IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 7 April 2023, accepted 8 May 2023, date of publication 11 May 2023, date of current version 17 May 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3275443

RESEARCH ARTICLE

Protecting Modbus/TCP-Based Industrial Automation and Control Systems Using Message Authentication Codes

FILIP KATULIĆ, DAMIR SUMINA, STJEPAN GROŠ, AND IGOR ERCEG, (Member, IEEE)
Faculty of Electrical Engineering and Computing, University of Zagreb, 10000 Zagreb, Croatia

Corresponding author: Filip Katulić (filip.katulic@fer.hr)

This work was supported by the European Regional Development Fund under the Project "An Innovative Solution for Cyber Security Management of Industrial System of Facility and Process Automation" under Grant KK.01.2.1.02.0009.

ABSTRACT Critical infrastructure (CI), such as energy and water distribution systems, is essential for the stability and well-being of the modern society. Industrial automation and control systems (IACSs) form the backbone of CIs and enable the operation of such systems in a safe and reliable manner. However, with the increasing use of industrial Ethernet communication protocols, such as Modbus-over-TCP (Modbus/TCP), once air-gapped IACSs are becoming vulnerable to potential cybersecurity threats. This paper presents a novel method for enhancing the cybersecurity of Modbus/TCP-based IACSs by implementing an authentication method based on message authentication codes (MACs). To provide partial protection of communication even when communicating with legacy Modbus/TCP peers, we propose a novel supervising device that analyzes exchanged messages and verifies the authenticity of the protected messages. To experimentally verify the protection method, a water-treatment cyber-physical system (CPS) was implemented as a digital twin in a programmable logic controller (PLC). The underlying MAC is the Chaskey-12, lightweight MAC defined in IEC 29192-6. It was implemented in the PLC program using the programming languages defined in IEC 61131-3. As an additional contribution, the presented implementation allows protection of communication between PLCs and other Modbus/TCP peers installed in existing IACSs without hardware or firmware modifications. The results show that the method provides protection against network attacks without significantly affecting performance, also demonstrating the feasibility of such protection in IACSs.

INDEX TERMS Automation, communication system security, cyber-physical systems, industrial communication.

I. INTRODUCTION

The term *industrial cybersecurity* encompasses knowledge and a set of activities performed with the aim of mitigating cybersecurity risks within industrial automation and control systems (IACSs) [1]. IACSs, such as supervisory control and data acquisition (SCADA) and programmable logic controllers (PLCs), are the main components of critical infrastructures (CIs), which are classified by the European Programme for Critical Infrastructure Protection (EPCIP) into two sectors: Energy and Transport [2]. Given the current global energy crisis and the COVID-19 pandemic, even a small disruption in energy or transport infrastructures could have a serious impact on the stability of modern societies [3]. Still, insecure and outdated industrial communication protocols, such as Modbus-over-TCP (Modbus/TCP) [4] are widely used for their efficiency and ease-of-use, which are standard requirements in IACSs. With the enlargement and distribution of such systems, they become increasingly interconnected with information technology (IT) systems, introducing cybersecurity vulnerabilities and attack vectors that were not analyzed at the time of their deployment [5]. As a result, once isolated IACS communication networks are much more vulnerable to cyberattacks than IT networks.

The associate editor coordinating the review of this manuscript and approving it for publication was Diana Gratiela Berbecaru.

Although one of the oldest recorded incidents related to IACS cybersecurity dates back to the late 1990s [6], it was not until the 2010 Natanz nuclear power plant cyberattack (also known as Stuxnet) that IACS cybersecurity came under the spotlight [7]. Stuxnet was a malware developed for the sole purpose of undermining Iran's nuclear development program. The complexity of the operation suggests that it was created by an advanced persistent threat (APT) that managed to destroy one-fifth of the uranium enrichment centrifuges. Next, the 2015 Ukraine blackout attack was the first attack recorded to target the power grid CI. It resulted in a power outage and affected more than 200 thousand people [8]. The 2022 Russo-Ukrainian conflict is the latest example of the importance of IACS cybersecurity, in which the state-sponsored APTs have developed complex IACS-specific malware which are capable of disrupting and degrading the power grid CI [9], [10].

When compared with cybersecurity requirements in IT systems, where confidentiality is the number one priority, IACSs mainly focused on availability, authenticity, and integrity [11], [12]. Additionally, the computational limitations of IACS devices led to the fact that most of cybersecurity requirements were sacrificed to satisfy the availability requirement. With that in mind, and the fact that most of the industrial Ethernet (IE) communication protocols lack the authenticity and integrity checks, it has become imperative to come up with a mechanism that can ensure integrity and authenticity of communication, with minimal overhead costs and latency impact. Without integrity and authenticity, protocols such as Modbus/TCP are vulnerable to Man-in-the-Middle (MITM) attacks [13].

This study deals with a potential cybersecurity enhancement of Modbus/TCP-based IACSs that satisfies the integrity and authenticity cybersecurity requirements, without significantly affecting the system availability. The novel protection method uses message authentication codes (MACs) as an underlying protection mechanism. Additionally, the developed protection method supports communication with legacy Modbus/TCP devices which do not support the developed protection, allowing integration of the method in existing IACSs. After the development and cybersecurity analysis of the protection method, a water treatment cyber-physical system (CPS) was created to provide an environment for experimental validation. The protection method was successfully implemented in the PLC program using IEC 61131-3 programming languages, allowing protection of legacy IACSs without drastically changing the system's network architecture or introducing bump-in-the-wire (BITW) components that could potentially introduce new threat vectors. Chaskey-12, a lightweight MAC defined by the IEC 29192-6 standard, was chosen as the underlying MAC algorithm. To our knowledge, this is the first time that Modbus/TCP has been protected using lightweight cryptographic algorithms. Furthermore, the contribution of this paper includes a proposal of a supervising device capable of verifying the authenticity of protected messages even when communicating with legacy devices. This protection method could serve as a transitional cybersecurity solution for existing IACSs that use insecure industrial communication protocols and where the transition to new, more secure protocols is not justified by the cost-benefit analysis. Finally, the experimental validation showed that the communication authenticity was preserved, without significant impact on the performance.

The paper is organized as follows. In Section II, the related work is presented. Section III gives a cybersecurity analysis of the Modbus/TCP communication protocol. Section IV gives detailed information about the protection method. In Section V, details about the implemented water treatment CPS are given, after which the protection method implementation is analyzed in terms of cybersecurity and performance. Finally, Section VI gives the discussion, and Section VII presents the conclusions and future work.

II. RELATED WORK

Given the significant and continuous increase in cybersecurity incidents and threats [3], both in number and complexity, researchers have extensively analyzed the Modbus/TCP cybersecurity. One of the first in-depth security analyses of Modbus/TCP dates back to 2008, which resulted in the attack taxonomies for Modbus protocols [14]. Additionally, in [15], the authors confirmed the conclusions concerning the insecurity of Modbus/TCP.

With the analysis of the Modbus/TCP cybersecurity, possible cryptography-based solutions emerged, such as [16] in which the authors implemented RSA digital signatures for the authentication and a selected SHA-2 hash function for the integrity protection of Modbus/TCP. The authors also emphasized that confidentiality is not a requirement for IACSs. The authors of [17] suggested the TLS communication protocol, currently used in IT networks, to be used on top of Modbus/TCP, and achieved a significant improvement in cybersecurity, however with substantial latency and overhead costs. Next, an interesting method for authenticating Modbus/TCP devices named ModbusSec was presented in [18], where authentication of communication was established by using hash-based MAC codes and the Stream Control Transmission Protocol. They also proposed the use of standard hash-based MAC codes without even considering the use of lightweight cryptography. In addition, the authors of [19] analyzed potential methods for protecting Modbus/TCP, but stressed that asymmetric cryptographic methods are significantly slower than the symmetric ones, questioning the use of asymmetric cryptography in IACSs. In [20], the authors presented a BITW authentication method that does not modify Modbus/TCP but transmits the cryptographic authentication information in the TCP *Options* header field. It should be noted that the format and content of the TCP *Options* header is strictly defined by the Internet Assigned Numbers Authority (IANA) and should generally not be used for purposes other than those defined. Similarly, the authors of [21] developed a security module that should be installed

on every Modbus/TCP node and could use different cryptographic methods based on the cybersecurity requirements of an IACS. Although BITW devices could protect legacy equipment within an IACS, such solutions could compromise the reliability of the system, introduce new vulnerabilities, and affect real-time operation, which is not acceptable [1]. As a result of an intensive security analysis of the Modbus/TCP communication protocol, the Modbus Organization developed a new industrial communication protocol called Modbus Security [22]. Modbus Security is a TLS-based Modbus/TCP that provides integrity, authentication, and encryption of data, but is not backward compatible with Modbus/TCP. Still, to our knowledge, it is yet to be implemented in a real-life IACS.

Intrusion detection systems (IDSs) were also analyzed as potential solutions to increase the IACS resiliency against cyberattacks [23], [24]. With the modification of the Modbus/TCP packet structure, such as encrypting the Modbus/TCP traffic, the datasets used for IDSs could become unusable, as is the case with the TLS protocol [25].

A different approach to the protection of Modbus/TCP is given in [26], in which the authors presented a method called TAMBUS that uses covert channels as an authentication path. Although the method does not modify the original Modbus/TCP structure, it requires additional equipment that embeds and verifies the TAMBUS code.

With the development of the IEC 29192-6 [27] standard, which describes lightweight cryptographic methods for data integrity and authentication of resource-constrained devices, new opportunities for IACS devices emerge. Lightweight MAC algorithms, such as Chaskey-12 could provide sufficient protection for IACSs that must meet strict timing requirements, as they provide the same for the Internet of Things (IoT) devices [28]. In [29], the authors compared the performance of the lightweight Chaskey MAC algorithm with that of AES-128-CMAC, in which Chaskey showed significant performance improvements. Tests were also performed on ARM Cortex-M microcontrollers, which are commonly used in PLCs.

Finally, some authors [30], [31], implemented lightweight cryptographic algorithms within the PLC's program and showed that the real-time performance of IACSs was preserved. To our knowledge, there are no studies that used the implemented algorithms to protect industrial communication protocols.

As can be seen from the literature review, there are several drawbacks we identified: (*i*) the authors did not consider the impact of cybersecurity solutions on availability [16], [17], [22] and did not define which of the cybersecurity requirements are actually required in IACSs, (*ii*) the protection methods were proposed without defining a specific threat model which would define situations in which the protection method is feasible [17], [20], [21], and (*iii*) backward compatibility with existing systems was preserved only by using BITW devices [16], [20], [21] or there was no backward compatibility at all [22].

