UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

EDUARDO KOCHENBORGER DUARTE

An End-to-End Defense Mechanism for Industrial Real-Time Networks

Thesis presented in partial fulfillment
of the requirements for the degree of
Master of Computer Science

Advisor: Prof. Dr. Edison Pignaton de Freitas
Coadvisor: Prof. Dr. João Cesar Netto

Porto Alegre
August 2020

CIP — CATALOGING-IN-PUBLICATION

Kochenborger Duarte, Eduardo

An End-to-End Defense Mechanism for Industrial Real-Time Networks / Eduardo Kochenborger Duarte. – Porto Alegre: PPGC da UFRGS, 2020.

65 f.: il.

Thesis (Master) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2020. Advisor: Edison Pignaton de Freitas; Coadvisor: João Cesar Netto.

1. Real-Time Industrial Networks. 2. Scalable Security Mechanism. 3. SCADA. 4. PLC. I. de Freitas, Edison Pignaton. II. Cesar Netto, João. III. Título.

“For in the sciences the authority of thousands of opinions is not worth as much as one tiny spark of reason in an individual man. Besides, the modern observations deprive all former writers of any authority, since if they had seen what we see, they would have judged as we judge.”

— GALILEO GALILEI, *Frammenti e Lettere*

ACKNOWLEDGMENTS

I would like to thank my family for all the support given to me during these long years of study. I thank all my colleagues, friends, and professors who supported and helped me in any way during this period. Specially, I would like to thank my advisors, Edison Freitas and João Netto for advising me during my studies; Tiago Dall'Agnol and Ivan Silva for helping me conduct the experiments and for being open to discussing my work.

ABSTRACT

Real-time Ethernet protocols are commonly used in control task industrial applications. Some of these applications are safety-critical, business-critical and/or confidential. A disruption in the well-functioning of the industrial plant can cause serious damage in terms of lives and costs. There are vulnerabilities caused by the clear-text communications of I/O data between the entities in the industrial (SCADA) network, which can be exploited once an attacker gains access to the network. It is imperative to ensure secure communications between all the entities to prevent unwanted attacks and leakage of confidential information. This work studies the viability of using an authenticated encryption algorithm by proposing the use of a security mechanism along with metrics to evaluate its performance. The proposed mechanism is easily deployable at a very low cost, preventing an attacker from tampering with input/output (I/O) data and preventing control information leakage. The usage of authenticated encryption as a defense mechanism in this kind of scenario had not been addressed so far. The solution can be used to implement scalable secure I/O communication that can be used alongside a secure fieldbus protocol to achieve a secure zone, according to IEC 62443. Furthermore, the solution can be used to implement a security mechanism that encompasses communications between one or more programmable logic controllers (PLCs) and the supervisory control and data acquisition (SCADA) system. Experiments have been conducted in two parts using a testbed comprised of industrial equipment as a proof of concept. The first part involves assessing the viability of the proposed security mechanism in the communications between PLCs and I/O stations, while the second part involves assessing the viability of the proposed mechanism in the communications between PLCs and the supervisory system. The experiments were designed based on the amount of I/O points of a moderate-sized application. The results show that this a viable solution to the problem in both cases, presenting scalability in terms of I/O points with processing time growing linearly and a small increase without spikes in the latency.

Keywords: Real-Time Industrial Networks. Scalable Security Mechanism. SCADA. PLC.

RESUMO

Protocolos de tempo real baseados em Ethernet são frequentemente usados em tarefas de controle em aplicações industriais. Algumas dessas aplicações são críticas em termos de segurança, críticas para os negócios e/ou confidenciais. Uma interrupção no bom funcionamento da planta industrial pode causar sérios danos em termos de vidas e custos. Existem vulnerabilidades causadas pelas comunicações em texto não criptografado dos dados de entrada/saída (E/S) entre as entidades da rede industrial, que podem ser exploradas quando um invasor obtém acesso à rede. É imperativo garantir a existência de uma comunicação segura entre todas as entidades para evitar ataques indesejados e vazamento de informações confidenciais. Este trabalho estuda a viabilidade do uso de um algoritmo de criptografia autenticado, propondo o uso de um mecanismo de segurança junto com métricas para avaliar seu desempenho. O mecanismo proposto é facilmente implantável a um custo muito baixo, impedindo que um invasor adultere os dados de E/S e impedindo o vazamento de informações de controle. O uso da criptografia autenticada como mecanismo de defesa nesse tipo de cenário não havia sido abordado até o momento. A solução pode ser usada para implementar uma comunicação de E/S segura escalável que pode ser usada juntamente com um protocolo de secure fieldbus para se obter uma zona segura, de acordo com o padrão IEC 62443. Além disso, a solução pode ser usada para implementar um mecanismo de segurança que engloba as comunicações entre um ou mais controladores lógicos programáveis (CLPs) e o sistema de supervisão (sistema SCADA). Os experimentos foram conduzidos em duas partes, usando uma plataforma de teste composta de equipamentos industriais como prova de conceito. A primeira parte envolve a avaliação da viabilidade do mecanismo de segurança proposto nas comunicações entre CLPs e estações de E/S, enquanto a segunda parte envolve a avaliação da viabilidade do mecanismo proposto nas comunicações entre CLPs e o sistema SCADA. Os experimentos foram projetados com base na quantidade de pontos de E/S de uma aplicação de tamanho moderado. Os resultados mostram que essa é uma solução viável para o problema nos dois casos, apresentando escalabilidade em termos de pontos de E/S, com o tempo de processamento crescendo linearmente e um pequeno aumento sem picos na latência.

Palavras-chave: Redes Industriais de Tempo-Real, Mecanismo de Segurança Escalável, SCADA, CLP.

LIST OF ABBREVIATIONS AND ACRONYMS

ASIC	Application-Specific Integrated Circuit
CRC	Cyclic Redundancy Check
DC	Distributed Clock
DCS	Distributed Control System
ERP	Enterprise Resource Planning
ESC	EtherCAT Slave Controller
FMMU	Fieldbus Memory Management Unit
ICT	Information and Communication Infrastructures
IDS	Intrusion Detection System
IPS	Intrusion Prevention System
I/O	Input/Output
MAC	Message Authentication Code
MES	Manufacturing Execution System
OPC	Open Platform Communications
OPC DA	Open Platform Communications Data Access
OPC UA	Open Platform Communications Unified Architecture
OS	Operating System
PDO	Program Data Object
PL	Performance Level
PLC	Programmable Logic Controller
SCADA	Supervisory Control and Data Acquisition
SIL	Safety Integrity Level

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

Modern industrial facilities are usually distributed processes, sometimes with devices placed geographically distant. It is imperative to ensure the well-functioning of the process remotely, as constant physically inspecting devices may be impractical. Every part of the system has to be monitored in real-time by the plant operators, supervising the process variables, and ensuring every device is operating within the specified parameters. The remote connection and monitoring are achieved by using control devices equipped with control networks. These control networks are called supervisory control and data acquisition (SCADA) networks (HOLM et al., 2015).

Nowadays, it is necessary for modern industries to connect to open access networks, such as the Internet. This makes the monitoring process much less troublesome, cheaper and faster (REZAI; KESHAVARZI; MORAVEJ, 2017). However, the focus of SCADA networks has been mostly on performing the desired functionalities while attending to the real-time requirements, despite the efforts in providing security to these systems (VOLKOVA et al., 2018). This fact exposes the SCADA networks to commonly found security problems, meaning they are vulnerable to attacks in which someone could easily control the entire process. These attacks might cause big troubles, unaffordable costs and life-threatening events.

SCADA networks can be used to control safety-critical processes that should not be put at risk. Disrupting the well-functioning of such systems can cause serious consequences. Different types of SCADA systems have been under attack, exposing the potential impacts of such events (ANTON et al., 2017). Real-life examples include the widely known Stuxnet, which is a computer worm that targets SCADA systems (FALLIERE; MURCHU; CHIEN, 2011) and Flame, which was used to spy, steal and delete data (MILLER; ROWE, 2012). It is important to notice that an attack can be focused on any of the entities the SCADA network is comprised of, which emphasizes the need for security in all nodes.

If an attacker gains access to the SCADA network, they could collect confidential control process data and also tamper with it, possibly causing operation problems at the plant. Obsolete standards for real-time data exchange still used in the industry, such as OPC DA, fail to define directives regarding security. The communications are in-clear, without any kind of security mechanism, which opens a door for attackers to damage the system. This means that once attackers are inside the network, they are free to do as they

please (GHOSH; SAMPALLI, 2019). The same problem shows up in the communications between other modules inside the network, which may use real-time Ethernet protocols with no security as well.

This work focuses on securing the communications related to I/O data. More specifically, for control plants that still use Open Platform Communications (OPC) Data Access (DA) as the specification to define how to transfer their real-time data and EtherCAT as the communication protocol between PLCs and I/O stations.

Although the OPC DA specification does not include any specific measures regarding security aspects, there is the option of using newer specifications, such as OPC Unified Architecture (UA). However, this will possibly incur the need of acquiring new equipment, installing and configuring it to work properly. Sometimes the cost of such operation might be too high, which makes the upgrade impossible or impracticable. Other solutions, such as implementing firewalls and anti-virus can lead to delayed delivery of data (GHOSH; SAMPALLI, 2019), which might not be acceptable for certain applications. Although the usage of OPC DA could be considered obsolete, the focus in this technology is due to the pervasive presence of old industrial equipment, as various industrial production units are kept with no software or hardware updates for years (ANTON et al., 2017).

Besides the SCADA and the PLCs, there are other entities which act upon and gather information from the industrial process, which are the I/O stations. It is also important to consider the security aspects of the communications between PLCs and I/O stations. More specifically, the security aspects of the communication protocol used to exchange data between these modules. Not every protocol is suitable for every application due to its requirements. For example, the standard Ethernet protocol is used in a wide range of application domains, but for specific applications, such as industrial control systems (ICS), it does not fulfill all the mandatory application requirements. These requirements include time-deterministic communication and real-time requirements (DECOTIGNIE, 2005). However, there are Ethernet-based protocols which comply with those requirements, e.g., PROFINET (Shi et al., 2016), EtherNet/IP (BROOKS, 2001) and EtherCAT (BECKMANN, 2004).

More specifically, this work focuses on EtherCAT-based programmable logic controllers (PLCs), which is one of the commonly used real-time Ethernet protocols due to its performance and scalability features. EtherCAT is able to address an increasing number of slaves with a minimal overhead increase, which is one of the reasons it is widely used

in the industry. These devices are susceptible to various security risks due to the cleartext (without encryption) communication, which is not a disadvantage exclusive to EtherCAT (PLIATSIOS et al., 2020). Despite the ability of EtherCAT and other real-time protocols to comply with real-time requirements, there is still a small number of approaches that handle security aspects in this segment. With the widespread usage of networked systems solutions in a great variety of domains, security became a concern with highlighted relevance. This is also true in industrial networked environments, as security vulnerabilities may compromise the field operation, causing important losses if explored by attackers (AKPINAR; OZCELIK, 2018). However, the adoption of security mechanisms is always associated with overhead imposition. Considering industrial networks, this overhead may potentially degrade the performance of the overall system in a way that it simply stops working, as it is not able to meet its timing requirements. This could lead to potentially delayed system reactions and unpredictable behavior. A study evaluating the performance of real-time Ethernet protocols is presented in (ROBERT et al., 2012).

Summarizing, there are two different components of the SCADA network that must be secured: the communication of input/output (I/O) data between I/O stations and programmable logic controllers (PLCs); and the communications between PLCs and the SCADA system. Each one of these components has its own performance and scalability considerations. This work proposes a defense mechanism based on authenticated encryption that encompasses both ends and everything in between, while presenting, analyzing and evaluating thoroughly the performance of the communications between PLCs and the SCADA system, and between PLCs and I/O stations. The proposed method has been designed considering how to optimize processing times, ensuring scalability and real-time properties. It takes into account the performance advantages considering the whole communication path, from one end to the other, i.e. from the I/O stations to the SCADA system.

While there are works that study the usage of authenticated-encryption on fieldbus communications, such as (WIECZOREK et al., 2012), to the best of the authors knowledge, there is no work which takes into account the scalability aspects related to the number of I/O points in the application. Observing this landscape, this work focuses on the security analysis of the EtherCAT protocol used in PLCs and on the communications between PLCs and SCADA system, proposing a software-based security mechanism. The mechanism can be implemented in an industrial plant without actually acquiring new devices, reducing the cost greatly. This solution adds a layer of security in the I/O exchange

by encrypting this data, which provides scalable secure I/O communication for applications using EtherCAT and OPC-DA. If integrated with a security fieldbus protocol to secure the communication such as (WIECZOREK et al., 2012), the proposed solution will provide a robust defense mechanism that achieves a secure zone according to IEC 62443 (SHAABAN; KRISTEN; SCHMITTNER, 2018). This is achieved without sacrificing the real-time features or the performance, even on applications with hundreds or thousands I/O points.

The proposed method represents a novel way of implementing secure I/O communication while taking advantage of specific application aspects which lower immensely the overhead imposed by the mechanism. The mechanism can also be used to secure just the EtherCAT communications or the data exchange between PLCs and SCADA system, although the greatest performance advantage shows up when using it on both cases.

While the method proposed in this work is not enough on its own to ensure fully secure communication, it is important to notice that there are alternatives for securing the communication besides I/O data. Thus, the proposed solution represents a novel way of implementing a security mechanism that integrates fully with EtherCAT-OPC-DA-based applications. The same method can be applied to other types of data as well, e.g., securing the distributed clock frames. Moreover, it can also instigate further EtherCAT-focused (and other real-time Ethernet protocols) security studies in the future, as well as studies regarding the current security standards and general SCADA network security. It is also important to emphasize that the actual cryptographic function is not part of the contribution of this work, as it can be updated or changed as desired, given that the real-time requirements remain fulfilled.

This work also addresses points stated by safety standards, such as IEC 61508. The standard states that if a malevolent or unauthorized action that can lead to hazards is reasonably foreseeable, then a security threat analysis should be carried out. It is important to emphasize that the increased security is also a requirement to comply with the safety standard IEC 61784-3, which also references IEC 61508 (KANAMARU, 2017).

Summarizing, the main contributions of this work can be stated as:

- Designing, proposing and structuring a novel software-based defense mechanism which takes advantage of application structure to provide secure I/O communication from I/O stations to the SCADA system, increasing security with very low cost and time;
- Implementing and determining if an authenticated encryption software-based so-

lution is suitable for practical applications, taking a step towards the guidelines of IEC 62443.

- Shedding light on security questions regarding SCADA networks and automation in general, while motivating further security studies in this area and instigating the use of low-cost, easily-deployable and viable security measures.

The rest of this paper is organized as follows: Section II presents an overview of background concepts. Section III presents the problem formulation and the proposed approach to address it is described in Section IV. Experiments and results are presented and discussed in Section V. Section VI discusses related work and, concluding the paper, Section VII brings discussions of possible future work directions.

