master's degree, in 2015. He was an Embedded Software Engineer in the industry, until 2020, when he joined Salzburg University of Applied Sciences. He is currently a Researcher with the Josef Ressel Centre for Intelligent and Secure Industrial Automation, focusing on cybersecurity, and a Lecturer with Salzburg University of Applied Sciences, with a focus on sensors, automation, and embedded systems.

CHRISTIAN SCHÄFER received the B.Sc. and M.Sc. degrees from the University of Applied Sciences Salzburg, Austria, in 2021 and 2023, respectively, where he researched toward improving security frameworks for industrial control systems. He is currently a Software Engineer with B&R Industrial Automation GmbH, focusing on embedded Linux and real-time systems, with particular interests in containerization, real-time optimization, and cybersecurity.

STEFAN HUBER received the B.Sc. degree in computer science, the B.Sc. degree in mathematics, the M.Sc. degree in computer science, the M.Sc. degree in mathematics, and the Ph.D. degree in computer science from the University of Salzburg, Austria, in 2006, 2007, 2008, 2009, and 2011, respectively.

He was a Postdoctoral Researcher with IST Austria, from 2013 to 2015, a Postdoctoral Researcher with the University of Salzburg, in 2012, and the Team and Project Leader of B&R Industrial Automation GmbH, from 2015 to 2019. He is currently a Professor with Salzburg University of Applied Sciences, Austria, the Head of Research with the Department of Information Technologies and Digitalisation, and the Head of the Josef Ressel Center for Intelligent and Secure Industrial Automation. His primary research interests include algorithms, computational geometry and topology, machine learning, and industrial automation.