III. CYBERSECURITY OF THE MODBUS/TCP COMMUNICATION PROTOCOL

Modbus/TCP is an industrial communication protocol developed by Modicon in 1999. It is widely used in IACSs because of its ease-of-implementation and the fact that it is open and licence-free [4]. The protocol is based on a proprietary Modbus-over-Serial-Line protocol data unit (PDU) that is encapsulated within TCP and can be used on Ethernet-based networks. Modbus/TCP clients, such as SCADAs, PLCs, and human machine interfaces (HMIs) initiate communication with Modbus/TCP servers (PLCs, remote terminal units (RTUs), other field devices), which then respond to an appropriate request (e.g., read or write process data) based on the function code value. The structure of a *Modbus/TCP PDU* is shown in Fig. 1. Modbus/TCP uses the Ethernet network for the data link layer, Internet Protocol (IP) for the network layer, TCP as the transport layer protocol, and the Modbus application protocol in the upper layers of the Open Systems Interconnection (OSI) network model. The Modbus/TCP application protocol also has its own header which is called the Modbus application header (MBAP). Finally, the function code and the Modbus data are embedded with the MBAP header to form the *Modbus/TCP application data unit* (ADU). In this paper, all the data starting with Ethernet header and ending with Frame Check Sequence (FCS) will be referred to as *Modbus/TCP PDU*, which differs from the term *Modbus PDU* used in the specifications [4].

As can be seen, the Modbus/TCP application protocol lacks any cybersecurity protection mechanisms. Further, the use of common Internet protocols, such as TCP and IP, implies the import of additional attack vectors that an adversary can exploit [14].

Based on [24], the main Modbus/TCP threat categories are (*i*) denial of service (DoS) attacks in which an adversary can impact the performance and availability of communication, (*ii*) information acquiring in which the attacker can collect information about IACSs (e.g., network architecture, application relations, equipment used and data within the system), and finally, (*iii*) spoofing attacks, which come as a direct consequence of the lack of authenticity and integrity checking mechanisms.

IV. PROTECTION METHOD

In this section we define a Modbus/TCP-based IACS model and an adversarial model. After defining the models, the developed protection method is presented, along with a cybersecurity analysis of the proposed method.

A. SYSTEM MODEL

While developing a Modbus/TCP-based IACS model, which is shown in Fig. 2, we followed the rules of the Purdue Enterprise Reference Architecture (PERA) [1]. The Purdue model is also used in the ISA/IEC 62443 industrial cybersecurity standard as a reference for the IACS network architecture [32]. In the Purdue model, the IACS network is divided into several levels, with the lowest level 0 (also referred

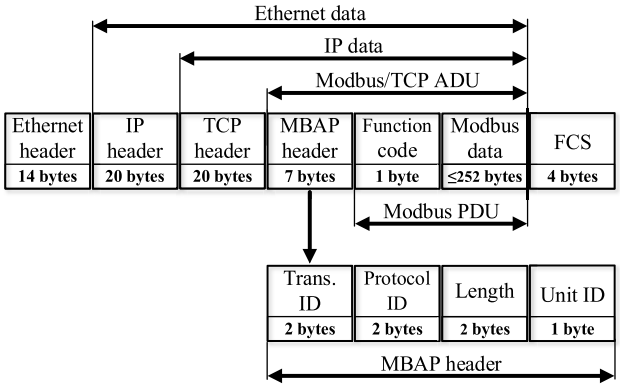

FIGURE 1. Structure of Modbus/TCP PDU.

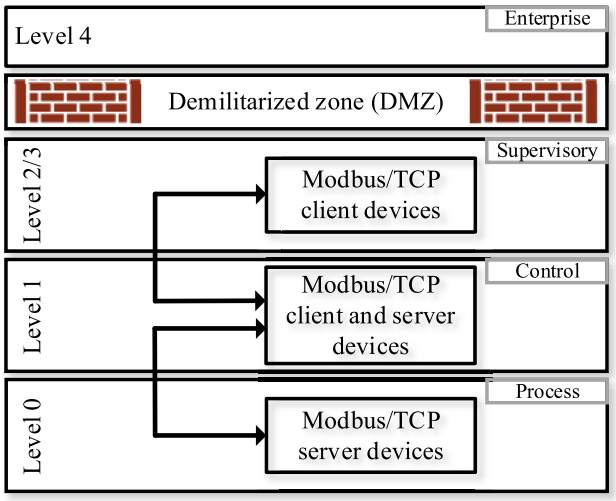

FIGURE 2. Modbus/TCP-based system model.

to as the process level) representing the physical process, while level 4 is the enterprise IT system. Within the model, IT and operation technology (OT) networks are separated by a demilitarized zone (DMZ), a zone designed to prevent direct communication between the given communication networks.

At the process level, it is common to find devices such as sensors and actuators, which serve as a direct interface to an industrial process. In terms of communication, process level devices are usually Modbus/TCP servers that respond to requests from clients located in the control level. The control level contains devices that control the automated industrial process and transfer process-related information to the supervisory level. An example of such a device is a PLC, which can be a Modbus/TCP client to the process devices, but also a Modbus/TCP server to the supervisory level systems. Finally, supervisory level devices allow human operators to supervise and partially control the industrial process. The process supervision is maintained by the client-server application relation to the control level. In addition, the supervisory level devices communicate with the enterprise devices through services located in the DMZ, but they generally do not use Modbus/TCP for communication [1].

The IACS levels are interconnected via Ethernet cables and switches, routers and/or firewalls located at the level boundaries. Based on that, our system model covers the four lower levels of the Purdue model. Also, we assume that the IACS is not distributed, meaning that all system components are located at one location site. The IACS is owned and maintained by a single organization, allowing full control of the system.

B. THREAT MODEL

Without a detailed threat model, it is not possible to devise appropriate protection against security incidents. The threat model specifies possible adversaries and threats which might impact the security of a system. The IEC 62443 categorizes adversaries according to their capabilities, the amount of resources available, and their motivation [32]. They range from hacktivists to highly motivated APTs and are modelled depending on the cyber threat environment.

Our threat model assumes an attacker that has gained unauthorized initial access to the industrial communications network. The initial access to the IACS could be gained by compromising IT devices within the OT network (e.g., engineering workstations, printers), implementing a "rouge" device within the industrial communication network, replicating via removable media, and compromising the supply chain (e.g., firmware updates, physical devices) [33]. Very importantly, our threat model assumes that Modbus/TCP peers are not compromised and that unauthorized access to the IACS devices is not possible (e.g., through physical protection measures such as locking devices in cabinets). Furthermore, we assume that the attacker has extensive knowledge of the IACS (e.g., used equipment, network architecture, hardware and firmware versions, understanding of the automation process), but does not have valid credentials to access the IACS devices.

Fig. 3 shows the threat model. After the initial compromise, the attacker takes control of the communication between the Modbus/TCP peers, which is similar to the threat model defined in [34]. By controlling the network, the attacker can perform passive and active attacks on the Modbus/TCP client-server communications, meaning that an adversary can read, inject, remove, or tamper with PDUs passing through the network [34].

Based on the chosen threat model, the attacker can violate all key cybersecurity requirements: confidentiality, data origin authentication, availability, and nonrepudiation [34].

However, depending on the results of the cybersecurity risk analysis, the violation of the given requirements may not always pose a risk to the organization managing the IACS. For example, enforcing confidentiality of the Modbus/TCP PDUs would imply that the messages contain secret information, which is not common in IACSs [35]. There are only a limited number of IACSs that require data confidentiality, such as:

- IACSs in the defense sector, where a disclosure of, e.g., energy consumption within a military IACS could

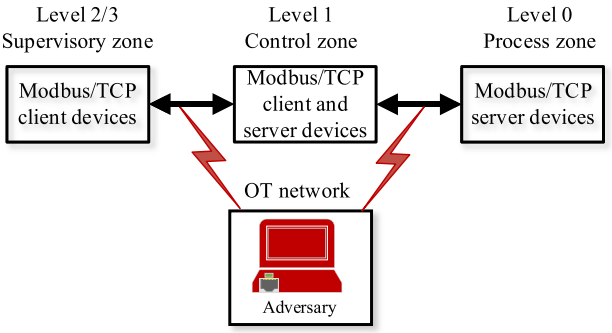

FIGURE 3. Threat model.

provide information to APT adversaries about possible military actions.

- Process control systems (PCSs), where disclosure of trademark recipes could seriously affect the IACS operations (industrial espionage).
- Other IACSs, such as the Smart Grid, that comply with data protection laws (e.g., the General Data Protection Regulation (GDPR)).

The same is true for nonrepudiation, which is defined as an assurance that peers cannot deny any performed actions [32]. This requirement allows a third party to determine whether or not peers have performed a particular action. Within an IACS managed and owned by one organization, such a requirement can theoretically be omitted, since a trusted third party could analyze the network logs to determine whether a peer has performed a disputed action.

Although the protocol itself does not need to provide nonrepudiation, the organization must prevent any modifications to captured network logs (integrity) and preserve them for an adequate amount of time. Network logging is already well established in IT and could be similarly implemented in OT networks [35]. On the other hand, an IACS managed by multiple organizations should always enforce nonrepudiation, which could play a critical role in determining responsibility in the event of an IACS failure. Availability, being the most important requirement in industrial communication systems [1], should always be preserved.

As for the data origin authentication, it allows the peers to determine the identity of communication partners and preserve the integrity of transmitted PDUs [32]. The authors of all previously mentioned papers stress that the attacks targeting the integrity of Modbus/TCP messages (e.g., modification, insertion) could cause significant damage to the IACSs. Due to the above-mentioned reasons, data origin authentication is essential to prevent active attacks on IACS communications, but should be developed with the availability requirement in mind. Based on that, in the next subsection we will present the protection method which ensures data origin authentication without any significant impact on the availability.

C. PROPOSED PROTECTION METHOD

Based on the presented system and threat models, we propose a MAC-based Modbus/TCP protection method. MACs are symmetric key cryptographic mechanisms used to authenticate messages exchanged between devices. They are commonly used in IT systems and have proven to be an effective method for ensuring the integrity and authentication of communication [35]. There are several standards that define methods for calculating MACs, the most known being the IEC 9797 series of standards. Additionally, MACs designed for resource-constrained devices are defined in the IEC 29192-6 lightweight cryptography standard. Lightweight cryptography mechanisms consume less computational and energy resources compared to standard cryptography and are therefore suitable for IACS devices. The proposed protection method could use any MAC algorithm whose performance satisfies IACS-specific requirements, such as resource constraints and low communication time delay.

The proposed protection method, shown in Fig. 4 consists of three main components, the *Peer authentication and key agreement* service, the *Modbus/TCP ADU MAC protection* service, and the *Data origin authentication* service. When a Modbus/TCP client initiates a communication request to the Modbus/TCP server, the *Peer authentication and key agreement* service establishes a session key and PDU counter on both peers. In addition, the given service takes care of potential issues regarding counter resets and overflows by establishing new session keys, while resetting the PDU counter. This study does not deal with the implementation of the *Peer authentication and key agreement* service, but assumes that the peers have authenticated themselves, and that both the session key and PDU counter values have been established.

After the establishment of the session key and the PDU counter, the Modbus/TCP client application sends its original ADU to the *Modbus/TCP ADU MAC protection* service, which protects the ADU as shown in Fig. 5. The service calculates a MAC using the original Modbus/TCP ADU, plus the additional PDU counter field.

After the MAC calculation, the MAC and PDU counter are concatenated to the Modbus data field and sent to the server. It should be noted that both the MAC and the PDU counter field lengths are implementation-defined, and their length depends on system requirements (e.g., Modbus data overhead, cybersecurity requirements) and the underlying MAC algorithm.