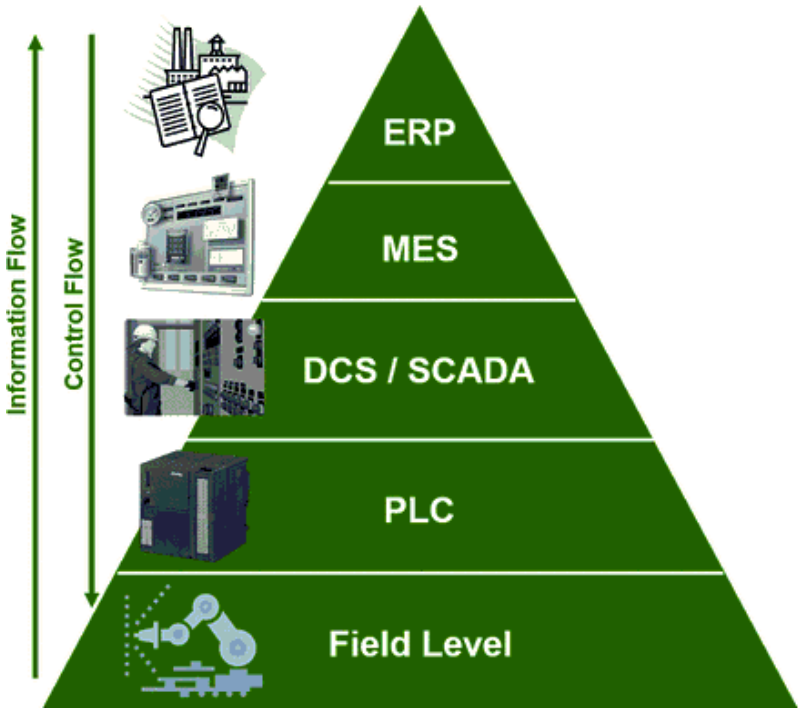
2 BACKGROUND AND RELATED WORK

2.1 Background

2.1.1 Automation

The automation process is commonly represented by a hierarchy of levels known as the automation pyramid. This hierarchy represents the gathering of information, i.e. the information flow and the control flow. Usually, the information flow is represented in a bottom-up fashion, with the information from the field devices rising through the hierarchy up until the top. Similarly, the control flow is usually represented in a top-down fashion, with the decisions coming from the top and reaching the field devices, at the bottom level. The automation pyramid serves as a general design pattern for creating information and communication infrastructures (ICT) for the industry (HOFFMANN et al., 2016). A representation of the pyramid is shown in Figure 2.1.

Figure 2.1: Automation Pyramid



Source: (HOFFMANN et al., 2016)

Starting from the bottom, the first level is the field level. This is where the field

devices, such as actuators and sensors are located. They are responsible for the physical actions and monitoring, and are most likely distributed in a large geographical area on the production floor. These devices do not have any kind of "intelligence" as they are controlled by the devices on the very next upper level. The entities which control the field devices are called programmable logic controllers (PLCs).

On the next level, a distributed control system (DCS) or supervisory control and data acquisition (SCADA) system will receive the information gathered by the PLCs. This enables the whole process to be monitored at once, ensuring the well-functioning and avoiding critical problems, as well as orchestrating one or more PLCs to perform a task. On the next level, there is the Manufacturing Execution System (MES) level, which gathers information from the levels below, monitoring the entire manufacturing process. At the top level, there is the Enterprise Resource Planning (ERP), which is based on historical data, such as product lists, processes, quantities. It also provides decisions for the entire plant and performs long-term planning in terms of human and material resources (HOFFMANN et al., 2016).

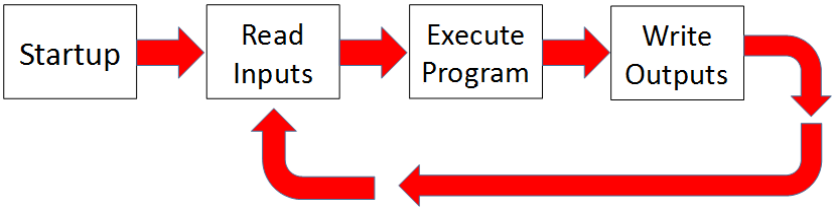
2.1.2 PLCs

PLCs are equipment designed to be used in automation and control tasks in industrial environments. They are specially designed to withstand temperature, humidity, physical vibrations, noise and electromagnetic phenomena. The control task which calculates the appropriate response based on the process's current state is implemented on a "user program". The program is programmed on the PLC using a development environment (BOLTON, 2015).

The module responsible for processing the control logic is not the same as the modules responsible for obtaining information from the sensors or generating the desired outputs. Although a PLC can be used as the only node of the system, in this work, the focus is on distributed systems, where each module is a separate entity designed for a task. The PLCs control the lower level (field level) devices according to the user program that has been downloaded to them. This means that the field devices will serve as both inputs and outputs of the system. They will receive information from the current state of the process and act upon it, outputting the appropriate response calculated by the user program (BOLTON, 2015). This is shown in Figure 2.2.

Due to the focus on distributed systems, there has to be some kind of communi-

Figure 2.2: PLC scan cycle



Source: Author

cation between the entities to exchange essential process information. The I/O stations act as an interface between the external world, where the actual process happens, and the control application, which is executed on the PLCs. Thus, the I/O stations have to relay the data they collected from the process to the PLCs in order to calculate the appropriate response. For this to be possible, a communication protocol between the modules is necessary.

2.1.3 EtherCAT

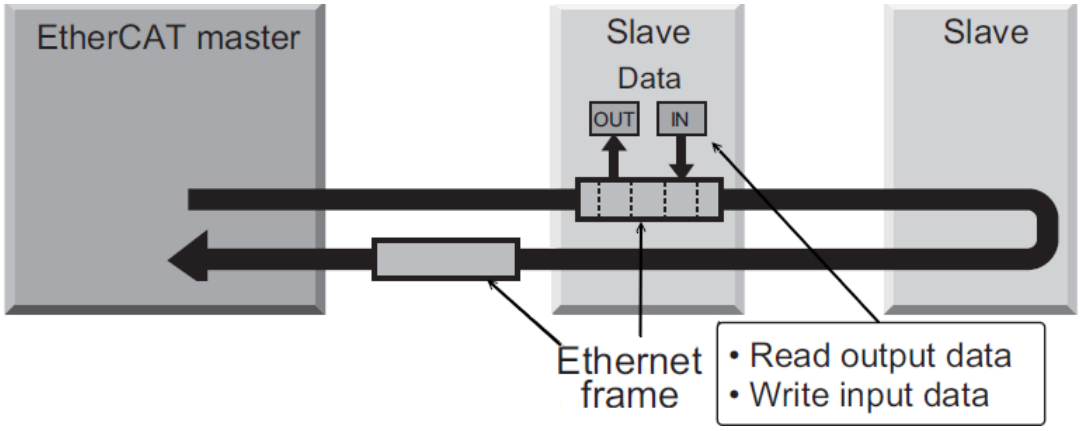
The PLCs themselves communicate with I/O stations by the means of an industrial communication protocol. However, not every protocol is suitable for industrial applications due to their specific real-time requirements. Therefore only protocols that comply with these requirements are suitable, such as PROFINET (Shi et al., 2016), EtherNet/IP (BROOKS, 2001) or EtherCAT (BECKMANN, 2004). The payload of the exchanged messages contains the information provided by each I/O station's port. This communication happens cyclically, enabling each PLC to periodically assess the current state of the process, through reading its inputs, and to act upon it, through writing its outputs. In terms of the control task implemented on the PLC, this communication happens transparently. The user program does not have to worry about dealing with sending and receiving EtherCAT frames. The user program will simply have access to specific variables which will be mapped to the inputs and outputs of the corresponding I/O stations. Similarly to the I/O updates, the user program will also be executed cyclically, with the cycle being a configuration related to the specific control task being performed.

EtherCAT complies with real-time requirements and aims to maximize the performance by maximizing the utilization of Ethernet's full-duplex communication. It uses

a master/slave communications model, where the master sends Ethernet frames and the slaves read/insert data accordingly. The frames are processed "on the fly" by each slave, effectively reading/writing data as the frame passes by. This introduces only a constant delay, independent of the size of the packet. The frame passes through all slaves and is reflected at the end of each network segment, being sent back to the master (BECKMANN, 2004), (Prytz, 2008). Also, the only node able to send frames is the master, preventing unpredictable behavior and ensuring consistency. This makes EtherCAT achieve remarkable performance in terms of latencies and scalability, which makes it a very interesting option.

The behavior of EtherCAT communications can be better understood with the aid of Figure 2.3, where a slave is shown retrieving output values sent by the master and writing its input values on the frame.

Figure 2.3: EtherCAT Master/Slave Communication



Source: Adapted from (OMRON Automation Pvt Ltd, 2016)

Each frame can contain one or more datagrams, which represent different actions provided by the master. These datagrams may be commands to read, to write, to read-write, or to access one or more slave devices. The network is able to expand up to 65,535 devices. Each slave has a Fieldbus Memory Management Unit (FMMU) which allows the logical addresses to be mapped to the slaves' physical addresses. This allows the EtherCAT master to be able to address several slaves at once, which is particularly important in motion control applications (NGUYEN; JEON, 2016). The registers mapped by the FMMU in each slave are called process data objects (PDOs), and there is a bidirectional cyclical exchange of PDO data between the master and the slaves (BECKMANN, 2004).

2.1.4 SCADA System

A real industrial application will most likely be comprised of several different devices, including PLCs, perhaps even from different manufacturers. Each one of these PLCs will control a set of I/O stations related to one specific part of the control process. Because these devices are usually far apart from each other, what would make monitoring all of them a tedious errand, they are all interconnected by a network. This interconnection enables the entire process to be monitored by operators using workstations. This is what is represented on the automation pyramid on the third level (DCS/SCADA), in Figure 2.1.

The workstations provide the operators with real-time information relevant to the process, ensuring the well-functioning by continuously polling field information and, eventually, sending control commands. These workstations are typically able to generate alarms in case the monitored parameters deviate from the expected outcomes, keep historical data to allow an analysis over time, among other functionalities. Typically, this network is used to communicate messages related to the control process to master and slave field devices.

The application needs all devices, even the ones of different manufacturers, to be fully and seamlessly integrated to function as intended. This means that there has to be some kind of "common language" to allow data to be exchanged between these devices. In automation, the challenges concerning exchanging information between different devices have been standardized by the OPC (Open Platform Communications) Foundation. It defines how data should be transmitted from a data source to a data sink, in a standard way. OPC is, nowadays, supported by most SCADA systems (ABBAS; MOHAMED, 2015), and it draws a line between hardware providers and software developers. This standardization allows interoperability between devices of different manufacturers. Furthermore, this allows SCADA developers to focus on their area of expertise, without having to worry about developing and maintaining communication drivers and other device-dependent components.

OPC DA does not define any specific security measures in its specification. Security can be configured to a certain extent, but as it is not integrated into the specification, this can be particularly troublesome. Some environments may not support security configurations and some applications cannot be configured to support all security options (HUNKAR,), which means the security options are really rudimentary.

To address several limitations of the Classic OPC, in 2008, OPC Unified Architec-

ture has been released. One of these addressed limitations is related to security. OPC-UA is concerned with authentication, authorization, encryption, data integrity, and auditing of client-server sessions (OPC Foundation,). However, while there are high-end devices that support OPC-UA, there are several applications that continue to use Classic OPC. Most notably, the manufacturer which designed the PLCs used on this work has only recently started to conduct tests regarding the usage of OPC-UA. Furthermore, the authors of (ANTON et al., 2017) emphasize the common presence of legacy devices in control plants, which might be years or decades with little or no updates. Therefore, it is safe to assume that the security concerns involving Classic OPC are not negligible, and this is the reason why this work focuses on it. While there are still legacy devices operating with less-than-ideal security features, it is important to try to achieve security by other, more feasible and affordable, means.

2.1.5 Standards

There have been several efforts with the objective of aligning both safety and security standards, as shown in Table 2.1. IEC 61508 classifies safety levels in "Safety Integrity Levels" (SIL), which represent risk reduction as the levels go up. Some of the most general requirements of IEC 61508 have been adapted specifically to industrial machinery on the IEC 62061. Safety levels may also be classified by means of "Performance Levels" (PL), according to ISO 13849.

IEC 61784-3 defines standards for industrial safety-related networking while referencing IEC 61508, which also defines as necessary for safety applications to address the security threats and risk assessments. This shows that there is a strong connection between safety and security aspects, while emphasizing even more the importance of securing the communications due to the possible risks imposed by an attacker in terms of cost and lives.

IEC 62443 and IEC 61784-4 address general security concerns and security related to specific technologies, respectively. The approach is to break the system into smaller parts, called "zones", which communicate with each other through "conduits". Because the two main standards for safety and security are applicable to a wide range of domains, there are standards which focus on the control and automation domain. These standards are called IEC 63069 and IEC 63074 and they focus on different approaches on how to bridge safety and security harmoniously.

Table 2.1: Security and safety aspects of some well-known standards

Standard	Safety	Security
IEC 61508	X	
IEC 62061	X	
ISO 13849	X	
IEC 62443		X
IEC 63069	X	X
IEC 63074	X	X
IEC 61784-3	X	
IEC 61784-4		X

Source: Author

2.2 Related Work

To ensure that the control process works as intended, there are two main points of view to look for: protecting the system against accidental failures, which can be assessed in terms of probabilities and protecting the system against intentional attacks. Such aspects are addressed by safety and security domains, respectively. This work focuses mostly on the security aspect, although it is important to emphasize that security is a requirement according to various safety standards, which means that there is a relation between the two parts. More specifically, in terms of this work, the security aspect is analyzed in two parts as well: the internal bus and the communications with the SCADA system.

The authors in (SHAABAN; KRISTEN; SCHMITTNER, 2018) follow the guidelines of IEC 62443 and IEC 61784-4 to define the automation control devices and I/O stations as one zone, according to the standard's recommendations. Using a scalable defense mechanism to secure I/O data, as proposed in this work, alongside a protocol to secure regular communications, the zone which contains the automation device and I/O stations will be fully secured.

Regarding the protocols used in the communications between PLCs and I/O stations, there are many studies which have discussed security aspects of Ethernet and many other commonly used protocols. Yet, a much smaller number of studies have discussed the security aspects of real-time protocols, such as the protocol focused in this work, the EtherCAT. Since EtherCAT is an Ethernet-based protocol, it also suffers from the same vulnerabilities as standard Ethernet (GRANAT; HÖFKEN; SCHUBA, 2017). The protocols which target the control automation domain are not the only ones to suffer from vul-

nerabilities. Other buses used in vehicular applications, such as CAN, LIN, and Flexray do not provide any kind of security mechanism either (WIECZOREK et al., 2012). Generally speaking, there are two main points to be considered: providing the means to achieve the necessary security, and taking into consideration the performance requirements of the system.