When the server receives a Modbus/TCP request, it is first processed by the *Data origin authentication* service, whose flow chart is given in Fig. 6. The service allows communication parties to determine whether the received Modbus/TCP PDU is authentic or if it has been tampered with. Additionally, it assumes the session key and counter from the *Peer authentication and key agreement* service. It should also be noted that the protected peers must be fed with a list of legacy peers that do not support the protection method, while assuming that all other peers are protected.

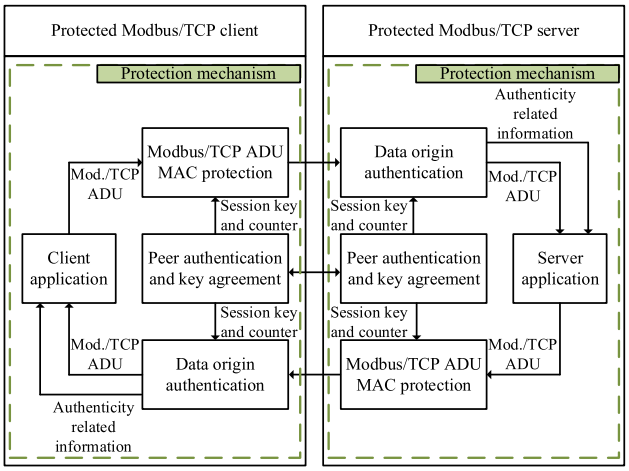

FIGURE 4. Flowchart of developed protection method.

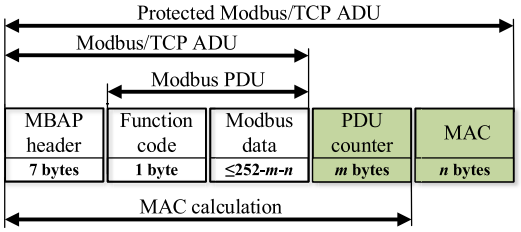

FIGURE 5. Modbus/TCP packet structure after performing MAC protection algorithm.

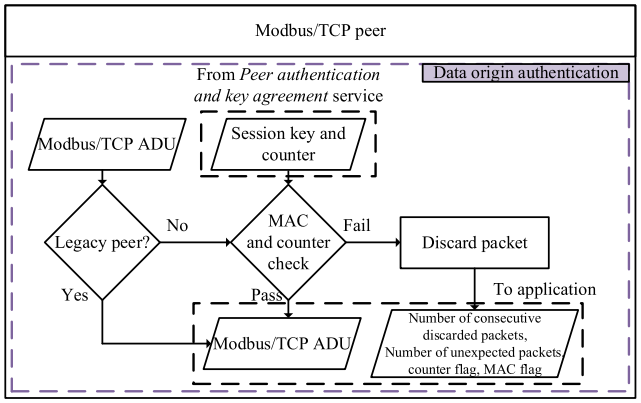

FIGURE 6. Flowchart of *Data origin authentication* service.

If the service detects that the PDU has been tampered with, it immediately discards the PDU and sets the appropriate flags. In addition, the service increments the internal counters which count different cybersecurity incidents. All internal data is then sent to the server application, which can initiate safety-related procedures, depending on the cybersecurity violation. If the request was authentic, the *Data origin authentication* service sends the original ADU to the server application. The server then responds to the client depending on the request, protecting the ADU using the *Modbus/TCP ADU MAC protection* service, as shown in Fig. 7. The authentication mechanism is repeated on the client side, allowing authenticated communication between peers.

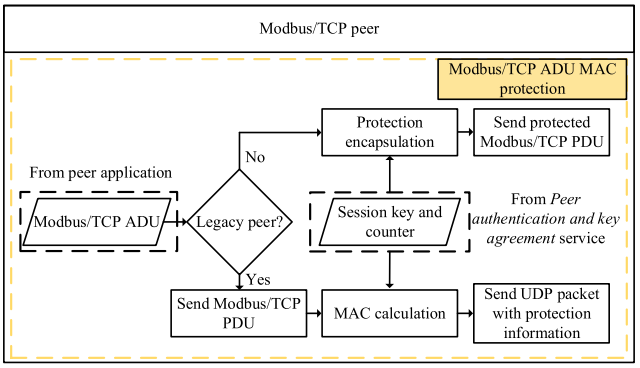

FIGURE 7. Flowchart of *Modbus/TCP ADU MAC protection* service.

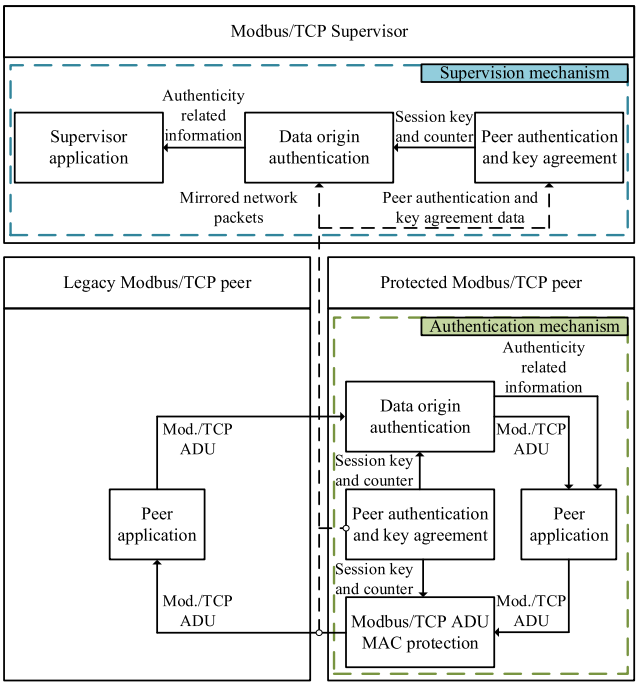

FIGURE 8. *Modbus/TCP Supervisor* device implementation.

D. HANDLING LEGACY PEERS

As can be seen from Figs. 6 and 7, the protection steps are different when communicating with devices that do not support the protection method. When communicating with such devices, the protected peers send standard unaltered Modbus/TCP PDUs, after which they create and send an additional UDP packet to the legacy peer, which is at least composed of a MAC and a PDU counter. Additionally, the UDP packet consists of information which should be sufficient to unambiguously determine which of the earlier sent standard Modbus/TCP PDUs the UDP packet relates to. Both the unprotected and the UDP packets are captured by an additional novel device implemented within the OT network, which we named the *Modbus/TCP Supervisor*, as shown in Fig. 8. The device verifies the authenticity of the Modbus/TCP PDUs sent by the protected peers on the network by using the *Data origin authentication* service, as well as the information received from the *Peer authentication and*

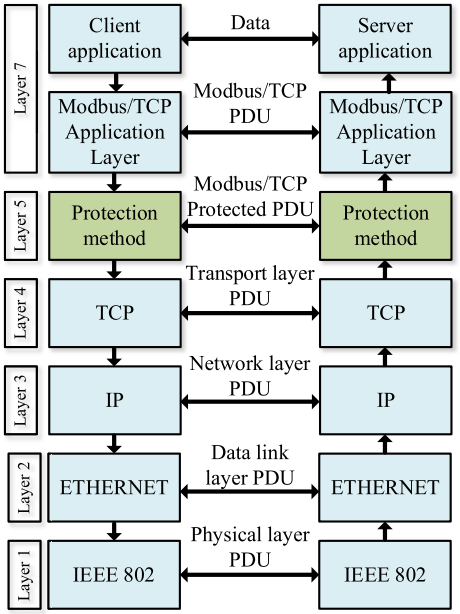

FIGURE 9. OSI location of the developed protection mechanism.

TABLE 1. Defence against active attacks on Modbus/TCP.

PDU attack	Protection mechanism	Description
Replay/ Order change	PDU counter implementation	With the PDU counter implementation, an attacker cannot replay a PDU that has already been processed by the *Data origin authentication* service.
Deletion		With the PDU counter implementation, an attacker cannot delete/discard a PDU without the *Data origin authentication* service detecting it.
Insertion	MAC implementation	Without a valid session key, it is not possible to insert a crafted Modbus/TCP PDU without it being detected by the *Data origin authentication* service.
Modification		Without a valid session key, which an attacker cannot obtain due to the threat model definition, it is not possible to modify an existing protected Modbus/TCP PDU without being detected by the *Data origin authentication* service.

key agreement service. *Modbus/TCP Supervisor* captures the Modbus/TCP PDUs and the UDP packets by using *switched port analyzer* (SPAN), also known as the port mirroring service. Port mirroring sends a copy of the network traffic between network ports to a SPAN port, to which *Modbus/TCP Supervisor* is then connected. It should be noted that the SPAN feature is not common in general-purpose Ethernet switches and can be found in switches designated as ''Managed''. The supervisor calculates the MAC of the unprotected Modbus/TCP PDU and compares it with the one acquired from the UDP packet. When the supervisor detects a violation of authenticity, it sets the corresponding flags, alarming the operator who supervises the process. If the *Modbus/TCP Supervisor* device is implemented within the network, it could also supervise the PDUs exchanged between the protected Modbus/TCP peers, allowing real-time cybersecurity monitoring of the complete OT network. Other than alarming, the *Modbus/TCP Supervisor* can also initiate security and safety related procedures, whose definition is IACS-specific.

E. ANALYSIS OF THE PROTECTION METHOD

Our protection mechanism does not change the Modbus/TCP Application Protocol but adds the developed authentication mechanism to the fifth layer (session layer) of the OSI model, as shown in Fig. 9. Unlike other authors, we emphasize that our protection method could be implemented along with legacy devices, such as PLCs, allowing protection without adding any BITW devices.

As for the data origin authentication requirement, Table 1 gives a list of active attacks which our protection mechanism protects from. The list is similar to the one given in [34], which defines a list of attacks to be considered when creating a Request For Comments (RFC) Security Consideration section.

As can be seen, the PDU counter allows peers to detect the replayed PDU and determine if any of the transmitted PDUs were deleted before being processed by the peers. The MAC, on the other hand allows the detection of PDUs that have been spoofed by an adversary, as well as the detection of the modified Modbus/TCP PDUs.

In addition to active attacks on Modbus/TCP PDUs, an attacker can also initiate DoS attacks on Modbus/TCP devices, mostly because TCP is vulnerable to several DoS attacks (e.g., TCP SYN flood, forged RST messages) [18]. Our protection mechanism, being located on top of TCP, cannot protect against such attacks. As for DoS attacks which try to exploit possible vulnerabilities of the Modbus/TCP application layer protocol, in combination with the PDU counter MAC prevents DoS attacks on the peers, since the attacker would need to break our protection in order to mount a DoS attack.

Regarding the size of the transferred MAC and PDU counter fields, it was said that the size of the fields is implementation-defined. In terms of cybersecurity, the MAC field length should be defined as long as necessary to reduce the probability of MAC-guessing attacks [29].

The PDU counter field length is important for analyzing potential counter overflows, which would theoretically allow PDU replay-attacks. For example, if a client application requires a communication cycle of 100 milliseconds, then the number of Modbus/TCP requests will be 10 within a second, 600 within a minute, 36,000 within an hour, 864,000 within a day, and finally $3.1536{*}10^8$ requests in a year. If the PDU counter length m is defined as 4 bytes, it gives a total of 2^{32} values that the counter can assume. Based on the given, the

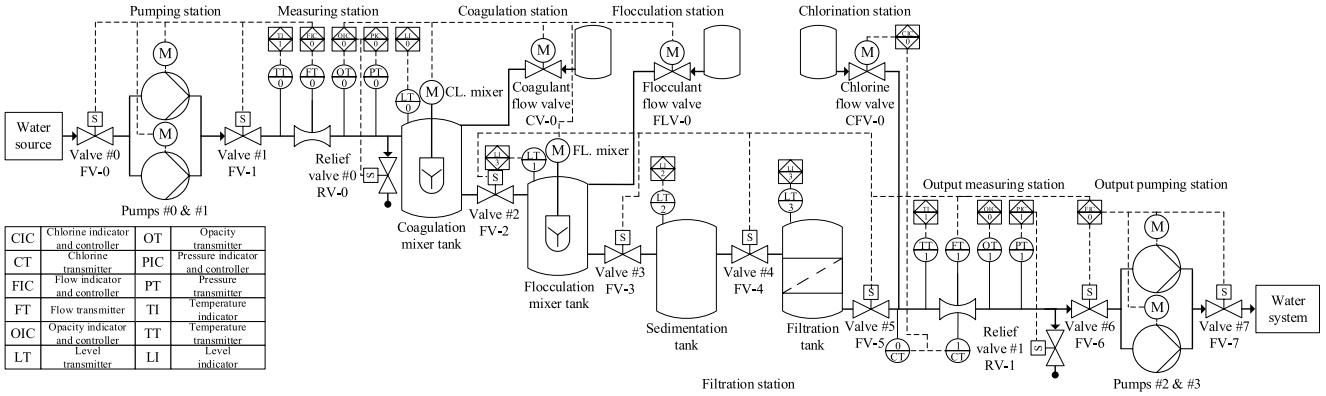

FIGURE 10. P&ID diagram of implemented CPS.

elapsed time before the PDU counter overflows T_{ov} equals:

$$Tov = \frac{2^{32}}{3.1536 * 10^8} \approx 13.62 \; years \tag{1}$$

while keeping the earlier stated fact in mind that the IACSs work continuously for decades [1]. As earlier defined, possible overflows are solved by the *Peer authentication and key management* service, which is not analyzed in this study, however the system designer should take care when specifying field length to prevent frequent key-agreement exchanges, which could potentially impact the system performance.

When we discuss the availability requirement, we could argue that by using symmetric key algorithms, the availability of the IACS communication system is impacted less than it would be if we used asymmetric key methods, as stressed in [19]. To confirm this, a detailed performance and cybersecurity analysis of the protection method will be given in Section V.

Because the protected Modbus/TCP PDU is not encrypted, the method supports additional automated IDS monitoring of the data within the network. The *Modbus/TCP Supervisor* device, which is in fact one form of a misuse-based IDS, is required only if the protected Modbus/TCP peers communicate with the unprotected ones, allowing the detection of attacks on the protected PDUs. To prevent the introduction of new attack vectors by adding a new device within the IACS network, it is advised that *Modbus/TCP Supervisor* is adequately protected using standard cybersecurity best-practices, such as network segmentation and access management.

When communicating with legacy devices, MAC and PDU counter are transferred to *Modbus/TCP Supervisor* by an UDP packet, which could be theoretically removed/dropped from the network by an attacker. Because the communication in IACSs is almost always cyclic, *Supervisor* can detect that it did not acquire the protection information, alarming the operator supervising the process. Although there is a possibility of a random (or targeted) UDP packet loss on the network, we chose the use of UDP (rather than TCP) because of the simplicity of implementation, reduced overhead, increased speed, as well as minimal impact on network bandwidth [1].

V. EXPERIMENTAL VALIDATION OF THE PROPOSED PROTECTION METHOD

The goal of this section is to show how the proposed method was validated. Firstly, we describe the architecture of the developed water treatment CPS. After that, we describe the implementation of the authentication mechanism and review its cybersecurity and performance.

A. CYBER PHYSICAL SYSTEM

To provide an experimental attack environment for the validation of the protection method, a water treatment IACS was chosen. The motivation for choosing a water treatment plant is that the EPCIP categorized water systems as Transport CI [2]. Additionally, water treatment plants have already been victims of cybersecurity incidents [36].

A detailed process and instrumentation diagram (P&ID) of the developed CPS is shown in Fig. 10, while the process values are listed in Table 2. The water treatment process starts with pumping water from a fresh water source. After the pumping process, the water enters the measuring station where the flow rate, temperature, opacity, and water pressure are measured. Depending on the opacity level, the coagulation and flocculation chemicals are added to coagulation and flocculation mixing tanks, respectively. During the passage through the sedimentation and filtration tank, sludge and harmful substances are removed from the water. The filtration system is purely mechanical, after which the water enters the chlorination phase. Based on the current chlorine content in the water, the required amount of chlorine is added. Finally, the water process values are measured, after which the water is pumped into the system.

The process control and software simulation of the chosen water treatment CPS was performed with two Siemens S7-1513-1 PN PLCs. The first PLC, called the Digital Twin (DT) PLC, was used to fully simulate the water treatment process. The advantages of the DT use in the cybersecurity analysis are significant when compared to physical testbeds as they eliminate the possibility of physical damage to the expensive industrial components and allow the modification of the CPSs free-of-cost and the creation of cybersecurity training ranges [37].

TABLE 2. List of process values.

Location	Actuator/ sensor	Process value	Controllable from SCADA
Pumping station	FV-0	Flow valve	YES (manually)
	FIC-0	Flow indicator and controller	YES
Measuring station	FV-1	Flow valve	YES (manually)
	TT-0	Temperature sensor	-
	FT-0	Flow sensor	-
	OT-0	Opacity sensor	-
	PT-0	Pressure sensor	-
	RV-0	Relief valve	YES (manually)
Coagulation station	OIC-0	Opacity indicator and controller	-
	LT-0	Level transmitter	-
	LI-0	Level indicator	-
	FV-2	Flow valve	YES (manually)
	CV-0	Coagulant control valve	-
	CM-0	Coagulant mixer	YES (manually)
Flocculation station	LT-1	Level transmitter	-
	LI-1	Level indicator	-
	FV-3	Flow valve	YES (manually)
	FLV-0	Flocculant control valve	-
	FLM-0	Flocculant mixer	YES (manually)
Filtration station	LT-2	Level transmitter	-
	LI-2	Level indicator	-
	LT-3	Level transmitter	-
	LI-3	Level indicator	-
	FV-4	Flow valve	YES (manually)
	FV-5	Flow valve	YES (manually)
Chlorination station	CIC-0	Chlorine indicator and controller	-
	CFV-0	Chlorine flow valve	-
	CT-0	Chlorine transmitter	-
Output measuring station	FV-6	Flow valve	YES (manually)
	TT-1	Temperature sensor	-
	FT-1	Flow sensor	-
	CT-1	Chlorine transmitter	-
	OT-1	Opacity sensor	-
	PT-1	Pressure sensor	-
	FV-6	Flow valve	YES (manually)
	RV-1	Relief valve	YES (manually)
Output pumping station	FIC-1	Flow indicator and controller	-
	FV-7	Flow valve	YES (manually)

The second PLC, named the Control PLC was used to automate the created CPS. The programming solutions were created using the TIA Portal v16 tool, an engineering platform for industrial equipment produced by Siemens. As a supervising platform, a PcVue SCADA system [38] was implemented within the CPS, while the engineering workstation allows the CPS maintenance. One part of the created SCADA interface is shown in Fig. 11. Both the SCADA and the engineering workstation use Windows 10 Pro version 21H2 operating systems. The SCADA operator can choose between two modes of operation, namely the *Automatic mode* in which the operator can change the flow reference, and the *Manual mode* in which the operator can additionally open and close the pipeline valves.

The automation system has integrated safety mechanism procedures in which the Control PLC brings the IACS to

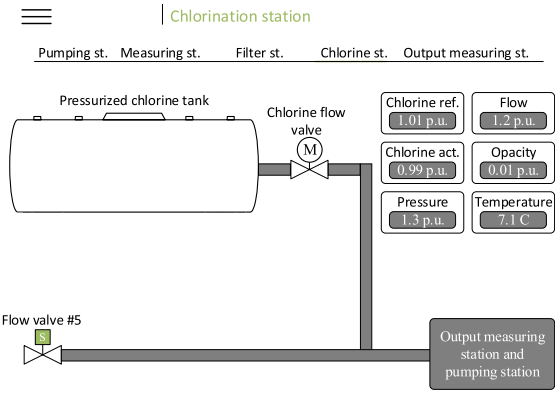

FIGURE 11. Chlorination station within SCADA interface.

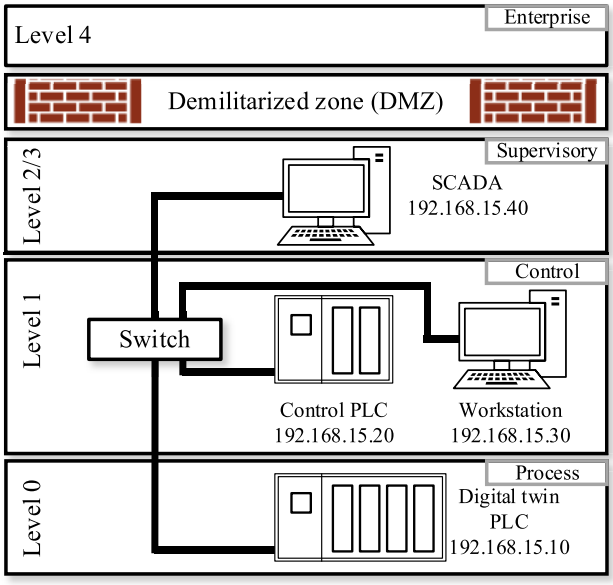

FIGURE 12. Network architecture of implemented CPS.

a safe state when, for example, the operator tries to close the pipeline valves while the pump is running, potentially causing overpressure in the system. The functional safety mechanisms are generally engineered after functional safety risk assessments have been conducted, such as the hazard and operability study (HAZOP) [1]. Functional safety risk assessments are used to identify potential situations within IACSs that pose a significant risk to the system and the environment. Their cybersecurity counterpart are cybersecurity risk assessments, which will be analyzed later in this section.

The network architecture of the developed CPS is shown in Fig. 12, which is similar to the aforementioned system model. As can be seen, the DT PLC is located within the process level, the Control PLC and the engineering workstation are placed in the control level, and finally, the SCADA interface is located within the supervisory level. All devices are connected to LAN using an Ethernet switch.

Regarding the Modbus/TCP application relations between the devices, the SCADA system is a Modbus/TCP client to the Control PLC, and the Control PLC is a client to the

TABLE 3. CPS risk matrix.