Among the currently available safety standards, there are PROFIsafe, CIP-safety, CC-Link Safety, PowerLink Safety, and TwinSAFE. They use black channel principles to achieve safety features using a standard, not safety-related network. They also encapsulate data relevant to safety, such as cyclic redundancy check (CRC), timestamps, etc. While this may provide some rudimentary security as well, they rely on underlying security infrastructure, such as MACsec, IPsec and TLS/SSL. This means they require specific equipment to be installed to be fully functional (SATO et al., 2015). As previously mentioned, one of the goals of this work is to provide a security mechanism that avoids the need of new equipment. Furthermore, an analysis shown in (ÅKERBERG; BJÖRKMAN, 2009) shows that it is possible to manipulate safety-related information on PROFIsafe without any of the protocol's safety mechanisms detecting it. This shows that the rudimentary security provided by the safety standards might not be enough on its own.

Specifically about EtherCAT, an analysis of the protocol's vulnerabilities has been performed, focusing on the EtherCAT communication principles case (AKPINAR; OZCELIK, 2018). Three different types of attack have been evaluated by the authors: MAC spoofing, data injection and slave address attack vectors, reviewing in detail each case. The authors propose an EtherCAT preprocessor created for Snort, an intrusion detection/prevention system (IDS/IPS). However, while the solution may be effective in the scenarios used to validate their proposal, EtherCAT still lacks a mechanism to provide secure communication between master and slaves.

The authors in (WIECZOREK et al., 2012) established a security protocol for securing fieldbus communications in security-critical applications using a stream cipher and a message authentication code (MAC). They have shown through a proof-of-concept implementation using EtherCAT that their security protocol is viable and causes minor overhead. However, this work does not take into account the scalability aspect, such as the increasing number of I/O points that a real application may have. Therefore, there is no way to affirm that the proposed solution is able to cope with real applications, which may be comprised of thousands of I/O points.

Regarding securing the communications between PLCs and SCADA system, as

previously mentioned, one of the possible ways of achieving in control networks is by using OPC UA, which allows signed and encrypted communications. The authors in (HANNELIUS; SALMENPERA; KUIKKA, 2008) highlight a roadmap of possible ways to adapt current systems to take advantage of the features introduced by this possibility. Similarly, in (CHUANYING; HE; ZHIHONG, 2012), the authors establishes a roadmap for the migration to OPC UA by proposing different alternatives. However, these works do not seem to assess if there is any performance drawback or how well these alternatives scale. Therefore, it is hard to predict whether these solutions can be used on a real application.

Specifically about SCADA systems security, in (NICHOLSON et al., 2012) and (IGURE; LAUGHTER; WILLIAMS, 2006), the authors discuss threats, risks, and vulnerabilities, which can be useful to help understand the security issues as a whole. In (SOMMESTAD; ERICSSON; NORDLANDER, 2010), the authors make an extensive comparison of guidelines and standards related to SCADA security, even comparing it with the international standard ISO/IEC 27002. According to the authors, the two most common countermeasures suggested by the standards are authentication and cryptography, respectively. Besides, the paper also describes spoofing, replay/interception/modification of data and information gathering as the third, fourth and fifth most commonly mentioned threats, respectively. These conclusions serve as great motivation to pursue a defensive mechanism which focuses on these aspects, given their importance.

An extensive survey about security issues and challenges in SCADA networks has been reported in (GHOSH; SAMPALLI, 2019). The authors provided a classification of attacks based on security requirements while comparing standards and security schemes proposed to overcome the weaknesses of the standards. Among the compared standards, the authors discuss crypto-suite standards, such as IEC 62210, IEC 62351, and AGA-12. The authors argue that AGA-12 is the best of the crypto-suite standards. However, they disclaim that its cryptographic protection relies on algorithms with very high computational cost, which is very unlike to scale in big control plant applications. However, there is no data to carefully analyze how well an application using these standards will scale. This is a huge obstacle when trying to compare the achieved results with current known standards.

In (ANTON et al., 2017), the authors present a history of attacks and exploitations directed at industrial systems. They present a graph representing the amount of PLC-exploitations over the years, which shows that these attacks appear to be growing as time

passes. Furthermore, the authors present their findings regarding field devices public addressable, while also stating that many industrial production units are kept decades with little or no software updates. All of this emphasizes the importance of studying security aspects of SCADA systems.

Another analysis of SCADA-related incidents is performed at (PLIATSIOS et al., 2020). The authors first explain in great detail the SCADA architecture and commonly used field-bus protocols before presenting security vulnerabilities for several SCADA protocols. Among the undesirable security vulnerabilities, the authors mention the unencrypted data exchange. It is remarkable that the majority of the protocols have various vulnerabilities. There is also an extensive survey on the proposed security solutions, with most of them involving traffic classification and attack detection approaches. Some works, such as (SHAHZAD et al., 2015) and (FOVINO et al., 2009) attempt to make use of encryption techniques to provide more security for Modbus protocol.

Another point of view in terms of security is presented in (VOLKOVA et al., 2018), where the authors compare various communication protocols in terms of the security standard IEC 62351. Interestingly, the authors conclude that the most important requirements of control systems involve the usage of legacy equipment and considerations regarding 24/7 operation and its real-time requirements. This emphasizes that even though certain solutions might be considered obsolete by now, in real applications they might still be in use. Furthermore, the authors also claim that future works regarding testing the performance impacts of applying IEC 62351 must be conducted to ensure it is viable, as well as measuring the impact of other proposed mechanisms.

Concerning the choice of which security mechanism to implement, there have been works on multiple different types of defense mechanisms. For example, the authors in (SONG; KIM; KIM, 2016) proposed an analysis based on the time in which messages arrive at the receiving end: certain anomalies would make the presence of an attacker quite evident. Considering the communications between PLCs and SCADA should occur periodically, it is possible that this is a different feasible approach.

The scalability is not a straightforward matter to solve. It is important to notice that it is a challenge to provide a mechanism that can ensure security and be lightweight at the same time, because depending on the security mechanism, it may incur an important performance degradation. There is most likely a trade-off between how lightweight and how secure an encryption system is. Using cryptography as a security mechanism is possible, but the two main points previously mentioned must be ensured. An exten-

sive survey of lightweight cryptography algorithms has been presented in (BIRYUKOV; PERRIN, 2017). It also presents authenticated lightweight encryption schemes, which provide an additional layer of protection against replay attacks, for example. It also discusses trade-offs commonly found on these algorithms, most notably performance and security, which are basically the two main points of concern. This kind of mechanism may also be applied to SCADA networks and industrial communication protocols. However, there is no cryptographic algorithm specifically designed to work with SCADA or control automation tasks. Therefore, the implemented algorithm must be carefully chosen so that it does not hinder the system's real-time properties and performance.

To properly evaluate and compare the performance of the proposed solution, the behavioural analysis proposed by (ROBERT et al., 2012) can be extremely useful to facilitate not only the comparison with other Ethernet-based protocols, but to evaluate the performance of the mechanism. The authors proposed metrics that can be adapted to provide an insight on the performance of the proposed solution.

It is also important to know if it is even possible to protect the network from one of the main threats, the replay attacks. In (NARULA; HUMPHREYS, 2017), the authors establish a set of conditions that must be fulfilled in order to make the secure clock synchronization possible. The conditions are developed, proposed and proved based on a generic system model. It has been proved that one-way clock synchronization protocols are vulnerable to replay attacks. Although this is a different scenario, the messages generated by the master during I/O updating share similar characteristics with the ones generated during the clock synchronization process. More specifically, due to the inherent ring topology of EtherCAT networks, it is a two-way protocol. Therefore, the same conditions can be applied to determine whether it is possible or not to protect the network from replay attacks, and to successfully design a protective mechanism.

There are other means of protecting the system other than cryptography. In (Huang et al., 2019), the authors propose a different approach to secure industrial cyber-physical systems: a security decision-making approach based on stochastic game model. The authors mention that existing countermeasures, such as encryption, for instance, lack the decision-making mechanism to defend against advanced persistent threats. The proposal is very promising, but there can be difficulties specifying the model parameters, which might delay the deployment of the countermeasures.

The authors of (NAZIR; PATEL; PATEL, 2017) have conducted an extensive research regarding different approaches of evaluating the efficacy of a SCADA system

against cyber-attacks. Various techniques to expose hidden vulnerabilities and to assess the degree of protection are presented. Naturally, securing the SCADA system is desirable as well, and these techniques can be used to evaluate if any proposed mechanism is effective.

A general summary of most related works used in this research is presented in Table 2.2. As it can be seen, although several works address individual points, none of them address the end-to-end security aspect of the communications, while evaluating performance and scalability metrics. For example, (ROBERT et al., 2012) does not take into account the security aspect applied to real-time protocols. In (PLIATSIOS et al., 2020) there is an extensive discussion regarding challenges and security aspects, but no performance evaluation in terms of scalability. In (VOLKOVA et al., 2018), there are numbers quantifying the performance impact on the Modbus protocol, but since it is a different standard, it is not directly comparable to EtherCAT or OPC DA. The circumstances present themselves as obstacles when trying to compare directly the results achieved at this work. The bibliographical research in this work has been conducted by using the relevant combinations of the column cells presented in Table 2.2. For instance, "EtherCAT performance evaluation" or "SCADA Networks Security". The search terms were used mainly on three digital bibliographic libraries/databases: Google Scholar, IEEE Xplore, and ACM Digital Library.

Table 2.2: Summary of Related Works

Author	**Security**	**Scalability**	**Discusses/Uses Standards**	**Performance Evaluation**	**EtherCAT**	**SCADA Networks**
Shaaban	X		X			
Wieczorek	X			X	X	
Sato	X		X		X	X
Åkerberg	X		X			
Akpinar	X				X	
Hannelius	X					X
Chuanying	X					X
Nicholson	X					X
Igure	X					X
Sommestad	X		X			X
Ghosh	X	X				X
Anton	X				X	X
Pliatsios	X		X		X	X
Shahzad	X					X
Fovino	X			X		X
Volkova	X		X			X
Song	X			X		
Biryukov	X			X		
Robert			X	X	X	
Narula	X				X	
Huang	X					X
Nazir	X		X			X
(this work)	X	X	X	X	X	X

Source: Author

3 PROBLEM STATEMENT

The problem studied in this work comprises securing two separate segments of the communication stack, considering the path from I/O information captured by the field devices to the SCADA system. There are two main segments: the first responsible for the communications between PLC and I/O stations, and the second responsible for the communications between PLCs and the SCADA system. Naturally, these two segments are part of the application and are not intended to function separately, but they have been subdivided to facilitate the study conducted in this work.

This chapter is subdivided as follows: the first section contains general aspects of the problem; the second part contains the aspects related to the communication between PLCs and I/O stations; the third part presents the aspects related to the communication between PLCs and the SCADA application; finally, the fourth part summarizes the problem taking into consideration aspects of both communication segments.

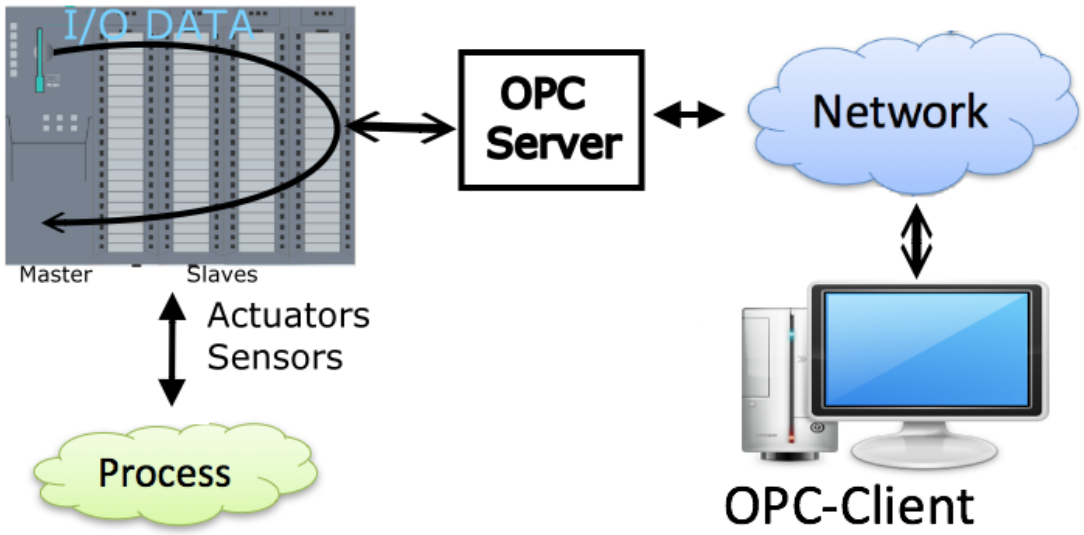
3.1 General Aspects

While achieving security in the communication of I/O would not be enough to guarantee fully secure communication, it is a step towards that, and therefore it is of interest. A solution that would secure the entire path of the I/O data, from the I/O stations to the SCADA system would be the ideal one, and that is the reason this study focuses on the two previously mentioned segments. However, this approach must take into consideration the performance and scalability aspects of the PLCs, the I/O stations and the SCADA system. If any of the nodes are not capable of handling the overhead introduced the proposed mechanism, it cannot be used. This problem can be better understood by the aid of Figure 3.2, which provides an overview of the entire system.

As shown in Figure 3.2, there is an exchange of information between the PLC itself, which is the master, and the I/O stations, which are the slaves. Note that an application might be composed of several PLCs, each one in a different bus. The PLCs will then be able to send the information to the SCADA software through the OPC Server. The OPC Server is connected to the same network as the SCADA software, which is the OPC Client.

The attacker is assumed to be able to enter the network and capture packets in both cases, which can be either altered or sent in a later moment as a replay attack. The

Figure 3.1: System overview



Source: Adapted from halvorsen.blog

attacker cannot read or alter any of the devices' memory to obtain any kind of sensitive information as the access to the devices is protected with a username/password scheme.

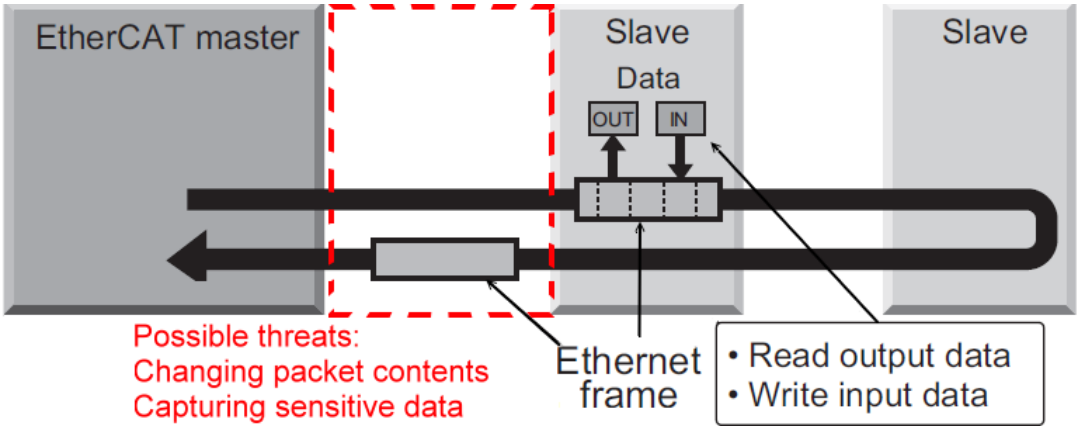
3.2 Communication between PLCs and I/O Stations

Regarding the communications between master and slaves, in this work, the focus is on EtherCAT-based PLCs. Since the communications are made using EtherCAT, this data is susceptible to the same vulnerabilities and threats as the protocol itself. This study focuses specifically on the security aspects of communications related to the I/O updating and its scalability. For securing the communications in its entirety, i.e. other data exchanges unrelated to I/O, the proposed solution can be used alongside a security protocol for fieldbuses, such as the one proposed at (WIECZOREK et al., 2012).