Asset	Loss of cybersecurity requirement	Worst case	Risk
Communication path between Control PLC and Digital twin PLC	Availability	CPS cannot be controlled.	High
	Data origin authentication	Destruction of water treatment plant, human casualties.	Critical
Communication path between SCADA and Control PLC	Availability	CPS cannot be supervised.	Medium
	Data origin authentication	Change of flow reference, tricking SCADA operator with different process values.	Medium

DT PLC. The Control PLC uses WRITE and READ Modbus/TCP function codes, which allow the transfer of 16-bit registers loaded with different process values, depending on the programming implementation. As for our implementation, the Control PLC writes 10 and reads 30 registers from the DT PLC. The Modbus/TCP communication cycle repeats itself every 100 milliseconds, allowing continuous control and supervision of the CPS.

As for the threat model, it was similar to the previously defined model, with a potential risk in IACS operations given in Table 3. The risk levels were modelled from low to critical, where low means little or no harm to the organization and critical implies possible human casualties [32]. As can be seen, the loss of availability and authenticity of system assets can have a significant impact on IACS operations. It was assumed that the organization managing the CPS does not consider confidentiality and nonrepudiation as cybersecurity requirements. In general, the initial IACS cybersecurity risk assessment can be performed by analyzing the worst-case scenarios in which the selected assets (e.g., equipment, communication conduits) are compromised [32]. Loss of availability and data origin authentication between the Control PLC and DT PLC poses a higher risk than an attack on the communication path between SCADA and Control PLC. The reason behind this are the previously mentioned safety-related functions inside the Control PLC in which the PLC sets the CPS into a fail-safe state if the attacker would try to modify SCADA commands while trying to destabilize the system. Additionally, the SCADA operator can manually initiate a limited set of operations, which cannot bring the CPS to an unsafe state. On the other hand, data tampering attack on the communication between the PLCs directly violates the data origin authentication requirement, which could be used by an attacker to destroy the water treatment plant or poison the water with chlorine, critically impacting the organization.

Finally, it is assumed that the organization has accepted the imposed risk if the communication between SCADA and Control PLC is attacked. On the other hand, it is required that the impact, in the case of an attack on the Control PLC – DT PLC communication path, is minimized.

Algorithm 1: CHASKEY-12

Step 1: Subkeys derivation
Generate **Subkeys** (K_1, K_2) using 128-bit key K
Step 2: Padding of the message $\boldsymbol{m}$
d = (m modulo 128)
if $d \neq 0$ **or** $|m| = 0$ **then**
  Pad the m with single "1", then 128-d-1 "0" bits
  Padding = TRUE
else
  Padding = FALSE
end if
m'=m
Step 3: Application of the permutation
Split the message m' into l 128-bit messages
$V \leftarrow K$
for $i = 1$ **to** l-1 **do**
  $V \leftarrow$ Function $\boldsymbol{\pi}$ (V **xor** m_i)
end for
if *Padding* = FALSE **then**
  $L \leftarrow K_1$
else
  $L \leftarrow K_2$
end if
$S \leftarrow$ Function $\boldsymbol{\pi}$ (V **xor** m_l **xor** L) **xor** L
Step 4: Truncation Save the t least significant bits of the S

FIGURE 13. Chaskey-12 MAC algorithm.

B. CHASKEY-12 MESSAGE AUTHENTICATION CODE

Lightweight cryptography methods use less computational and energy resources when compared to standard cryptography, which makes lightweight MACs suitable for implementation in IACS devices [28]. Based on this fact and the analysis [39] in which the performance of lightweight block ciphers was given, we chose Chaskey-12 as the underlying MAC algorithm. Also,

Chaskey-12 was optimized for 32-bit microcontrollers [39], which are commonly used by PLC vendors. It uses only basic addition, rotation, and XOR functions, meaning that it can be implemented using the IEC 61131-3 languages. In general, Chaskey-12 uses a 128-bit symmetric key K on an arbitrary-length message m to generate a MAC with a length of 128-bits or less, depending on the security requirements. The algorithm of the Chaskey-12 MAC is given in Fig. 13. Within the Chaskey-12 algorithm, *Subkey* function is initially called to derive subkeys from the given key K. After deriving the subkeys, the message is padded depending on its size, while setting the corresponding *padding* bit. When the padding process is completed, the padded message is split into 128-bit messages, after which the 12-round permutation function π iterates over the split messages. Depending on the *padding* bit, the last operation on 128-bit message is executed using subkey K_1 or K_2. The final result is the Chaskey-12 MAC, which can be $\leq$ 128 bits. More details on Chaskey-12 are given in [27].

C. IMPLEMENTATION OF THE CHASKEY-12-BASED AUTHENTICATION MECHANISM

We have developed a Chaskey-12 function block (FB) using Structured Text (ST) programming language, a PLC programming language defined by the IEC 61131-3 standard.

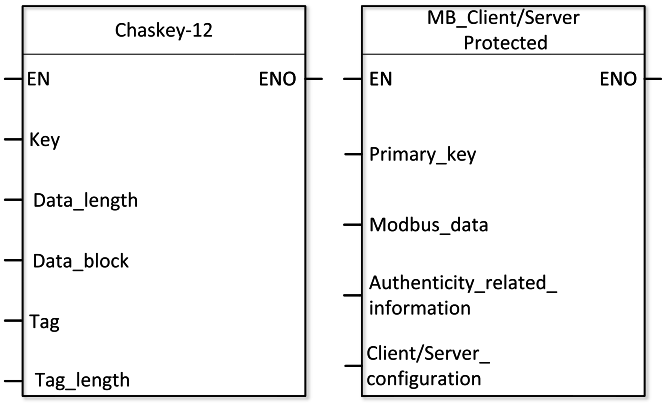

FIGURE 14. Chaskey-12 function block (left), Modbus/TCP client/server protected function block (right).

The ST programming language is a textual programming language which is derived from the Pascal programming language, as opposed to the Ladder Diagram (LD), a graphic programming language which resembles relay ladder logic diagrams [40]. The use of a standardized programming language enables the use of the developed FB on any other PLC that supports the IEC 61131-3 defined languages. Our implementation assumes little-endian architecture, which is used both in the S7-1200 and S7-1500 PLC series [41]. It should be noted that endianness in IACS equipment is not strictly defined, so the programmer should take care when implementing such algorithms. Additionally, the block could be used for any other use that is not Modbus/TCP related, such as protecting other IE protocols using MACs. The Chaskey-12 implementation was tested against the test examples given in [27].

The development platform used for the algorithm implementation was the TIA Portal v16. Fig. 14 (left) shows the developed FB, whose block's interface is described in Table 4. When developing an FB interface, there are multiple variable types in the IEC 61131-3 programming languages, namely the Inputs, InOuts, and Outputs. An *Input* to the block passes the value of the input variable to the block, while an *InOut* passes the variable pointer. *Outputs* are used to pass the block's result to the assigned variables.

After developing and testing the Chaskey-12 FB, we integrated the FB with Modbus/TCP using proprietary Modbus/TCP FBs, namely the *MODBUS_CLIENT* and *MODBUS_SERVER* blocks. The given MODBUS blocks are developed for Siemens S7-1200/1500 PLC series and allow Modbus/TCP communication via IE. Such FB block implementation of Modbus/TCP is common in industrial automation, as other PLC vendors [42], [43] use this type of interface for the Modbus/TCP communication as well.

The proposed MAC-based protection method of Modbus/TCP was created using the ST and LD programming languages, and it was finally encapsulated within the two FBs named *MB_Client_Protected* and *MB_Server_Protected*. Fig. 14 (right) depicts the layout of the developed FBs, while Table 5 lists the block interfaces.

As can be seen, the blocks assume three different user defined types (UDTs), a structure-like IEC 61131-3 data type which can be freely defined by the user [40]. The UDTs transfer the information about the client/server configuration (e.g., requested function code, number of registers, if the peer supports the protection algorithm) and information from the *Data origin authentication* service to the user program. Details on the UDTs are given in Appendix. As for the *Primary_key* variable, it is a value that is pre-shared on the PLCs by the automation engineer, from which the session key and counter values would be calculated by the *Peer authentication and key management* service, which was not implemented. For testing purposes, the symmetric key was pre-shared on both peers, while the PDU counter, starting at 0, was internally incremented for every successfully received PDU. Finally, the FBs assume access to the data block via *Modbus_data* in which the process values are stored. The implemented *Data origin authentication* and *Modbus/TCP ADU Protection* services manipulate the Modbus/TCP ADU right before sending/receiving PDUs, allowing the peer's application to access only standard Modbus/TCP ADUs.

The motivation behind encapsulating the protection method in the FBs is to satisfy the deployment requirement, in which the programmer should be able to deploy the engineered protection method as easy as possible, by just replacing the proprietary Modbus/TCP communication FBs with as little program modification and configuration as possible.

Finally, the client and server security-related functions are encapsulated within the *MB_Client/Server_Security_ procedure* FBs, whose layout is given in Fig. 15.

As an Input, the FBs assume *Authenticity _related_ information* UDT which provides the external security/safety FB with information regarding cybersecurity violation, as well as the Modbus/TCP data block which is used as an interface to the IACS process values.

Fig. 16 shows a flowchart of the developed security FB, which was created specifically for our water treatment CPS. The security FB allows 10 consecutive MAC mismatches, as well as 10 PDUs whose PDU counter is not the same as the internal one. Finally, if the authentic PDU did not arrive

TABLE 4. Chaskey-12 function block solution structure.

Variable type	Variable name	Data type	Description
Input	EN	Boolean (BOOL)	Enable the execution of the block.
	Key	Array[0..3] of Unsigned double integers (UDINT)	128-bit symmetric key.
	Data_length	Unsigned integer (UINT)	Data block length (in 16-bit words).
InOut	Data_block	Array[0..125] of 16-bit string (WORD)	Data block for the MAC calculation.
	Tag	Array[0..3] of Unsigned double integers (UDINT)	Chaskey-12 MAC.
	Tag_length	Unsigned integer (UINT)	Length of Chaskey-12 MAC in bits.
Output	ENO	Boolean (BOOL)	Enable the execution of the next block.

within the time interval of 60 seconds, the safety procedure is initiated. The threshold values for the security FBs were chosen based on the defined communication cycle of 100 milliseconds, assuming the process dynamics allow the CPS to work safely if there was no communication for one minute. It should be noted that there is a risk of blocking the safety values sent from the client by the attacker, so additional safety guards should also be implemented.

The communication treshold values must be based on the aforementioned functional safety risk assessments, which answer questions related to different failures within the IACS, including communication failures [1]. After the security procedure has been initiated, the automation process can only be reinitialized manually by an authenticated programmer.

Our implementation of the authentication mechanism supports two standard Modbus/TCP function codes, READ, and WRITE holding register functions, respectively. Further, the transferred MAC length is 16 bytes, while the length of the counter field is defined as 4 bytes. The 16 bytes MAC length is the complete MAC generated by the Chaskey-12 algorithm, while the 4-byte counter field allows continuous communication before overflow occurs for 13.62 years, as previously mentioned. Within the experimental validation, we assume that the peers have authenticated themselves, and that both the session key and the PDU counter values have been established.

D. EXPERIMENTAL VALIDATION OF THE PROPOSED PROTECTION METHOD

First, we tested the performance of the Chaskey-12 MAC algorithm implementation in the S7-1500 PLC. For the test vectors, we used data blocks ranging in size from 128 to 8,000 bits, filled with randomly selected values. In addition, the performance test was executed on a Siemens S7-1200 series PLC, which is commonly used for smaller and less resource-intensive applications. The S7-1200 work memory is around three times smaller than that of S7-1500, which can be observed when looking at the average runtimes of the Chaskey-12 FB, see Fig. 17.

When compared to the Chaskey-12 MAC implementation in the Allen Bradley ControlLogix 5571 PLC presented in [30], our implementation has a significantly lower runtime cost for the same amount of data. For example, the average execution time of their implementation is around 65 ms for a data size of 260 bytes (maximum size of the Modbus/TCP ADU), while our implementation takes around 5.3 and 2.8 ms for S7-1200 and S7-1500 PLCs, respectively. These results were obtained by averaging 1,000 runtimes of the algorithm.

Assuming that both implementations are correct, which was proven by testing the implementations against the test vectors given in [27], such runtime difference could theoretically be caused by the fact that the authors had to create their own bitwise shift/rotate instructions as the used PLC does not provide them in the ST language [30]. Concerning the difference between the PLC resources, both the ControlLogix 5571 and S7-1500 series are high performance PLCs, which means that they are used for similar complex and high-performance systems. As for our results, there are no significant deviations from the average runtimes, while the CPU runtime linearly increases with the data size. The runtime of the S7-1500 series is lower than that of S7-1200 for its higher amount of resources, which was also expected. It should be noted that all measurements of the program runtimes were performed using the proprietary Siemens *RUNTIME* instruction [41].

After proving the feasibility of the real-time execution of Chaskey-12 MAC within the PLC program cycle, the next step was to determine the same for the developed protection method. As mentioned earlier, we have not implemented the *Peer authentication and key agreement* service, so a prerequisite for developing such a solution is that our protection method leaves enough time for the given service to authenticate the peers and exchange the session key. The tests were carried out with the S7-1500 Modbus/TCP peers.

Fig. 18 shows the communication latency caused by the protection method for different transmitted data sizes. The results were obtained by subtracting the arrival times of Modbus/TCP requests and responses with the protection method being enabled and disabled, respectively. As for the Modbus/TCP data field size, the minimum test size was 2 bytes (1 holding register) and the maximum size L_{max} was determined as

$$L_{max} = 250bytes - PDUCounter - MAC = 230bytes \quad (2)$$

where 250 bytes was the maximum length of the Modbus/TCP data field when using the proprietary MB_CLIENT/SERVER FBs, MAC field with a length of 16 bytes and the PDU Counter field with a length of 4 bytes. Regarding the overhead, our implementation of the protection method results in an overhead of around 8.7% when the maximum allowed number of Modbus/TCP registers is exchanged.

The communication latency of the unprotected Modbus/TCP increases linearly with the data size, leading to

TABLE 5. MB_CLIENT/SERVER function block solution structure.

Variable type	Variable name	Data type	Description
Input	EN	Boolean (BOOL)	Enable the execution of the block.
	Master_key	Array[0..3] of Unsigned double integers (UDINT)	128-bit symmetric master key.
InOut	Modbus_data	Array[0..125] of 16-bit words (WORD)	Modbus/TCP data block.
	Client/Server_ configuration	UDT "Client_ Configuration" or "Server_ Configuration"	Client/Server IP and port configuration, function codes, Modbus data length.
	Authenticity_ related_ information	UDT "Authenticity_ related_ information"	Flags and dropped PDU counter values from the *Data origin authentication* service.
Output	ENO	Boolean (BOOL)	Enable the execution of the next block.

TABLE 6. PLC CPU runtime during request/response handling.

PLC	Protection OFF		Protection ON	
	Average (ms)	Worst (ms)	Average (ms)	Worst (ms)
Control PLC	2.7	3.7	4.5	5.7
Digital Twin PLC	2.1	2.6	6.2	7.3

similar results as in [44]. When the Modbus/TCP communication is protected, the additional latency also rises linearly and is mainly caused by MAC calculations, as both peers need to check the received PDUs as well as calculate the MAC when sending PDUs.

Next, we present the performance experiments on the implemented CPS. The tests were performed using the predefined communication cycle of 100 ms, with the Control PLC writing 10 and reading 30 16-bit registers from the DT PLC.

Fig. 19 (a) shows a comparison of the PLC's CPU runtime with and without the protection method implemented for the Modbus/TCP client (Control PLC), while Fig. 19 (b) gives the same for the server (DT PLC). As can be seen, the protection method increases the PLC's runtime when Modbus/TCP requests and responses are handled by the peers. The protected Modbus/TCP servers must compute a Chaskey-12 MAC for both the request and the response in the same runtime cycle, while clients handle the requests and responses in different CPU runtime cycles due to communication latency. Table 6 shows the average and worst runtimes calculated using the CPU runtime values when the Modbus/TCP PDU is sent/received.

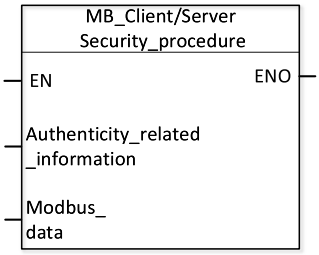

FIGURE 15. Security procedure function block.

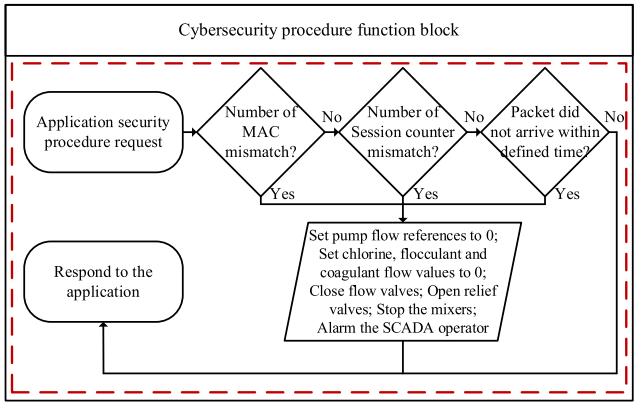

FIGURE 16. Cybersecurity procedure function block.

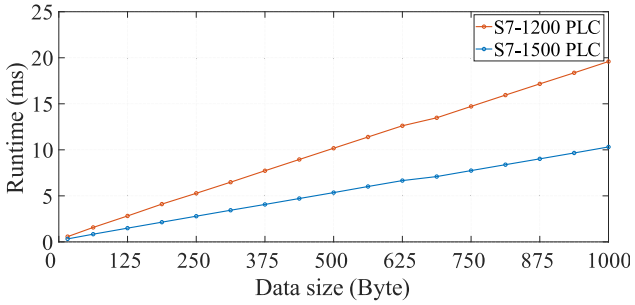

FIGURE 17. Chaskey-12 FB runtime on S7-1200 and S7-1500 PLCs.

On average, the protection method increases the runtime of the Control PLC during the Modbus/TCP request/response to about 4.5 ms, which is a 66% increase compared to 2.7 ms when no protection is implemented. For the DT PLC, the difference is greater with runtimes increasing by 195%, from 2.1 to 6.2 ms, when the protection is added. The added CPU runtime does not affect the CPS, as it does not cross the 100 ms threshold which was defined based on the CPS automation requirements.

After the performance evaluation, we tested the implemented protection method against the Modbus/TCP ADU modification, insertion, replay and deletion attacks, while assuming the previous defined threat model. When located within the OT network, the attacker can easily assume an MITM position between the peers by exploiting lower-level protocol vulnerabilities. One example of such attack is the address resolution protocol (ARP) spoofing, which sets the attacker directly in the communication path, allowing execution of the aforementioned active attacks [1].

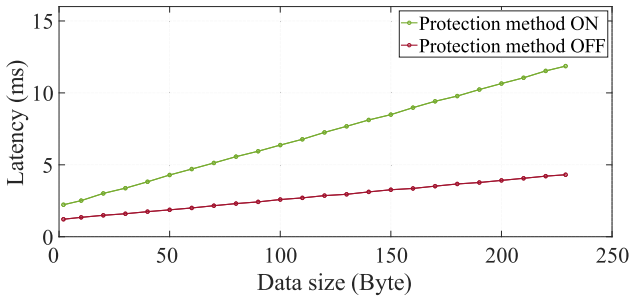

FIGURE 18. Communication latency of Modbus/TCP protection method.

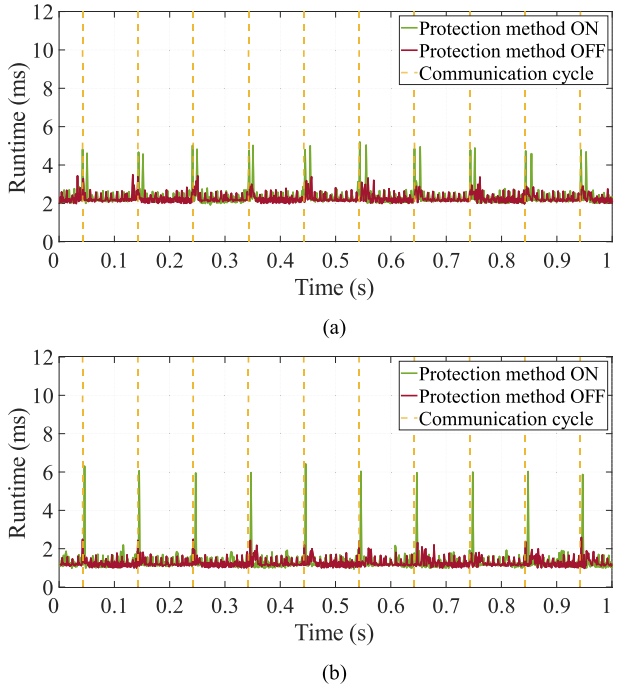

FIGURE 19. CPU program runtime with and without protection method on (a) Control PLC (client) and (b) Digital Twin PLC (server).

All of the active attacks were performed using the tools included into Kali Linux, an OS specifically designed for ethical penetration testing [45]. The Kali Linux OS has multiple different tools used for information gathering, vulnerability analysis and many more, one of which is the Ettercap sniffing and spoofing tool [46]. Ettercap enables the execution of all given active attacks, with an additional help from the Wireshark tool used for the network PDU analyzing and capturing. The replay attack was performed by using Tcpreplay, a tool which allows a replay of previously captured network traffic [47].

The first attack performed was a PDU modification attack, in which the attacker took an MITM position and created a special Ettercap script (also known as *filter*) that modified the WRITE holding register requests, specifically the chlorine setpoint, which was a part of the CIC-0 control loop. For clarity, the values are expressed as per unit (p.u.). Fig. 20 shows the results of the attack, where the protection method successfully detected the manipulated data and

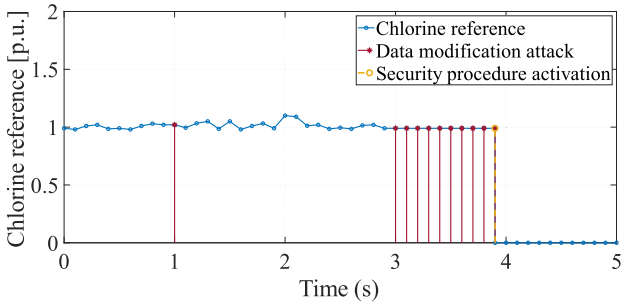

FIGURE 20. PDU modification attack on Modbus/TCP client-server communication path.