If a malicious user gains access to the internal bus, it is possible to change the contents of the packets, possibly causing severe damage to critical systems and even putting lives in danger. Since the protocol itself does not contain any security mechanisms, all the data is transmitted as cleartext. The data transmitted without encryption means the attacker can also obtain sensitive and/or confidential information about the industrial control process, which might pose a threat. Besides, it also enables the attacker to capture a frame passing through the bus and to attempt to create a replay attack by resending some

old frame, as there are no security schemes to detect such attack. The vulnerability in the EtherCAT communications is shown in Figure 3.2.

Figure 3.2: Vulnerability in EtherCAT master/slave communication



Source: Adapted from (OMRON Automation Pvt Ltd, 2016)

One way of preventing such malicious interferences is ensuring that communications between the master and the slaves are encrypted through a cryptographic algorithm. Furthermore, ensuring that the messages are properly authenticated so that each side of the communication will only accept incoming packets from the other side. In the scope of this work, a "message" is the I/O data exchanged between the master and the slaves. Even though there are other ways of damaging the control system, such as altering EtherCAT commands inside a telegram, the focus of this work is on securing the I/O data itself and to show that it can be done through a software-based solution with low cost, easy deployment and without any major performance loss. The low cost and easy deployability are important because they enable industrial plants equipped with old equipment that use, for example, OPC DA, to implement security mechanisms without a big investment. The reasoning is that the plants which still use legacy equipment are most likely trying to avoid the costs of acquiring new equipment. Therefore, if the proposed solution required a considerable financial investment, it would most likely not reach the target applications.

Due to its characteristics, the EtherCAT traffic is dealt with by hardware (commercial solutions), which means there is no way to implement security on the EtherCAT commands without acquiring/designing new hardware, making the solution not as low-cost and not as easily deployable. In this case, further studies would be required to correctly assess the viability of hardware implementation. However, even in the more specific case of securing the I/O data only, there are multiple performance considerations to be taken

into account when dealing with control systems, which have hard real-time requirements.

One of the challenges of using an approach based on authenticated encryption is that the scale of the applications PLCs are used for can vary widely. That means certain applications are more strict concerning cycle times, like motion control. Others might have hundreds of I/O points (or even more), like a beer brewery automation process.

More specifically, while it is important to achieve a certain degree of security to mitigate this security threat, it is also important not to compromise the determinism of the system or cause excessive overhead in ways that could compromise its well-functioning. It is possible that a solution using encryption/authentication might not be suitable for every kind of application, depending on how much it impacts the performance of both the EtherCAT master and slaves.

3.3 Communication between PLCs and the SCADA Application

Regarding the data communicated from the PLCs to the SCADA (or vice-versa), there is a similar problem with the lack of security. This data contains information related to inputs and/or outputs and is transmitted as cleartext, without encryption. In the event that attackers succeed in entering the network, they can capture or tamper with the packets. There are many possible ways they could interfere with the well-functioning of the process in both segments, which could lead to serious damage and confidential data leakage.

One possible solution to the problem would be to insert several external cryptographic modules in the middle of the communication stack, which would ensure fully encrypted communication between all the nodes of the network. This would mean that the PLCs, the I/O stations and the SCADA system would not have to worry about the overhead associated with encoding or decoding data. However, this may not necessarily be a feasible option due to difficulties accessing the nodes and the possible costs in acquiring new equipment. This work aims precisely to overcome difficulties such as replacing and/or acquiring new hardware (changing the PLCs for a more robust model or acquiring external cryptographic modules).

In terms of an attacker gaining access to the communications between SCADA and PLCs, besides possibly obtaining confidential information, there are mainly two cases in which the attacker could cause damage: the first occurs if the attacker decides to change an input variable being read by the SCADA system from a PLC (either by modifying

an ongoing packet or by injecting a new one); and the second, if the attacker decides to change an output value being sent by the SCADA system to a PLC (once again, by modifying or injecting a packet). For example, altering values read from a sensor or changing the desired output of an actuator.

3.4 Problem Summarization

Therefore, the problem to be addressed is how to provide the necessary security only by software, implementing it in the application layer, because it is much easier to deploy software updates in a plant than to install new hardware. In fact, it is not only much easier but much cheaper and faster too. Thus, a software-based solution is more appealing for both segments.

It is absolutely critical to take into consideration the performance of all entities in the application. The focus of this work is on proposing an approach to use authenticated encryption to secure I/O data, while determining how the proposed mechanism will affect the performance of each entity. This will allow the question regarding the viability of authenticated encryption in this context to be answered for all entities. The entities in question are: the I/O stations, the PLCs and the SCADA application. Therefore, the research questions proposed by this work can be summarized as:

- How much slower will an EtherCAT slave be when using a security mechanism based on authenticated encryption?
- How scalable is this solution on different scenarios, with different quantities of I/O points, when analyzing the master's point of view? How much slower will the master be?
- How scalable is this solution in terms of I/O points when analyzing the SCADA application's point of view? Is the solution viable in real applications, with hundreds or thousands of I/O points?

4 PROPOSED APPROACH

This chapter is subdivided in three sections based on the partitioning of the problem. First, the common aspects of both parts of the problem are presented. Then, the aspects related to the communication between PLCs and I/O stations. Finally, in the last section, the aspects related to the communication between the PLCs and the SCADA application are presented.

4.1 Common Aspects

Having the problem properly stated, it is possible to look into finding solutions to defend the system. The main objective is to prevent malicious tampering of the packets contents and disclosure of confidential information. As previously mentioned, one way of defending the system is by using an authenticated encryption scheme. There are studies in the area of cryptography that focus on algorithms, suitable for hardware and/or software implementations, for resource-constrained devices, typically targeting one of a list of metrics. Some of these metrics are memory consumption, throughput, and code size (ENGELS et al., 2011).

Out of the proposed authenticated encryption algorithms, Hummingbird-2 (ENGELS et al., 2011), which is a symmetric-key algorithm, was chosen for the implementation of the proposed solution due to its properties, such as having authentication and not increasing the size of data upon encryption. Maintaining the same number of bits after the encryption is particularly interesting in terms of scalability. Besides, it will also help to isolate the actual overhead from the possible side effects that could be caused by increasing the network traffic. Certainly, other algorithms can also be used, but for the sake of providing a proof of concept, this work focuses on one algorithm. Future works regarding comparing the usage of different algorithms can also be conducted.

Hummingbird-2 is an authenticated encryption algorithm which works with a 128-bit encryption key and a 64-bit initialization vector. It operates on 16-bit blocks, inheriting properties from both stream and block cipher categories. The encryption and decryption are entirely based on 16-bit operations, such as exclusive or between words, addition and subtraction modulo 65536, and a nonlinear mixing function.

The encryption and decryption, are based on a 16-bit keyed permutation function $WD16$ and its inverse, respectively. Let $S(x)$ be the computation of S-boxes $S_1...S_4$

defined in (ENGELS et al., 2011), $L(X)$ be the linear transformation described below, $\ll$ be the left circular rotation and $\oplus$ be the exclusive or operation. The $WD16$ function, which is used for encryption, is defined as follows:

$$S(x) = S_1(x_0) \mid S_2(x_1) \mid S_3(x_2) \mid S_4(x_3)$$
$$L(X) = x \oplus (x \ll 6) \oplus (x \ll 10) f(x) = L(S(x))$$
$$WD16(x, a, b, c, d) = f(f(f(f(x \oplus a) \oplus b) \oplus c) \oplus d)$$

And its inverse, $WD16^{-1}$, which is used for decryption, is defined as follows, using the inverses of each function (CHAI; GONG, 2012):

$$S^{-1}(x) = S_1^{-1}(x_0) \mid S_2^{-1}(x_1) \mid S_3^{-1}(x_2) \mid S_4^{-1}(x_3)$$
$$L^{-1}(X) = x \oplus (x \ll 2) \oplus (x \ll 4) \oplus (x \ll 12) \oplus (x \ll 14)$$
$$f^{-1}(x) = L^{-1}(S^{-1}(x))$$
$$WD16^{-1}(y, a, b, c, d) = f^{-1}(f^{-1}(f^{-1}(f^{-1}(y) \oplus d) \oplus c) \oplus b) \oplus a$$

The presented steps are only part of the computation executed during the encryption or decryption process. For more details about the algorithm and the step by step encryption/decryption and authentication process, refer to (ENGELS et al., 2011).

According to its authors, the algorithm is suitable for both hardware and software implementations. Besides the interesting properties previously discussed, another of the main reasons behind its choice are that this algorithm is mainly designed for low-cost devices and is suitable for software implementations.

4.2 Communication between PLCs and I/O stations

First, the communication between PLCs and I/O stations must be analyzed. Specifically, to evaluate the performance of these entities the main metric proposed in this work is the latency metric represented by the cycle time. The cycle time allows the performance of both PLCs and I/O stations to be properly evaluated. By comparing the baseline case with the case using the proposed method, it is possible to determine the performance impact of the proposed mechanism. Since the only change in the cases is the usage of the mechanism, it is possible to isolate its performance impact by calculating the difference between the measured times. The introduction of the security mechanism must not cause any kind of interference in the compliance of the device with the application's real-time

requirements. The latency will be most important on the most extreme application scenarios, such as: the ones where the control system is executed with a very small period, using few I/O points; and those in which the control system is executed with a larger period, using large numbers of I/O points. For the proposed method to be valid, it is a requirement that the real-time properties of the system are preserved, even with the necessary computations.

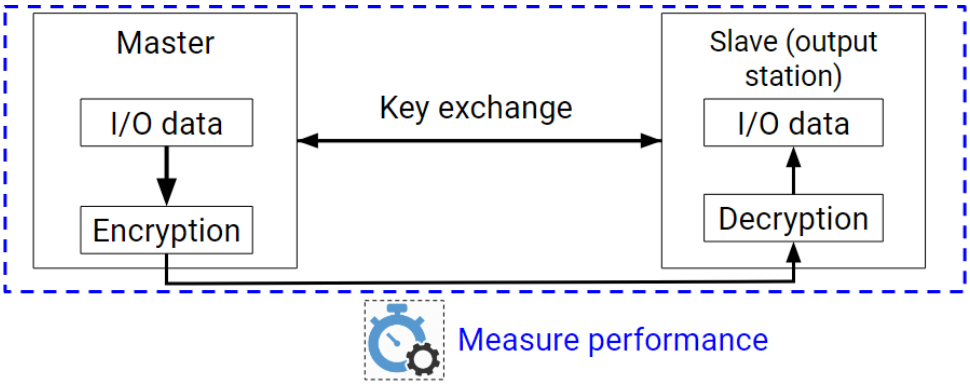
The first step towards a solution is to consider whether it should be implemented on hardware, e.g., ASIC, or software, such as a user or OS program. Usually, implementing on hardware yields better performance, but it is much harder to deploy on the actual plant due to the hassle of changing equipment and possibly higher costs. A software solution could be downloaded on the firmware of each slave and master, making the deployment of the solution much easier, cheaper and faster. Also, the proposed solution could be implemented in hardware in the future if needed. Evaluating the software based solution is the most general case, as the performance of a software implementation tends to be lower. Therefore, this is the reasoning the support the choice of implementing the proposed solution in software.

Based on the chosen algorithm previously detailed in Section 4.1, the proposed mechanism is to use the algorithm to encrypt relevant I/O data. An overview of the solution is described in Figures 4.1 and 4.2. The EtherCAT master encrypts and authenticates its messages to the slaves using the chosen algorithm. The EtherCAT slaves decrypt the message sent by the master and process it. Depending on whether they are input or output stations, they will either just read the EtherCAT commands and data sent by the master, or they will read the EtherCAT commands and write their own encrypted data, respectively. Naturally, the stations can also be de facto I/O stations, i.e. one station which has inputs and outputs. In that case, inputs and outputs can be dealt with separately using the cases presented in Figures 4.1 and 4.2.

It is important to notice that, due to the design of the algorithm, the number of bits generated after the encryption operation is performed remains the same. Since the same number of bits will be sent (even though they are now encrypted), there is absolutely no change in the on-the-fly characteristic of the EtherCAT protocol, as the packet forwarding is independent of the algorithm execution. The only difference is the meaning associated with each bit and the processing that will be necessary once the data is received by a node. This would be the only difference seen when comparing the packets with or without the mechanism. It is also possible to include a variable that would act as a packet counter,

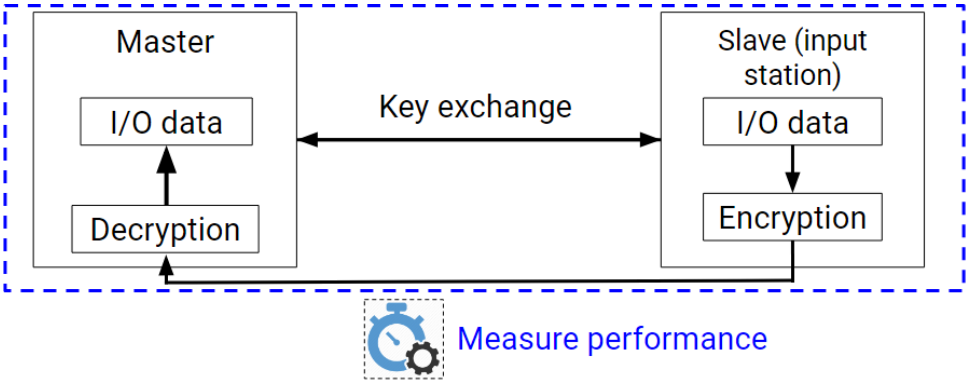
having a unique identifier for each packet. Once encrypted, an attacker would not be able to update this number, and therefore, old packets can be rejected. In this case, there would be the addition of a single variable of, for example, 16 bits. Thus, it is possible to prevent a replay attack using this scheme.

Figure 4.1: Overview of the secure I/O data exchange with an output station



Source: Author

Figure 4.2: Overview of the secure I/O data exchange with an input station



Source: Author

Naturally, the master and all the slaves must have some shared key in order to communicate. In (SAARINEN, 2013), it has been noted that Hummingbird-2 is susceptible to certain related-key attacks. The first 64 bits of the key can be independently recovered with only 2^{36} effort. Although this makes the algorithm less strong, the proposal here is that the secret key should be exchanged (or renewed) from time to time, periodically. This interval should not be too long that it would be time enough for the key to be discovered, but should also not be too short to avoid unnecessary overhead. Evaluating this time inter-

val is outside of the scope of this work, as this would also depend on the chosen algorithm, and the proposed solution does not intend to obligate the usage of Hummingbird-2.

Regarding the key exchange, it is important to take into consideration the premise that states the network is considered unsafe due to the possibility of an attacker having access to it. For instance, a simple straightforward exchange of keys will not prevent an attacker from obtaining the key. Therefore, whichever the key exchange algorithm might be, it must be one that will be able to achieve a secure key exchange over an unsafe network, such as Diffie-Hellman (WIECZOREK et al., 2012). After the first key exchange occurs, in which the master sends the key to the slaves, a time interval to renew the key should be specified on the master. When the time interval has passed, the master renews the key and the new key should be encrypted with the old one. Once the slaves receive the new key, both master and slaves should start securing the data with it.

The exchange of messages will be made using mechanisms implemented by the EtherCAT protocol itself. On the slave, to ensure the data consistency, EtherCAT uses a mechanism called SyncManager to prevent multiple simultaneous access to the EtherCAT Slave Controller (ESC) memory. In practice, this memory can be accessed by the EtherCAT network (the master) or by the local microcontroller (the slave). The SyncManager has two modes of operation: the buffered mode, which is used for cyclic data exchange (the I/Os, for example), and the mailbox mode, which is used for acyclic data exchange by the means of a handshake mechanism.

In this proposed solution, each slave encrypts its own I/O information using the secret key, which should be known by all devices in the bus. It is important to notice that each slave's algorithm processing time will not affect other slaves, since the EtherCAT stack is implemented by hardware and it does not have to wait for the slave to finish processing it to forward the packet to the next slave. Therefore, while one slave is encrypting its I/O data, the packet is already being forwarded to the next node on the bus. It is also important that all modules must have the same key to ensure that the application will perform as expected. Techniques such as using hot swap features which allow redundancy can be used to ensure that the security mechanism will only start being used when all modules have the key.

In this case, as defined, the encrypted message contains only the I/O data, meaning it is only a part of the EtherCAT frame. This data is then sent to the master using the SyncManager's buffered mode. This is the standard operation of a slave when exchanging I/O information. Due to the properties of the algorithm and because the size of the exchanged

data remains the same, the only difference will be the fact that the master will have no more access to the input data in clear (without performing a decryption operation). Input data is mentioned separately in this case because the master can maintain a copy of the unencrypted value, as the master is the one who sets these values.

It is also proposed that the exchange of keys between modules should use the SyncManager's mailbox mode. With this mode, it is possible to avoid causing unnecessary overhead or exchanging the same key multiple times, as the mailbox communication does not happen cyclically like the I/O exchange.

To assess how viable this security mechanism is in practice, a performance evaluation is performed focusing on two main points: the master and the slaves.

On the master, the performance evaluation method is based on two metrics:

- Cycle time: the time between the activation of the control task and the completion of its execution, which comprises the execution times of the embedded software routines and of the security mechanism. With this metric, it is possible to isolate how much impact the mechanism will have. The same insight has been used on (ROBERT et al., 2012) to compare the performance of various real-time Ethernet protocols.
- Time variation amplitude ($t_{max} - t_{min}$): the difference between the maximum and minimum cycle times measured during a certain period of observation. With this metric, it is possible to determine if the computations related to the security mechanism introduce any peaks in time, which would deteriorate the time-determinism of both master and slaves in terms of jitter.

First, the performance of the master running without the security mechanism must be determined. This is achieved by collecting samples of the minimum, average and maximum cycle times. These samples are collected over a period of time and represent the baseline performance of the master. Then, the process is repeated in the master, now with the security mechanism running. After this, it is possible to compare the performances of both cases and measure the actual impact of the implemented mechanism using the two metrics. The sample times are measured with the master's internal timers, which are displayed on the programming tool. It is important to notice that, while the security mechanism has the objective of securing EtherCAT communication, the cycle times refer to the whole task execution time. This comprises the times from interrupt requests, the entire communication stack, and the operating system itself, among others. Therefore,

these times include not only the EtherCAT communication time itself, but other significant times as well.

It is also important to emphasize how critical the scalability is for the master. The master must handle the communication with all slaves, as big as the number of slaves may be. On the other hand, the slaves will have to deal exclusively with its own number of I/O points (e.g, 8 points). This does not mean, however, that scalability can be neglected on the side of the slaves. However, in an application with, for example, 1000 I/O points, while the master will have to process all the points, the plant will be comprised of several individual slave modules, each with a much smaller number of I/O points.

Usually, the master can process a task with the same computational effort in less time than the slaves. Because the entity responsible for running the control task is the master, it is equipped with more "powerful" hardware capable of running much bigger applications than the slave. This means that it is possible to introduce in the master more calculations related to the security mechanism without disrupting the intended functioning of the device. While it is also important to verify if the mechanism does not introduce peaks or jitter in the cycle time, a more worrisome concern is the scalability, when considering bigger quantities of I/O points. Therefore, it is absolutely critical to verify how scalable this mechanism is for an increasing number of I/O points if the results for a single slave prove to be acceptable.

In the case of communicating with an input station, the master sends a reading command to the corresponding slave. Since this packet does not contain any sensitive data, only the command itself, and since the scope of this work is to secure the I/O data, not the EtherCAT commands, it can be transmitted in clear. However, when the master receives the input data, which has been encrypted by the slave, the master needs to decrypt it.

In the other case, when the master communicates with an output station, the master will send a value to be written on one (or more) of the slave's outputs. Therefore, the master has to encrypt the sensitive data in the packet, i.e., I/O data, before sending it to the slave. Then, the slave has to decrypt it once the packet has been received.

Therefore, considering that computations related to the chosen algorithm occur on both ends, these operations add some overhead to the slave as well. One way of evaluating the performance of the slave is based on (CENA et al., 2012). While the scope of the authors' work was directed at evaluating the performance of EtherCAT's Distributed Clock (DC) mechanism, the same tests could be used to compare the performance of slaves with

and without the security mechanism by measuring how synchronized they are. Specifically, in the performed test two input stations (slaves), one with the security mechanism and the other without, are fed with the same nonconstant signal, and compared the timestamp they take on specific level variations.

However, the proposed test to evaluate the performance is to actually measure the software cycle time in the slave module using its internal software timers. The reason for this choice is that the algorithm is implemented in software and, measuring the time using software avoids introducing more factors, such as another slave module or even the precision of EtherCAT's DC mechanism, while also allowing an analysis on the impact of the data transfers to the application. Although software timers might not be the most precise option for measuring hard real-time applications, it is believed that it is precise enough as a proof-of-concept, given that the slave cycle time is in the order of magnitude of milliseconds.

4.3 Extending the Security Mechanism to the Communications between PLCs and SCADA Application

This section analyzes the aspects related to the communications between PLCs and SCADA application and how to prevent the threats previously defined. Similarly to the case of the communications between PLC and I/O stations, it is also possible to deploy a hardware-based solution to secure the communications between PLCs and SCADA. However, this would also cause unwanted costs and efforts in this case, which go against the objectives of this work. Besides, the proposed solution involves integrating the same security mechanism in both segments of the communication to take advantage of its structure to increase the performance. Therefore, just like in the previous case, this will be dealt with using a software-based solution.

In the case of the SCADA application, it is possible to implement user-created scripts that can be used to process data, generate charts, visual representations, etc. This can also be used to implement the encrypting and decrypting operations of the chosen algorithm. There must be, however, a concern regarding performance in both ends due to the real-time requirements, whatever the adopted defense mechanism is. The performance of such operations will also most likely be affected by the computational power of the computer running the SCADA application. Therefore, it is important to specify the parameters of the system while designing the whole industrial application. Naturally,

depending on the chosen algorithm, the system may present a different performance as well.

Using regular cryptographic algorithms on real-time systems can be tricky due to the performance considerations, as it may incur an unacceptable latency or power consumption. This is the case of the algorithms mentioned by the SCADA security standards analyzed in (GHOSH; SAMPALLI, 2019), which show how important is the performance aspect, especially considering scalability aspects. While neither the PLCs themselves or the SCADA system are low-end devices, they have an important requirement related to response time, which must be met.

The proposed solution focuses on ensuring the secure communication of I/O data while targeting scalability and performance aspects of all nodes, including the SCADA software application. This is achieved by making use of the already encrypted data calculated by each PLC in the application. The proposed approach can be better understood with the aid of Figure 4.3, by dividing it in the two parts previously defined: the first part is the communication of each individual PLC with the I/O stations, which are their slaves; the second part represents the communication of all the PLCs present in the plant with the SCADA system.

The first part has been thoroughly described in Section 4.2. The important detail to understand how the security will be applied to SCADA is to note that each PLC will store the encrypted values of each I/O variable. This is necessary to avoid the overhead of encrypting all the relevant variables once again before sending them to the SCADA application, which would most likely not be feasible in terms of performance. When looking at the whole control process, it is possible to say that the encryption will be computed distributedly, as they will be calculated by the PLCs and I/O stations beforehand.

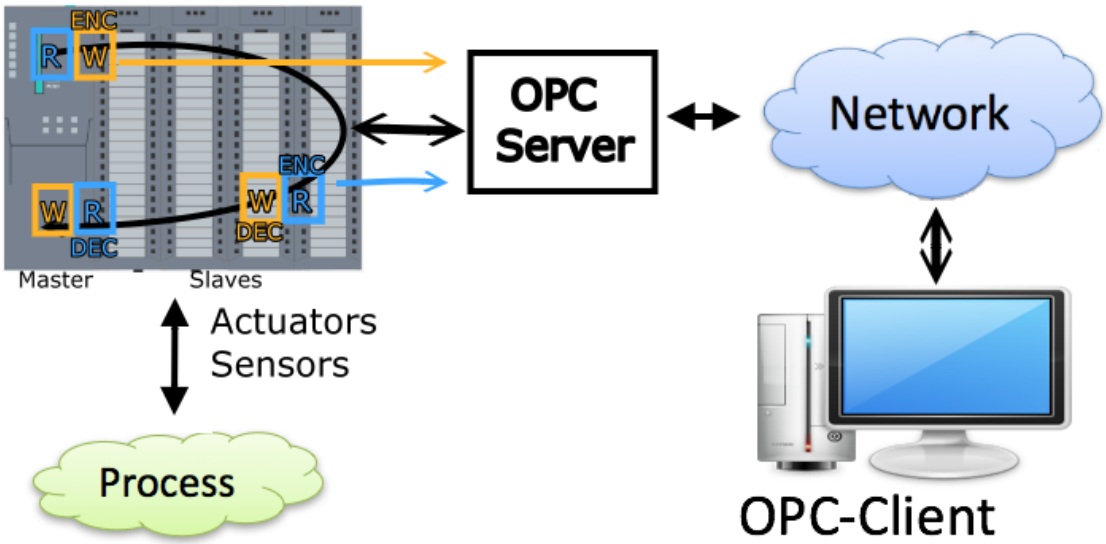
At this point, given that all variables of interest have their encrypted form calculated, there is no additional overhead for the PLCs or the I/O stations. Therefore, the PLCs are able to send the encrypted variables in a very straightforward way to the SCADA system. This also means that extending the security mechanism to the SCADA system will happen in an almost transparent way from the point of view of the PLCs and I/O stations, the only drawback being having to use more memory to store the copies of the variables (on the PLC).

The exchange of data between all the entities is represented in Figure 4.3. The two arrows which go from the writing value (W) and the reading value (R) represent the values that were previously calculated and will be sent to the SCADA system by each

PLC. Naturally, the writing value should only be sent once the master receives a confirmation that the value has been successfully written. However, this has been represented differently in the figure to facilitate the understanding.

The same process can be repeated for all the PLCs in the control plant. Each one of the PLCs will then send their relevant variables to the SCADA system. As these variables are all encrypted, the I/O data is now secure. Once the variables reach the SCADA system, the software application must decrypt all of them before showing them to the end user, whatever the user application may be. This part might be a problem in terms of scalability, depending on the overall performance of the chosen cryptographic algorithm and on the computer running the SCADA application. As the SCADA system will receive variables from all PLCs, which could sum up to thousands of variables, it is important to carefully choose the algorithm to be used. If it is possible to estimate the size of the control application, that information can also be used to determine the viable option when choosing the algorithm.

Figure 4.3: Solution overview



Source: Adapted from halvorsen.blog

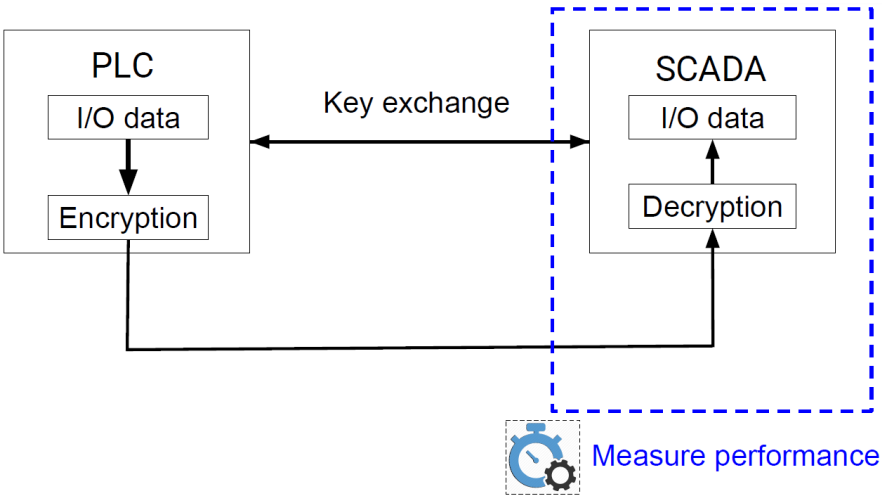
While it is not part of the contribution of this work to actively specify one cryptographic algorithm, Hummingbird-2 has been chosen as a proof of concept in order to assess if authenticated encryption is a valid way of securing I/O data. However, the same mechanism can be applied using different algorithms, and the algorithm can be changed if any other better option is proposed. Ideally, it would be good to conduct tests to com-

pare different algorithms and determine which one of them serves the best. Even a small difference in the performance will reflect on each one of the possibly thousand variables of the application, which could end up having a great impact on the final performance.

Similarly to the first part presented, there also must be some kind of key exchange between the SCADA system and the PLCs. There are proposed key exchange schemes in standards such as IEC 62531 and AGA-12 (GHOSH; SAMPALLI, 2019). This work does not aim to assess the viability or to evaluate the performance of key exchanging protocols, but to evaluate the overhead imposed by the proposed security mechanism. Therefore, the performance considerations related to the key exchange are not considered in this study, although they should be considered in future works and in real applications.