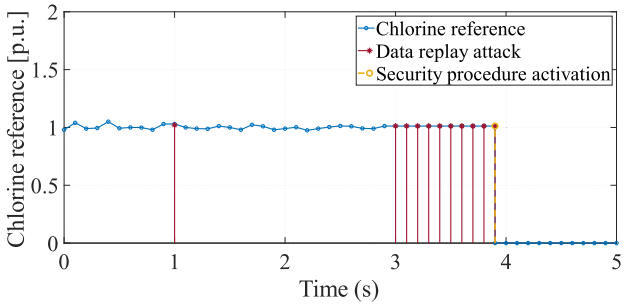

FIGURE 21. PDU replay attack on Modbus/TCP client-server communication path.

discarded the PDU. First, the attacker injected one PDU with a modified chlorine setpoint of 2.0 p.u. at time T= 1 second, after which the PDU was discarded due to the failed *Data origin authentication* MAC check. The DT PLC held the last valid chlorine setpoint value which had been acquired in the previous communication cycle. In the third second, the attacker started modifying the Modbus/TCP requests/responses in each communication cycle, activating the safety procedure after 10 consecutive PDU modifications. As previously defined, the safety procedure set the chlorine reference value to 0 p.u., which prevented the attacker from causing damage to the CPS.

The second attack performed was a message insertion attack, where the attacker created a forged message with desired data values (modified chlorine setpoint value) and forged the TCP header data (sequence and acknowledgement numbers) to trick the remote peers that the PDU was valid. The created PDU was dropped again by the protection method due to the failed MAC check, and the peer's performance was similar to the results obtained when testing against the PDU modification attack. When we tried to insert a forged message without changing the TCP header accordingly, TCP detected and dropped the inserted PDU, while maintaining the TCP communication session between the Modbus/TCP peers.

When we tried to replay the captured network PDUs, TCP detected and dropped the spurious packets. However, when we modified the TCP session and acknowledgement number fields accordingly, the *Data origin authentication* service in the protection method detected that the Modbus/TCP PDUs were replayed due to the failed PDU counter check, as shown in Fig. 21. After 10 consecutive replayed PDUs (starting from

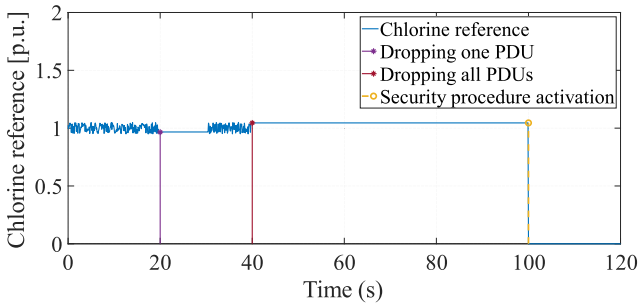

FIGURE 22. PDU deletion attack on Modbus/TCP client-server communication path.

T= 3 seconds), the safety procedure was activated, protecting the CPS from cybersecurity incident.

The last attack analyzed was a message deletion attack, where an attacker dropped one or multiple PDUs. Fig. 22 shows the results of the performed attack, in which the attacker first dropped one PDU in T= 20 seconds, after which it took the peers about 10 seconds to restore the communication. This was due to the settings of the MB_CLIENT/SERVER FBs used, which were set to retransmit the packets every 10 seconds if they had not received an acknowledgement of the PDU transmission. In the 40th second we initiated an attack in which we dropped all transmitted PDUs, effectively causing a DoS. 60 seconds after the start of the attack, the safety procedure was initiated, bringing the CPS into a fail-safe state. In general, message deletion attacks are relatively difficult to detect, as the peers cannot tell if the communication path is under attack, or the communication is interrupted due to a physical failure (e.g., broken or unplugged Ethernet cables, network congestion).

For this reason, time intervals for detecting a DoS attack should be long enough to prevent accidental activation of the engineered safety procedure, but short enough to prevent any significant damage to the organization managing the CPS.

VI. DISCUSSION

We have proposed and presented one potential protection method for Modbus/TCP communication, but we had to make several simplifications in the implementation process.

First, by assuming the existence of the *Peer authentication and key management* service, we skipped the development of the *key management* procedure [48]. Currently, there are no generally accepted methods and procedures for key management in IACSs as they range from pre-shared keys to complex key distribution centers, each of which brings its own set of problems [35].

In addition, the assumption that session keys are protected by preventing unauthorized physical access to PLCs is also problematic, as there are real-life examples where attackers have obtained information from the PLC memory through network access [49]. Due to the use of proprietary Modbus/TCP communication FBs, we cannot implement our developed PLC protection program directly in PLCs manufactured by other vendors, as each manufacturer has their own implementation of such blocks.

Based on the fact that most vendors create their own implementations of such FBs, we urge PLC manufacturers to develop a product-independent Modbus/TCP communication solution that could possibly take a form similar to the one we have presented, where the PLC programmer could easily implement the protected Modbus/TCP solution within an IACS. Similar product-independent programming solutions already exist, for example, in the form of *PLCopen*, which provides programming solutions for motion control, safety, etc. [48].

Further, our protection method could be integrated in Modbus/TCP by creating *authenticated* versions of standard public Modbus/TCP function codes (e.g., read/write multiple holding registers) as the Modbus/TCP specification allows the creation of *user-defined* function codes [4]. Apart from creating new function codes, it could be possible to add new exception codes which would be returned to communication peers if a received PDU is not authentic, allowing the creation of even more precise and correct security and safety related procedures.

Finally, an attacker could theoretically use the protection method to compromise the availability of an IACS. Our CPS safety procedure was the same for every cybersecurity violation, but this need not be the case for some other IACS. Depending on different fail-safety procedures, an attacker could maximize the damage to the IACS by activating the safety procedure that causes the most damage. Of course, this would imply that the attacker has detailed information about the IACS, which could be expected since APTs commonly target IACSs [10].

VII. CONCLUSION AND FUTURE WORK

In this study, we have proposed a novel cybersecurity protection method for Modbus/TCP that provides message authenticity without a significant impact on system availability. The protection is achieved through MAC authentication, which provides integrity protection and authentication of Modbus/TCP PDUs. The proposed method adds an additional layer of security on top of TCP without directly modifying the Modbus/TCP application protocol. To demonstrate the feasibility of this method, a water treatment CPS was implemented as DT in a PLC program. The lightweight Chaskey-12 algorithm was chosen as the underlying MAC algorithm and successfully implemented using IEC 61131-3 programming languages. After testing the performance of the Chaskey-12 MAC, the proposed protection method was implemented and encapsulated in the MB_Protected FBs. Finally, the protection method was successfully tested against active attacks, enabling the protection of legacy PLCs without any additional cost or equipment. To our knowledge, this is the first time that Modbus/TCP has been protected using lightweight cryptography. As an additional novelty, we propose a supervising device that checks the protected Modbus/TCP PDUs when communicating with devices that do not support the protection method, allowing an integration of the protection method in existing IACSs.

In the future, we will investigate and develop possible key management solutions that would be suitable for the developed protection method as well as develop the already mentioned supervising device, which will be used to verify that the protection is provided when protected peers communicate with unprotected peers. Additionally, we will investigate the use of TCP *Authenticated Option* (AO) as a possible solution for ensuring authenticity of TCP segments without modifying the original Modbus/TCP.

APPENDIX

TABLE 7. Structure of CLIENT/SERVER_CONFIGURATION UDT.

Variable name	Data type	Description
MB_time_cycle	Time	Communication time cycle.
MB_mode	Unsigned short integer (USINT)	Modbus/TCP function code.
MB_data_address	Unsigned double integer (UDINT)	Pointer to Modbus/TCP data block.
MB_data_length	Unsigned integer (UINT)	Number of Modbus/TCP registers to be transferred.
MB_connect	System data type (SDT) TCON_IP_v4	Proprietary SDT for TCP connection parameters (IP addresses, ports, etc.).
MB_status	Word	Connection status.
MB_done	Bool	The block was successfully executed.
MB_busy	Bool	The block is being executed.
MB_error	Bool	Connection error.
MB_disconnect	Bool	Connection lost.

TABLE 8. Structure of AUTHENTICITY_RELATED_INFORMATION UDT.

Variable name	Data type	Description
MAC_error	Bool	Calculated MAC does not match the one received.
Number_of_consecutive_MAC_errors	Unsigned double integer (UDINT)	Number of consecutive MAC mismatches.
PDU_counter_error	Bool	Internal PDU counter does not match the one received.
Number_of_consecutive_PDU_errors	Unsigned double integer (UDINT)	Number of consecutive PDU mismatches.
Successful_communication	Bool	Communication was successful with no cybersecurity violations.

ACKNOWLEDGMENT

The authors would like to thank the companies S.C.A.N. d.o.o. and Autegra d.o.o. for their suggestions and support.