To assess the performance of the SCADA system, an experiment is proposed as depicted in Figure 4.4. Because Hummingbird-2 is an algorithm which has the decryption operation much more costly than the encryption, according to its authors. Therefore, an experiment based on decryption was designed to evaluate the worst-case scenario. If the worst-case scenario shows an acceptable performance, it is safe to say that any combination of encryption and decryption operations will have an acceptable performance as well. Therefore, the experiments are comprised of one or more PLCs that send their encrypted variables through the network to the SCADA system. The SCADA will then decrypt the received variables.

Figure 4.4: Experiment schematic (SCADA reading PLC's inputs)



Source: Author

Based on the experience of engineers who work with SCADA system on a daily

basis, two metrics have been proposed to evaluate the impact of the proposed security mechanism:

- Synchronization time: time to effectively start communicating (i.e, from starting the system to the first data displayed on the screen);
- Response time: how long it takes for an I/O point to be updated, i.e., the time between two reads or writes.

5 EXPERIMENTS AND RESULTS

5.1 Communication between PLC and I/O station

The experiments were conducted on a real setup comprised of one master, a commercial PLC, and one slave, 8 analog input station, in which each input is represented by 16 bits. The slave is an embedded device, which runs a firmware, equipped with a 32 bits ARM Cortex-M3 processor running at 72 MHz. The slave firmware code is written in C and has been made available for this work. This firmware was adapted to implement the proposed changes, such as the encryption and decryption of I/O variables.

The master is an ordinary PLC designed for small/medium-sized industrial applications, with up to 64 I/O stations in the same bus, which are addressed with EtherCAT's auto-increment addressing mechanism. Considering 8 input/output points per I/O station, using a PLC which supports up to 64 I/O stations, that means up to 512 I/O points. It runs a Linux operating system and executes a runtime system, which handles the tasks directly related to making the device compliant with the IEC 61131-3 standard for industrial controllers. Besides the I/O updates, there are other features also implemented on the master, such as diagnostics; the discovery cycle, where the master tries to discover new modules on the bus, which have not yet been configured; and the communication with the programming tool. It is important to notice that the execution of these tasks also introduces an overhead, as well as some variability in the execution time. On the EtherCAT master, the measured times take into account the overhead introduced by these software layers. It is equipped with a 32 bits PowerQUICC II Pro processor running at 417 MHz and 64 MB RAM. The chosen algorithm was implemented on both modules according to the proposed implementation.

According to the number of instructions for the implementation, the decoding operation should be approximately 50% more costly. Although it is difficult to accurately predict execution time, making a rough estimate by considering an average number of cycles per instruction of one or two, the mechanism should be viable on both devices.

While it is expected that different PLCs and I/O stations have different performances, the experiments can be conducted as a proof-of-concept, determining if it is viable to use cryptography as a security mechanism for communications between master and slave.

For the sake of evaluating the performance, the specific part about the key ex-

change between master and slave nodes, as proposed in Figures 4.1 and 4.2, has been omitted because it is not an event that should frequently happen. The key exchange messages can be appended as one or more datagrams in an EtherCAT packet, such as read and write commands, meaning its impact should be minimal. Therefore, this impact will be considered negligible.

The testbed is comprised of the two modules, one master and one slave, connected on a high-speed 100 Mbit/s network which uses EtherCAT. The master communicates with the development environment through a point-to-point connection. In the execution of each experiment, a user program has been programmed on the master to encrypt and/or decrypt the I/O data received from the slave, according to its objective. The slave encrypts or decrypts the data according to the algorithm implemented on its firmware. There is no need for more than one slave to evaluate the performance of the algorithm, as each slave will process its own I/O independently and that does not affect the packet forwarding to the following modules.

As previously defined, the first experiment focuses on obtaining the baseline performance of both modules running without the security mechanism by measuring cycle times. On the master, 3500 samples of the cycle time were acquired.

The number of samples in each experiment was chosen to both optimize the testing time by not collecting an excessive number of samples and, yet, provide meaningful and trustworthy results. The results of experiments comprising 3500 and 7000 samples were compared and no significant discrepancies have been found. Therefore, all the following experiments on the master have been conducted with 3500 samples. The samples were measured with an internal hardware timer with a precision higher than 1 μs.

On the side of the slave, due to memory restrictions on obtaining the samples, the experiments have been conducted with 800 samples. Still, judging by the variability of the samples, an experiment comprised of 800 samples is statistically meaningful. The experiments were repeated ten times each and no significant discrepancies have been found. The timer used on the slave has a precision of 62.5 μs, approximately. This has been considered enough precision because of the order of magnitude of the cycle time, which is in the order of milliseconds.

According to the obtained results reported in Table 5.1, it is noticeable that the time variation amplitude is quite big (809 μs) even in a normal execution. Preliminary studies point that the majority of this jitter is caused by the routines with process the packets. These routines use a semaphore to achieve mutual exclusion of certain specific

Table 5.1: Master baseline performance

Cycle Time(μs)				
Min	Avg	Max	Std. Dev.	$t_{max} - t_{min}$
1693	1878	2502	28.134	809

Source: Author

Table 5.2: Slave baseline performance

Cycle Time(ms)				
Min	Avg	Max	Std. Dev.	$t_{max} - t_{min}$
6	6.07	6.5	0.216	0.5

Source: Author

resources, which cause certain threads to be blocked at times, generating a variation in the cycle time. Still, since exploring the reasons for this variability and these peaks in the cycle time is out of the scope of this work, the baseline performance will be considered acceptable. The impact of the security mechanism will be evaluated in comparison to the baseline performance, considering only the potential increase in the time variation amplitude. It is also noticeable that the standard deviation is pretty small (approximately 28 μs). This information will be used to evaluate if the algorithm introduces variability in the cycle time.

Analyzing the data in Table 5.2, it has been found that the cycle time in the slave is much bigger than the one in the master. However, because of the fixed amount of I/O points each slave must process, the potential impact on the cycle time is also much smaller. It is also noticeable that the standard deviation is very small, meaning that there is a very small variability over various cycles.

The first experiment conducted on the slave is comprised of encrypting its own 8 analog inputs, equivalent to 128 bits. This is the scenario that would actually be encountered in a real application using this mechanism in an input station.

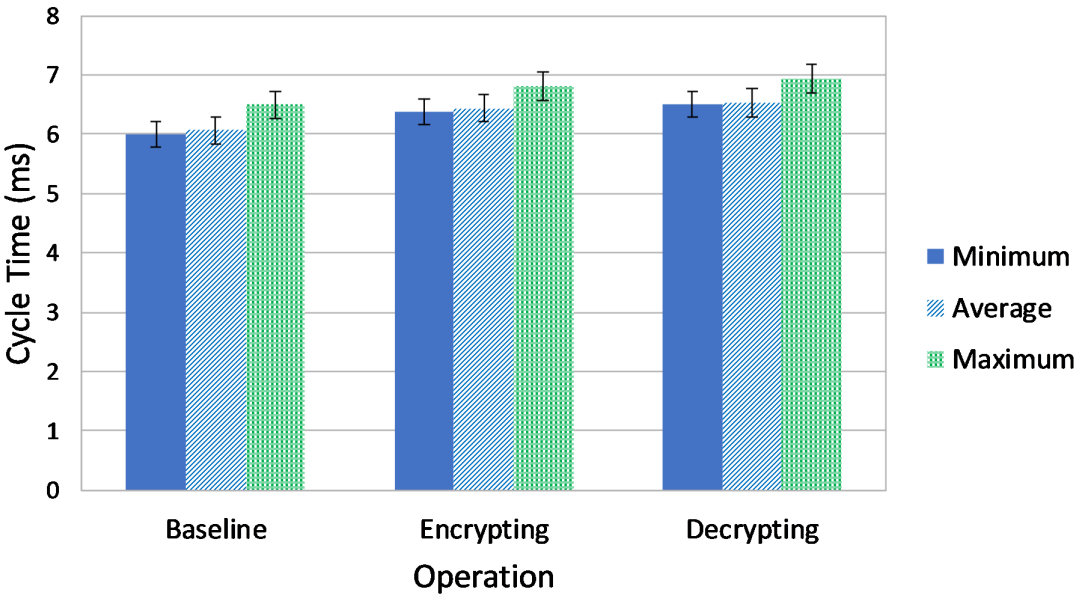
In the second experiment, the slave decrypted its own 8 analog inputs, which is performance-wise equivalent to the eight output values which would be sent by the master. While the used slave station is an input station, i.e. it does not have to use decryption when communicating with the master (it would only encrypt its I/O data before sending it), one experiment evaluated the performance of the decryption. Although it does not necessarily make sense for an input station to decrypt the I/O data in a real application, this scenario is still useful to evaluate the case in which an output station would be used. Therefore,

this is a way of evaluating this scenario without actually using another different slave, thus simplifying the setup for the experiments.

The comparison between the baseline and the two experiments can be seen in Figure 5.1. For the sake of a more detailed analysis, the cycle times are also presented in Table 5.3. As previously mentioned, considering the slave performance, it is not necessary to analyze or experiment with bigger quantities of I/Os, because each I/O station is responsible for encrypting/decrypting the I/O data related only to its own I/O points. Therefore, a control task with hundreds of I/O points would be comprised of several different I/O stations, each with a fixed number of I/O points (e.g., 8 points). It is therefore assumed that on a real application, the individual performance of each I/O station can be evaluated on a smaller testbed, as the number of I/O points each station will process will be the same independently of the size of the application (assuming that the station is processing all its available points).

The impact on the slave is percentually small, approximately 6% for encrypting and 7.5% for decrypting I/Os. Also, the time variation amplitude did not present significant changes, even showing smaller values. This is due to the precision of the timer used in the measurements and that difference is therefore considered negligible. It is important to notice that, if the station is a digital I/O station, the cycle times would be even closer to the baseline, as the overhead would be much smaller because fewer bits would be processed.

Figure 5.1: Measured slave cycle times in different scenarios



Source: Author

Table 5.3: Detailed measured slave cycle times in different scenarios

Cycle Time (ms)					
	Min	Avg	Max	Std. Dev.	$t_{max} - t_{min}$
Baseline	6	6.07	6.5	0.216	0.5
Encrypting 128 bits	6.375	6.430	6.812	0.228	0.437
Decrypting 128 bits	6.5	6.531125	6.9375	0.233	0.437

Source: Author

Table 5.4: Detailed measured master cycle times in different scenarios

Cycle Time (μs)					
	Min	Avg	Max	Std. Dev.	$t_{max} - t_{min}$
Baseline	1693	1878	2502	28.134	809
Encrypting 128 bits	1823	1953	2575	33.488	752
Decrypting 128 bits	1944	2024	2711	33.888	767

Source: Author

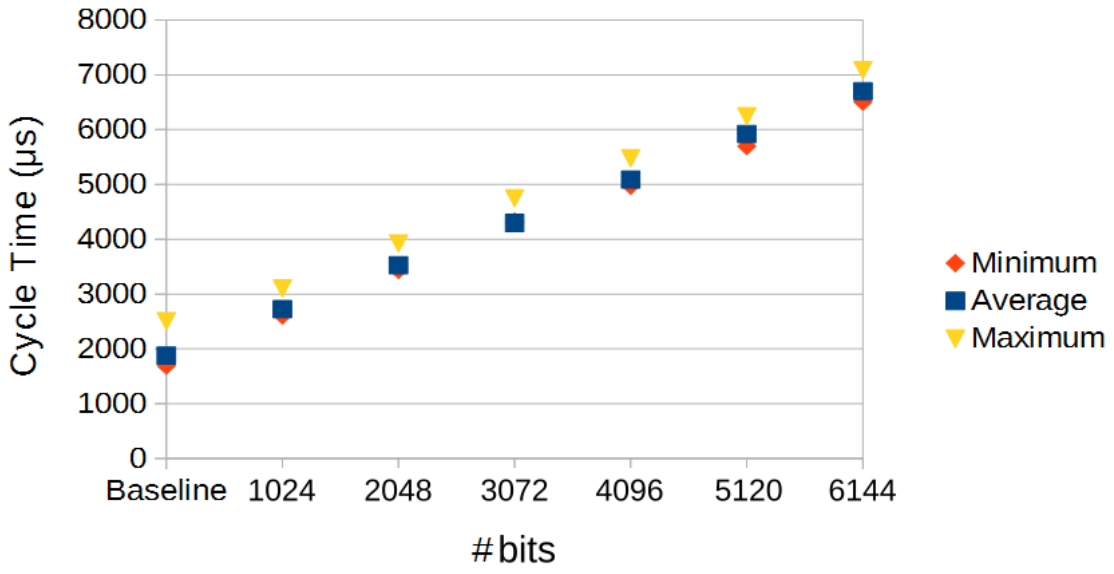
The first step in assessing the impact on the master is to consider a scenario where the master communicates with a single slave. The experiment was conducted considering scenarios in which the master communicates with an analog input station (i.e., it decrypts received data) and in which it communicates with an analog output station (i.e., it encrypts data before sending). The results are presented in Table 5.4.

According to the obtained results, there has been a slight increase in standard deviation in both cases, encrypting and decrypting I/Os. It appears there is no significant increase in the time variation amplitude, which suggests no peaks in the processing time caused by the security mechanism. While the obtained values are smaller than the baseline, this is due to the inherent variability in the cycle time of the master, which results in rare peaks in time, as can be noticed by analyzing the standard deviation of the baseline.

Last but not least, evaluating the scalability with an increasing number of I/O points is also an important aspect. With this concern in mind, first, several experiments were conducted, each time increasing 64 analog I/O points (1024 bits), to determine if the cycle time grows in a linear fashion, proportional to the number of I/O points. The I/O points were increased through software, i.e. the slave itself maintains its original number of points. Since the evaluation is focused on the security of the I/O data itself, in practice,

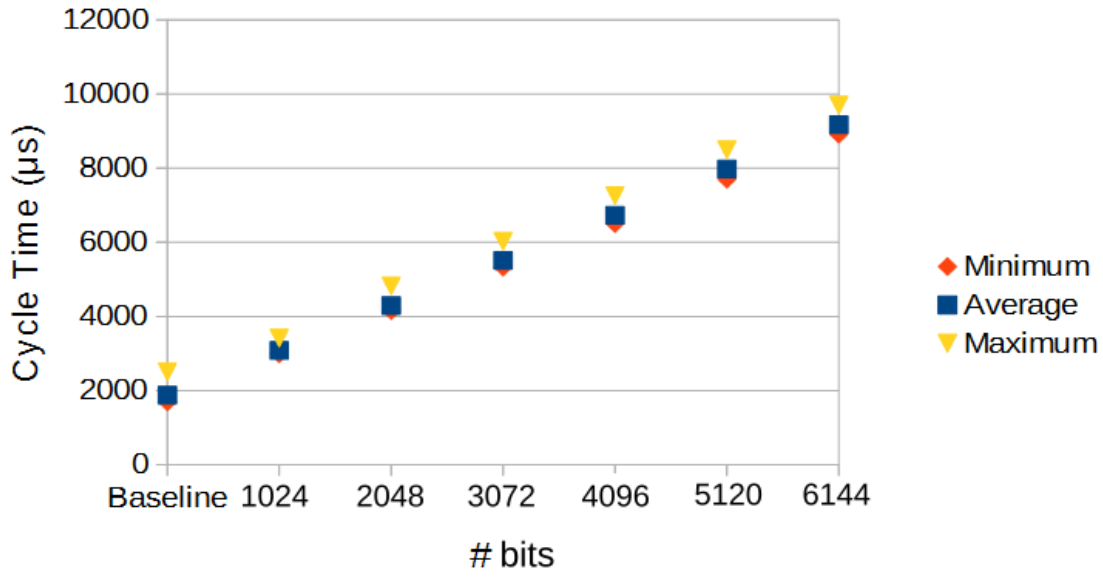
this is equivalent to securing the data of a slave with more points. These experiments were conducted for both encryption and decryption. The results of these experiments are presented in Figures 5.2 and 5.3. Similarly to the results obtained on the slave, the standard deviation while encrypting/decrypting many I/O points did not have a significant change in respect to the values obtained in the baseline experiment.