REFERENCES

[1] E. D. Knapp, *Industrial Network Security: Securing Critical Infrastructure Networks for Smart Grid, SCADA, and Other Industrial Control Systems*, 2nd ed. Boston, MA, USA: Syngress, 2014.
[2] European Commission, "Council directive 2008/114/EC of 8 December 2008 on the identification and designation of European critical infrastructures and the assessment of the need to improve their protection," *Off. J. Eur. Union*, vol. 345, pp. 75–82, Dec. 2008.
[3] *Proposal for a Council Recommendation on a Coordinated Approach by the Union to Strengthen the Resilience of Critical Infrastructure, COM/2022/551 Final*, Eur. Commission, Brussels, Belgium, Oct. 2022.
[4] *Modbus Messaging on TCP/IP Implementation Guide V1.0B*, Modbus Org., New York, NY, USA, 2006.
[5] D. Pliatsios, P. Sarigiannidis, T. Lagkas, and A. G. Sarigiannidis, "A survey on SCADA systems: Secure protocols, incidents, threats and tactics," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 1942–1976, 3rd Quart., 2020.
[6] K. Hemsley and R. Fisher, "A history of cyber incidents and threats involving industrial control systems," in *Proc. Int. Conf. Crit. Infrastruct. Protection*, 2018, pp. 215–242.
[7] R. Langner, "Stuxnet: Dissecting a cyberwarfare weapon," *IEEE Secur. Privacy Mag.*, vol. 9, no. 3, pp. 49–51, May 2011.
[8] G. Liang, S. R. Weller, J. Zhao, F. Luo, and Z. Y. Dong, "The 2015 Ukraine blackout: Implications for false data injection attacks," *IEEE Trans. Power Syst.*, vol. 32, no. 4, pp. 3317–3318, Jul. 2017.
[9] ESET Research. (Apr. 2022). *Industroyer2: Industroyer Reloaded*. [Online]. Available: https://www.welivesecurity.com/2022/04/12/industroyer2-industroyer-reloaded/
[10] Cybersecurity & Infrastructure Security Agency (CISA). (Apr. 2022). *APT Cyber Tools Targeting ICS/SCADA Devices*. [Online]. Available: https://www.cisa.gov/uscert/ncas/alerts/aa22-103a
[11] B. Galloway and G. P. Hancke, "Introduction to industrial control networks," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 2, pp. 860–880, 2nd Quart., 2013.
[12] K. Stouffer, J. Falco, and K. Scarfone, *Guide to Industrial Control Systems (ICS) Security*, Standard NIST SP 800(82), 2015.
[13] B. Chen, N. Pattanaik, A. Goulart, K. L. Butler-Purry, and D. Kundur, "Implementing attacks for Modbus/TCP protocol in a real-time cyber physical system test bed," in *Proc. IEEE Int. Workshop Tech. Committee Commun. Quality Rel. (CQR)*, May 2015, pp. 1–6.
[14] P. Huitsing, R. Chandia, M. Papa, and S. Shenoi, "Attack taxonomies for the modbus protocols," *Int. J. Crit. Infrastruct. Protection*, vol. 1, pp. 37–44, Dec. 2008.
[15] Z. Drias, A. Serhrouchni, and O. Vogel, "Taxonomy of attacks on industrial control protocols," in *Proc. Int. Conf. Protocol Eng. (ICPE) Int. Conf. New Technol. Distrib. Syst. (NTDS)*, Jul. 2015, pp. 1–6.
[16] I. N. Fovino, A. Carcano, M. Masera, and A. Trombetta, "Design and implementation of a secure Modbus protocol," in *Proc. 3rd Int. Conf. Crit. Infrastruct. Protection*, 2009, pp. 83–96.
[17] M. K. Ferst, H. F. M. de Figueiredo, G. Denardin, and J. Lopes, "Implementation of secure communication with modbus and transport layer security protocols," in *Proc. 13th IEEE Int. Conf. Ind. Appl. (INDUSCON)*, Nov. 2018, pp. 155–162.
[18] G. Hayes and K. El-Khatib, "Securing modbus transactions using hash-based message authentication codes and stream transmission control protocol," in *Proc. 3rd Int. Conf. Commun. Inf. Technol. (ICCIT)*, Jun. 2013, pp. 179–184.
[19] A. Shahzad, M. Lee, Y.-K. Lee, S. Kim, N. Xiong, J.-Y. Choi, and Y. Cho, "Real time MODBUS transmissions and cryptography security designs and enhancements of protocol sensitive information," *Symmetry*, vol. 7, no. 3, pp. 1176–1210, Jul. 2015.
[20] E. Pricop, J. Fattahi, N. Parashiv, F. Zamfir, and E. Ghayoula, "Method for authentication of sensors connected on Modbus TCP," in *Proc. 4th Int. Conf. Control, Decis. Inf. Technol. (CoDIT)*, Apr. 2017, pp. 679–683.
[21] W. Hupp, A. Hasandka, R. S. de Carvalho, and D. Saleem, "Module-OT: A hardware security module for operational technology," in *Proc. IEEE Texas Power Energy Conf. (TPEC)*, Feb. 2020, pp. 1–6.
[22] *Modbus/TCP Security Protocol Specification*, Modbus Org., New York, NY, USA, 2018.
[23] I. N. Fovino, A. Carcano, T. De Lacheze Murel, A. Trombetta, and M. Masera, "Modbus/DNP3 state-based intrusion detection system," in *Proc. 24th IEEE Int. Conf. Adv. Inf. Netw. Appl.*, Apr. 2010, pp. 729–736.
[24] F. Katulić, D. Sumina, I. Erceg, and S. Groš, "Enhancing Modbus/TCP-based industrial automation and control systems cybersecurity using a misuse-based intrusion detection system," in *Proc. Int. Symp. Power Electron., Electr. Drives, Autom. Motion (SPEEDAM)*, Jun. 2022, pp. 964–969.

[25] F. Wilkens, S. Haas, J. Amann, and M. Fischer, "Passive, transparent, and selective TLS decryption for network security monitoring," in *Proc. Int. Conf. ICT Syst. Secur. Privacy Protection (IFIP)*, 2022, pp. 87–105.
[26] G. Bernieri, S. Cecconello, M. Conti, and G. Lain, "TAMBUS: A novel authentication method through covert channels for securing industrial networks," *Comput. Netw.*, vol. 183, Dec. 2020, Art. no. 107583.
[27] *Information Technology—Lightweight Cryptography—Part 6: Message Authentication Codes (MACs)*, Standard ISO/IEC 29192-6, 2019.
[28] S. Choi, J. Ko, and J. Kwak, "A study on IoT device authentication protocol for high speed and lightweight," in *Proc. Int. Conf. Platform Technol. Service (PlatCon)*, Jan. 2019, pp. 1–5.
[29] M. Mouha, B. Mennink, A. Van Herrewege, D. Watanabe, B. Preneel, and I. Verbauwhede, "Chaskey: An efficient MAC algorithm for 32-bit microcontrollers," in *Proc. Int. Conf. Sel. Areas Cryptogr. (SAC)*, 2014, pp. 306–323.
[30] Z. Yang, Z. Bao, C. Jin, Z. Liu, and J. Zhou, "PLCrypto: A symmetric cryptographic library for programmable logic controllers," *IACR Trans. Symmetric Cryptol.*, vol. 2021, no. 3, pp. 170–217, Sep. 2021.
[31] A.-V. Duka and B. Genge, "Implementation of Simon and SPECK lightweight block ciphers on programmable logic controllers," in *Proc. 5th Int. Symp. Digit. Forensic Secur. (ISDFS)*, Apr. 2017, pp. 1–6.
[32] *Security for Industrial Automation and Control Systems Part 1–1: Terminology, Concepts, and Models*, Standard IEC 62443-1-1, 2007.
[33] M. Slunjski, D. Sumina, S. Groš, and I. Erceg, "Off-the-shelf solutions as potential cyber threats to industrial environments and simple-to-implement protection methodology," *IEEE Access*, vol. 10, pp. 114735–114748, 2022.
[34] E. Rescorla and B. Korver, "Guidelines for writing RFC text on security considerations," Tech. Rep., 2003. [Online]. Available: https://www.ietf.org/rfc/rfc3552.txt
[35] *Industrial Communication Networks—Network and System Security—Part 3–1: Security Technologies for Industrial Automation and Control Systems*, Standard IEC TR 62443-3-1, 2009.
[36] Cybersecurity & Critical Infrastructure Agency. (Feb. 2021). *Alert (AA21-042A) Compromise of U.S. Water Treatment Facility*. [Online]. Available: https://www.cisa.gov/uscert/ncas/alerts/aa21-042a
[37] C. Gehrmann and M. Gunnarsson, "A digital twin based industrial automation and control system security architecture," *IEEE Trans. Ind. Informat.*, vol. 16, no. 1, pp. 669–680, Jan. 2020.
[38] ARC Informatique. *Why Choose PcVue SCADA/IoT Platform*. Accessed: Mar. 13, 2023. [Online]. Available: https://www.pcvuesolutions.com/index.php/pcvue-hmiscada-48584
[39] D. Dinu, Y. L. Corre, D. Khovratovich, L. Perrin, J. Großschädl, and A. Biryukov, "Triathlon of lightweight block ciphers for the Internet of Things," *J. Cryptograph. Eng.*, vol. 9, no. 3, pp. 283–302, Sep. 2019.
[40] *Programable Controllers—Part 3: Programming Languages*, Standard IEC 61131-3, 2013.
[41] Siemens, Munich, Germany. (2018). *Programming Guideline for S7-1200/1500*. [Online]. Available: https://cache.industry.siemens.com/dl/files/040/90885040/att_970576/v1/81318674_Programming_guideline_DOC_v16_en.pdf
[42] ABB, Zürich, Switzerland. (2016). *AC500 PLC and ABB ACS355 Drive Via Modbus TCP/IP With ABB Standard Library*. [Online]. Available: https://library.e.abb.com/public/e992e0980eb04dbaa8aa5d251400e8e/-EOTN118U-EN_REVA.pdf
[43] Rockwell Automation. *Modbus TCP Client Add-On Instruction for ControlLogix and CompactLogix*. Accessed: Nov. 1, 2022. [Online]. Available: https://www.myplctechnology.com/2020/03/modbus-tcp-client-add-on-instruction.html
[44] S. Kenner, R. Thaler, M. Kucera, K. Volbert, and T. Waas, "Comparison of smart grid architectures for monitoring and analyzing power grid data via Modbus and REST," *EURASIP J. Embedded Syst.*, vol. 2017, no. 1, pp. 1–13, Dec. 2017.
[45] OffSec Services Limited. *Kali Docs Official Documentation*. Accessed: Mar. 15, 2023. [Online]. Available: https://www.kali.org/docs/
[46] OffSec Services Limited. *Ettercap, Kali Documentation*. Accessed: Mar. 15, 2023. [Online]. Available: https://www.kali.org/tools/ettercap/
[47] OffSec Services Limited. *Tcpreplay, Kali Documentation*. Accessed: Mar. 15, 2023. [Online]. Available: https://www.kali.org/tools/tcpreplay/
[48] M. Cheminod, L. Durante, and A. Valenzano, "Review of security issues in industrial networks," *IEEE Trans. Ind. Informat.*, vol. 9, no. 1, pp. 277–293, Feb. 2013.
[49] Cybersecurity & Critical Infrastructure Agency. (Sep. 2021). *ICS Advisory (ICSA-21-152-01) Siemens SIMATIC S7-1200 and S7-1500 CPU families (Update A)*. [Online]. Available: https://www.cisa.gov/uscert/ics/advisories/icsa-21-152-01

FILIP KATULIĆ received the B.Sc. and M.Sc. degrees in electrical engineering from the Faculty of Electrical Engineering and Computing, University of Zagreb, in 2019 and 2021, respectively.

He is currently a Researcher with the Department of Electrical Machines, Drives and Automation, Faculty of Electrical Engineering and Computing, University of Zagreb. His research interests include industrial automation and cybersecurity in process control systems.

DAMIR SUMINA received the Dipl.Eng., M.Sc., and Ph.D. degrees in electrical engineering from the Faculty of Electrical Engineering and Computing, University of Zagreb, in 2001, 2005, and 2009, respectively.

He is currently a Professor with the Department of Electrical Machines, Drives and Automation, Faculty of Electrical Engineering and Computing, University of Zagreb. His research interests include industrial automation, cybersecurity in process control systems, control of electrical drives, and energy conversion systems.

STJEPAN GROŠ received the Dipl.Eng., M.Sc., and Ph.D. degrees in electrical engineering from the Faculty of Electrical Engineering and Computing, University of Zagreb, in 1998, 2004, and 2009, respectively.

He is currently an Assistant Professor with the Faculty of Electrical Engineering and Computing, University of Zagreb. He participates in the implementation of several EU-funded projects in the field of cyber security, focusing on the study of attacker behavior and automation using machine learning algorithms. He has published a number of papers in the field of information security, computer networks, and operating systems. His research interests include information and cyber security.

IGOR ERCEG (Member, IEEE) received the Dipl.Eng. and Ph.D. degrees in electrical engineering from the Faculty of Electrical Engineering and Computing, University of Zagreb, in 2004 and 2010, respectively.

He is currently an Associate Professor with the Department of Electric Machines, Drives, and Automation, Faculty of Electrical Engineering and Computing, University of Zagreb. His research interests include industrial automation, cybersecurity in process control systems, control of electrical drives, and energy conversion systems.