Figure 5.2: Master's encoding operation showing a linear behavior



Source: Author

Figure 5.3: Master's decoding operation showing a linear behavior



Source: Author

Table 5.5: Measured master cycle times while encrypting I/O points

Cycle Time (μs)					
Encrypting	Min	Avg	Max	Std. Dev.	$t_{max} - t_{min}$
Baseline	1693	1878	2502	28.134	809
128 bits	1823	1953	2575	33.488	752
256 bits	1989	2066	2715	34.243	726
512 bits	2173	2270	2900	31.396	727
1024 bits	2608	2724	3402	30.439	794
2048 bits	3438	3527	4123	29.741	685
4096 bits	4976	5089	5577	29.772	601
8192 bits	8166	8345	8861	30.348	695
16384 bits	14535	14760	15224	31.586	689

Source: Author

Table 5.6: Measured master cycle times while decrypting I/O points

Cycle Time (μs)					
Encrypting	Min	Avg	Max	Std. Dev.	$t_{max} - t_{min}$
Baseline	1693	1878	2502	28.134	809
128 bits	1944	2024	2711	33,888	767
256 bits	1995	2163	2730	36,554	735
512 bits	2276	2492	2977	34,329	701
1024 bits	2789	3082	3454	30,556	665
2048 bits	4155	4295	4823	32,93	668
4096 bits	6503	6730	7258	37,592	755
8192 bits	11208	11424	12039	31,578	831
16384 bits	20670	21022	21570	35,658	900

Source: Author

According to these results, it is possible to state that the cycle time grows linearly related to the number of encrypted/decrypted I/O points. Therefore, the final experiment was the assessment of how much a big number of I/O points would affect the master. The experiments were conducted up to 1024 I/O points (16384 bits), considering encryption and decryption operations, and the results are presented in Tables 5.5 and 5.6.

With the obtained results, it is noticeable that the decryption operation demands more execution time than encryption. This makes sense with the statement of the authors of Hummingbird-2, who mention that the decryption operation is more costly. Evaluating the time amplitude variation, it is also possible to conclude that the decryption does have a certain contribution to the execution time peaks. Another fact worth mentioning is that the decryption of 1024 I/O points results in an increase of approximately 20 ms in the

cycle time.

On the slave, for 8 analog I/Os, which correspond to 128 bits, the security mechanism increased the cycle time by approximately 0.4 ms, when encrypting, and by 0.5 ms, when decrypting. On the master, for a big application with 1024 analog I/O points (16384 bits), there was an increase of 20 ms in average cycle time, with no significant spikes in execution time.

5.2 Communications between PLCs and SCADA System

The experiments were conducted on a real setup, using real equipment, four PLCs and a computer running the SCADA system. Each PLC is equipped with a 32 bits PowerQUICC II Pro running at 417 MHz and 64 MB RAM. The SCADA system runs on a computer equipped with a Intel® Core™ i7-3770 running at 3.4 GHz and 8 GB RAM.

The PLCs and the SCADA system were connected to the university network during work hours, meaning the network presented regular traffic from other users and university affairs. The reason to avoid a point-to-point connection is that the university network, with other users sending and receiving data through it, is much more similar to a real use scenario. One of the PLCs used in the experiments is shown in Figure 5.4. This figure shows two separate modules, one external supply (on the left) and one PLC (on the right), connected on the same bus, as well as the PLC connected to the university network by an ethernet cable.

As the focus of the work of these experiments is on the performance of the SCADA system, the experiments were conducted with the PLC sending its variables to the SCADA. In other words, the SCADA reads the PLCs' inputs. This means that, according to the proposed solution presented in Figure 4.4, the PLC encrypts the data and the SCADA system decrypts it. The reason for this way of performing the experiments is that the decryption process in Hummingbird-2 is more costly in terms of processing time. Therefore, this is the worst-case scenario for the SCADA in terms of computational effort. If this case proves to be acceptable, then there should be no problems with others.

The test application on the SCADA is a project with 20,000 integer variables, each with a size of 16 bits, totaling 320,000 bits of data. Each one of the four PLCs has 5,000 variables, which are sent to the SCADA through the network. The SCADA system chosen for the experiments is a commercial tool used in real applications. In a real plant, not every variable is an integer, many can be just one bit (digital variables), but for the

Figure 5.4: Example of a bus containing an external power supply and a PLC (manufacturer name omitted as requested)

Source: Author

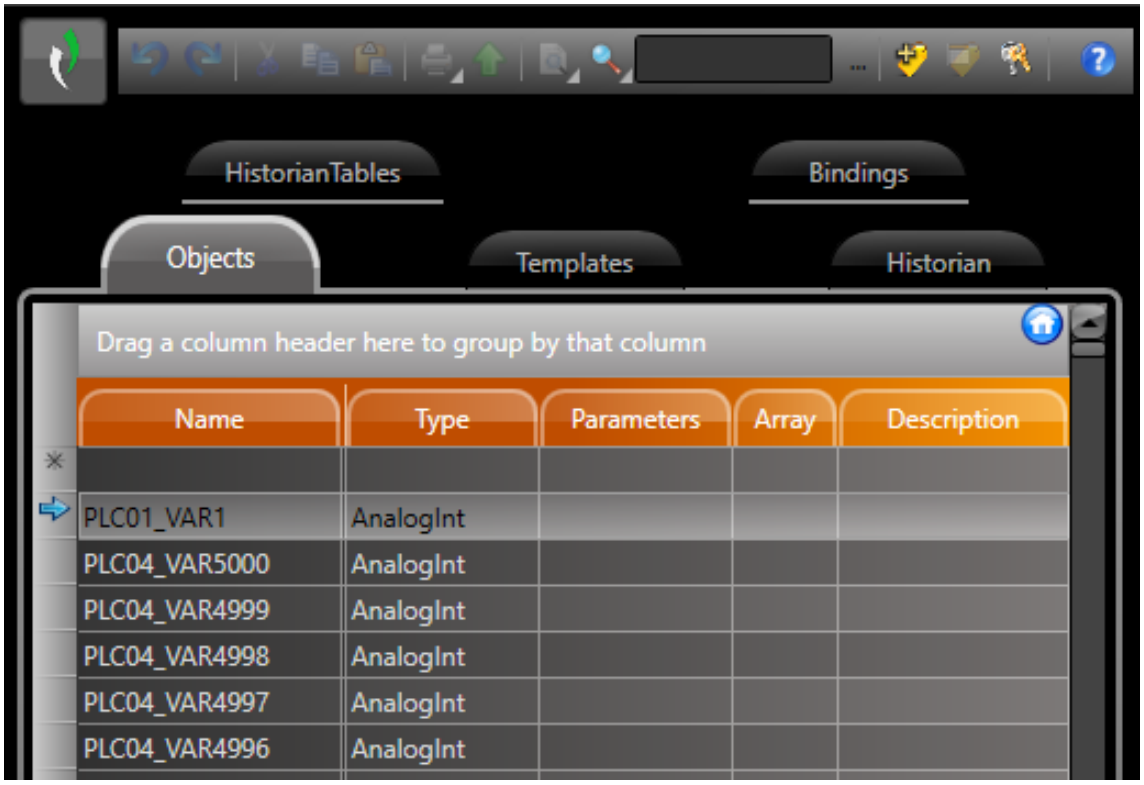
sake of this experiment, it has been decided to use a larger quantity of data. According to discussions with industry personnel, 20,000 tags is a reasonable number of tags for a real application. Notably, not necessarily a real application with 20,000 tags would be comprised solely of four PLCs.

In Figure 5.5, the mapping of tags on the SCADA application is shown. This is where the variables to be monitored are specified. In Figure 5.6, the PLCs which will communicate with the SCADA are configured, defining OPC DA as the interface to be used. In Figure 5.7, each tag is mapped to a real variable, i.e. the actual variable in one of the PLCs.

The experiments have been repeated 100 times in both scenarios, i.e. with and without the proposed security mechanism. The results from the baseline performance (without security) are presented in Table 5.7. The results from the secure communication case are presented in Table 5.8.

In the baseline case, the experiments have shown an average synchronization time of approximately 41 seconds, while the case using the security mechanism has shown an average of approximately 43.3 seconds. Considering the standard deviation and median

Figure 5.5: Tag mapping on SCADA



Source: Author

Table 5.7: SCADA system baseline performance

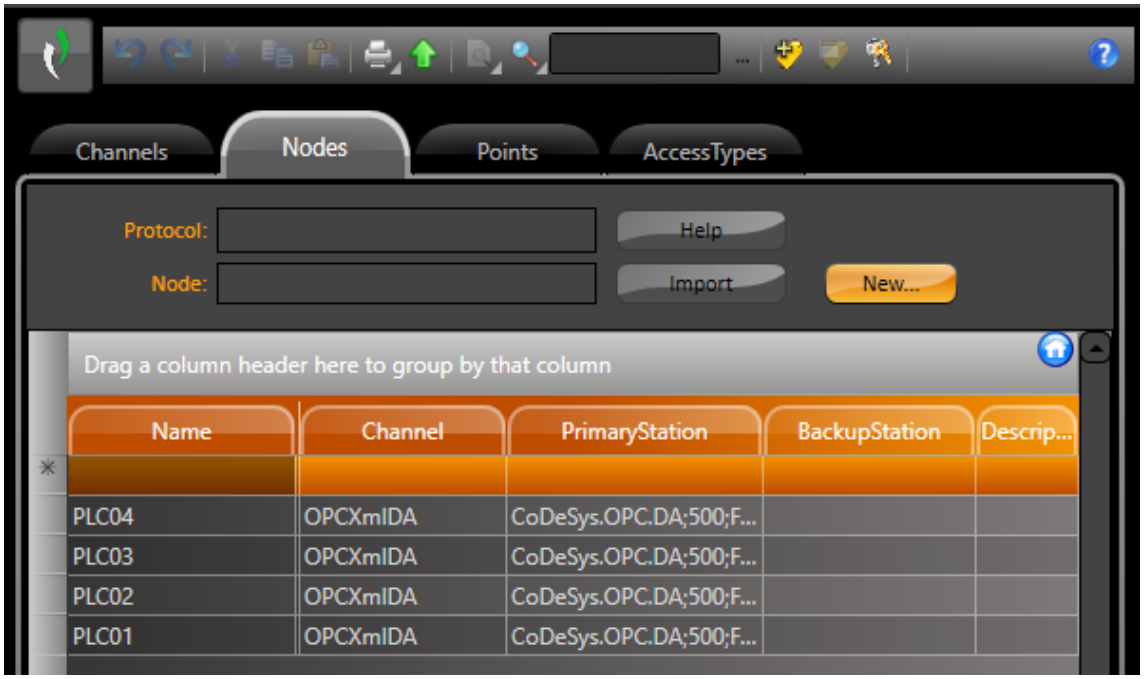
Metric	Average	Std. Dev.	Median
Synch. Time (s)	41.03	11.03	41.18
Response Time (s)	15.11	5.58	14.36

Source: Author

of both cases, there is a slight increase which is most likely caused by the use of the proposed mechanism. Still, considering the average time, it is an increase of just 5.6%, which is considered a good result. While more than 40 seconds can be seen as too much time in industrial applications, it is important to notice that this will only happen once, at the startup of the plant. This means that, if the plant does not stop at any time, this would not have an impact on the production. Besides, as the SCADA software used in the experiments is a commercial solution, its baseline performance is deemed acceptable for use in industrial applications. Therefore, every comparison is made considering the baseline performance as an acceptable value.

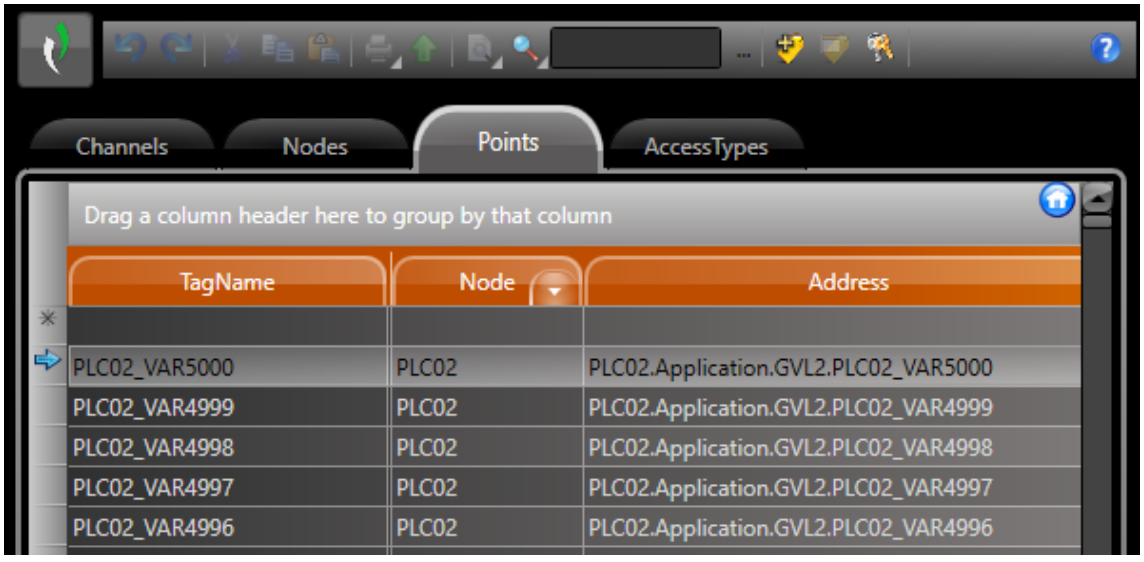
Similarly, the response time in both cases is very similar, approximately 15 seconds for the baseline case and 16.3 seconds with the security mechanism active. In this

Figure 5.6: Nodes configuration on SCADA



Source: Author

Figure 5.7: Points mapping on SCADA



Source: Author

case, the increase is a bit bigger, approximately 8.6%. It is important to notice that this does not mean the system can only act upon the field devices once every 15 seconds. It is the average time it takes for an operator, using a workstation, to notice a change in a

Table 5.8: SCADA system with secure communication performance

Metric	Average	Std. Dev.	Median
Synch. Time (s)	43.36	11.90	43.62
Response Time (s)	16.31	5.76	16.60

Source: Author

variable, considering the moderate-sized plant with 20,000 integer variables, although the control plant is still operating as intended in this meanwhile.

According to the results from the performed experiments, there have been no significant differences in the usability of the SCADA system, considering the chosen metrics, using or not the proposed security mechanism. The reason behind this is because the computer which will run the SCADA system has a vastly superior computational power compared to the PLCs. This enables the SCADA application to deal with a much greater number of operations, although it is important to emphasize that the computer used in these experiments is by no means a high-end device. Still, it is important to be aware of the performance considerations, meaning a lightweight cryptographic algorithm is still a good choice for such an application. This is particularly important considering that the computers running the SCADA system can be more low-end, or the plant can be big.

6 CONCLUSION

This work proposed and assessed the viability of introducing a security mechanism based on authenticated encryption in industrial communications. Particularly, in the EtherCAT communications between a master and its slaves and in the communications between the PLCs and the SCADA system achieved using OPC DA. The studied target applications are automation and control tasks, which have hard real-time requirements and the focus has been on determining the impact introduced by the proposed mechanism.

Experiments have been conducted on both master and slave to answer two fundamental research questions: whether it is possible to implement this on a slave, considering the overhead it would introduce; and how scalable the proposed solution is, considering bigger applications with bigger number of I/O points, where the master would need to communicate with many more slaves.

Evaluating the results on the side of the slave, it is noticeable that the impact in performance is not very high, but it also depends on how many I/O points the slave has. Naturally, using the mechanism would introduce a delay in the response time of that station. Considering an extreme case where a fast response time is needed, e.g. a saw safety braking mechanism, it would still take at least a few milliseconds. Therefore, while there are some very selective applications that this small increase could potentially cause damage, for most applications, the slave can be secured.

According to the results on the master's side, it is possible and viable to use the proposed mechanism for many applications, depending on the size of the application (in number of I/O points) and on the required task cycle time.

Based on the obtained results from both master and slave, it is possible to state that the mechanism is viable for many applications. Therefore, after concluding the security mechanism is viable on both master and slave sides, causing no change to the real-time characteristics of EtherCAT, it is safe to state that this is a viable option to prevent malicious changes in packet content and retrieval of sensitive/confidential control information for most applications. Some time-critical applications may not be able to benefit from this type of defense, but in general, it is a suitable solution and enables securing the EtherCAT I/O communications without requiring new devices. Furthermore, it is possible to state that the proposed approach is a suitable way of providing a scalable defense mechanism to secure the automation devices and I/O stations zone, according to IEC 62443 (SHAABAN; KRISTEN; SCHMITTNER, 2018).

Analyzing the presented results related to the communications between PLCs and SCADA, it is possible to conclude that cryptographic algorithms are a good choice to secure the communications between PLCs and SCADA systems using OPC-DA. There is no unviable or intolerable performance degradation by using the proposed security mechanism. The proposed mechanism works well even with a large number of variables being monitored as shown in the experiments.

As future work, the same security mechanism can be applied to a wide range of other situations, such as securing EtherCAT's distributed clock frames or securing data that does not have strict temporal requirements, e.g. configuration frames. This shows that the mechanism is versatile.

Moreover, a key exchange mechanism on EtherCAT can be developed, possibly through mailboxes, and evaluating its performance to ensure that a complete security mechanism is still viable. It would also be interesting to study the performance impact of the key exchange algorithms related to the exchange between PLCs and the SCADA application, studying if they scale well with increasing numbers of tags and variables, comparing the known security standards with each other. Another possible work is to evaluate other cryptographic algorithms, even hardware-optimized ones, to determine which ones are optimal for each application. It is also possible to compare these results with ones that follow some of the standards previously mentioned.

REFERENCES

ABBAS, H. A.; MOHAMED, A. M. Review on the design of web based scada systems based on opc da protocol. **arXiv preprint arXiv:1506.05069**, 2015.

ÅKERBERG, J.; BJÖRKMAN, M. Exploring network security in profisafe. In: SPRINGER. **International Conference on Computer Safety, Reliability, and Security**. [S.l.], 2009. p. 67–80.

AKPINAR, K. O.; OZCELIK, I. Development of the ecat preprocessor with the trust communication approach. **Security and Communication Networks**, Hindawi, v. 2018, 2018.

ANTON, S. D. et al. Two decades of scada exploitation: A brief history. In: IEEE. **2017 IEEE Conference on Application, Information and Network Security (AINS)**. [S.l.], 2017. p. 98–104.

BECKMANN, G. Ethercat communication specification, version 1.0. **EtherCAT technology group**, 2004.

BIRYUKOV, A.; PERRIN, L. State of the art in lightweight symmetric cryptography. **IACR Cryptology ePrint Archive**, v. 2017, p. 511, 2017.

BOLTON, W. **Programmable logic controllers**. [S.l.]: Newnes, 2015.

BROOKS, P. Ethernet/ip-industrial protocol. In: IEEE. **Emerging Technologies and Factory Automation, 2001. Proceedings. 2001 8th IEEE International Conference on**. [S.l.], 2001. v. 2, p. 505–514.

CENA, G. et al. Evaluation of ethercat distributed clock performance. **IEEE Transactions on Industrial Informatics**, IEEE, v. 8, n. 1, p. 20–29, 2012.

CHAI, Q.; GONG, G. A cryptanalysis of hummingbird-2: The differential sequence analysis. **IACR Cryptology ePrint Archive**, Citeseer, v. 2012, p. 233, 2012.

CHUANYING, Y.; HE, L.; ZHIHONG, L. Implementation of migrations from class opc to opc ua for data acquisition system. In: IEEE. **2012 International Conference on System Science and Engineering (ICSSE)**. [S.l.], 2012. p. 588–592.

DECOTIGNIE, J.-D. Ethernet-based real-time and industrial communications. **Proceedings of the IEEE**, IEEE, v. 93, n. 6, p. 1102–1117, 2005.

ENGELS, D. et al. The hummingbird-2 lightweight authenticated encryption algorithm. In: SPRINGER. **International Workshop on Radio Frequency Identification: Security and Privacy Issues**. [S.l.], 2011. p. 19–31.

FALLIERE, N.; MURCHU, L. O.; CHIEN, E. **Symantec Security Response**. [S.l.: s.n.], 2011.

FOVINO, I. N. et al. Design and implementation of a secure modbus protocol. In: SPRINGER. **International conference on critical infrastructure protection**. [S.l.], 2009. p. 83–96.

GHOSH, S.; SAMPALLI, S. A survey of security in scada networks: Current issues and future challenges. **IEEE Access**, IEEE, v. 7, p. 135812–135831, 2019.

GRANAT, A.; HÖFKEN, H.; SCHUBA, M. Intrusion detection of the ics protocol ethercat. **DEStech Transactions on Computer Science and Engineering**, n. cnsce, 2017.

HANNELIUS, T.; SALMENPERA, M.; KUIKKA, S. Roadmap to adopting opc ua. In: IEEE. **2008 6th IEEE International Conference on Industrial Informatics**. [S.l.], 2008. p. 756–761.

HOFFMANN, M. et al. Continuous integration of field level production data into top-level information systems using the opc interface standard. In: **Automation, Communication and Cybernetics in Science and Engineering 2015/2016**. [S.l.]: Springer, 2016. p. 855–868.

HOLM, H. et al. A survey of industrial control system testbeds. In: BUCHEGGER, S.; DAM, M. (Ed.). **Secure IT Systems**. Cham: Springer International Publishing, 2015. p. 11–26. ISBN 978-3-319-26502-5.

Huang, K. et al. A game-theoretic approach to cross-layer security decision-making in industrial cyber-physical systems. **IEEE Transactions on Industrial Electronics**, p. 1–1, 2019. ISSN 0278-0046.

HUNKAR, P. **OPC UA vs OPC Classic**. Available: <http://www.dsinteroperability.com/OPCClassicVSUA.pdf>. Access: 12 aug. 2020.

IGURE, V. M.; LAUGHTER, S. A.; WILLIAMS, R. D. Security issues in scada networks. **Computers & Security**, Elsevier, v. 25, n. 7, p. 498–506, 2006.

KANAMARU, H. Bridging functional safety and cyber security of sis/scs. In: IEEE. **2017 56th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)**. [S.l.], 2017. p. 279–284.

MILLER, B.; ROWE, D. A survey scada of and critical infrastructure incidents. In: ACM. **Proceedings of the 1st Annual conference on Research in information technology**. [S.l.], 2012. p. 51–56.

NARULA, L.; HUMPHREYS, T. E. Requirements for secure clock synchronization. **IEEE Journal of Selected Topics in Signal Processing**, v. 12, p. 749–762, 2017.

NAZIR, S.; PATEL, S.; PATEL, D. Assessing and augmenting scada cyber security: A survey of techniques. **Computers & Security**, Elsevier, v. 70, p. 436–454, 2017.

NGUYEN, V. Q.; JEON, J. W. Ethercat network latency analysis. In: IEEE. **2016 International Conference on Computing, Communication and Automation (ICCCA)**. [S.l.], 2016. p. 432–436.

NICHOLSON, A. et al. Scada security in the light of cyber-warfare. **Computers & Security**, Elsevier, v. 31, n. 4, p. 418–436, 2012.

OMRON Automation Pvt Ltd. **EtherCAT Communication Manual**. [S.l.]: OMRON, 2016. Available: <https://www.edata.omron.com.au/eData/Vision/Q179-E1-01.pdf>. Access: 12 aug. 2020.

OPC Foundation. **OPC-UA Security; 2018-12**. Available: <http://wiki.opcfoundation.org//index.php?title=Security>. Access: 12 aug. 2020.

PLIATSIOS, D. et al. A survey on scada systems: Secure protocols, incidents, threats and tactics. **IEEE Communications Surveys & Tutorials**, IEEE, 2020.

Prytz, G. A performance analysis of ethercat and profinet irt. In: **2008 IEEE International Conference on Emerging Technologies and Factory Automation**. [S.l.: s.n.], 2008. p. 408–415. ISSN 1946-0740.

REZAI, A.; KESHAVARZI, P.; MORAVEJ, Z. Key management issue in scada networks: a review. **Engineering science and technology, an international journal**, Elsevier, v. 20, n. 1, p. 354–363, 2017.

ROBERT, J. et al. Minimum cycle time analysis of ethernet-based real-time protocols. **International Journal of Computers, Communications and Control**, v. 7, p. 743–757, 08 2012.

SAARINEN, M.-J. O. Related-key attacks against full hummingbird-2. **IACR Cryptology ePrint Archive**, v. 2013, p. 70, 2013.

SATO, T. et al. **Smart grid standards: specifications, requirements, and technologies**. [S.l.]: John Wiley & Sons, 2015.

SHAABAN, A. M.; KRISTEN, E.; SCHMITTNER, C. Application of iec 62443 for iot components. In: SPRINGER. **International Conference on Computer Safety, Reliability, and Security**. [S.l.], 2018. p. 214–223.

SHAHZAD, A. et al. Real time modbus transmissions and cryptography security designs and enhancements of protocol sensitive information. **Symmetry**, Multidisciplinary Digital Publishing Institute, v. 7, n. 3, p. 1176–1210, 2015.

Shi, W. et al. Edge computing: Vision and challenges. **IEEE Internet of Things Journal**, v. 3, n. 5, p. 637–646, Oct 2016. ISSN 2327-4662.

SOMMESTAD, T.; ERICSSON, G. N.; NORDLANDER, J. Scada system cyber security—a comparison of standards. In: IEEE. **IEEE PES General Meeting**. [S.l.], 2010. p. 1–8.

SONG, H. M.; KIM, H. R.; KIM, H. K. Intrusion detection system based on the analysis of time intervals of can messages for in-vehicle network. In: IEEE. **2016 international conference on information networking (ICOIN)**. [S.l.], 2016. p. 63–68.

VOLKOVA, A. et al. Security challenges in control network protocols: A survey. **IEEE Communications Surveys & Tutorials**, IEEE, v. 21, n. 1, p. 619–639, 2018.

WIECZOREK, F. et al. Towards secure fieldbus communication. In: SPRINGER. **International Conference on Computer Safety, Reliability, and Security**. [S.l.], 2012. p. 149–160.

APÊNDICE A — RESUMO EXPANDIDO

Com a iminente necessidade de se aumentar a produtividade das plantas industriais, o uso de equipamentos que automatizem os processos está cada vez mais frequente. Estes equipamentos — chamados controladores lógicos programáveis (PLCs) — podem estar localizados em locais de difícil acesso, ou geograficamente distantes, o que torna necessário o uso de redes de comunicação para monitorar o bom funcionamento dos processos. Este tipo de rede é denominado de *supervisory control and data acquisition (SCADA) network*. O processo é monitorado e controlado através do software supervisório, chamado de *software* SCADA, que está conectado aos dispositivos através desta rede.

Contudo, devido à magnitude e à importância das atividades que se utilizam destes dispositivos, é absolutamente crítico que sejam considerados aspectos de segurança desta rede. Diversos protocolos e padrões utilizados na indústria para especificar a comunicação entre dispositivos não possuem quaisquer mecanismos de segurança para evitar interferências maliciosas. Um atacante que venha a obter acesso à rede pode facilmente alterar comandos de entrada e saída (I/O) destinados a receber informações do processo ou atuar sobre ele, causando danos irreparáveis tanto em custo quanto em vidas humanas. Notavelmente, os padrões trabalhados neste estudo (EtherCAT e *Open Platform Communications Data Access* (OPC DA)) não possuem qualquer tipo de mecanismos de segurança em suas especificações.

Existem opções atualmente utilizadas em instalações modernas, como o uso de módulos externos de criptografia e o uso de padrões mais modernos, como OPC UA, que especifica mecanismos de segurança. Contudo, essas opções inevitavelmente implicam na aquisição de novos equipamentos e/ou investimento pesado em desenvolvimento. Isso vai de encontro ao *modus operandi* das próprias fábricas, que continuam utilizando equipamentos obsoletos para não realizarem grandes investimentos, ou seja, não é uma abordagem factível na prática. Notavelmente, a empresa fabricante dos equipamentos utilizados neste trabalho começou apenas recentemente a desenvolver equipamentos que trabalhem com OPC UA, o que mostra pervasividade de equipamentos legados na indústria. Outros tipos de métodos podem ser utilizados desde que preservem as propriedades de tempo real da aplicação. Ainda, a escalabilidade é um dos requisitos mais importantes, pois estas propriedades precisam ser mantidas mesmo em aplicações de grande porte.

Este trabalho propõe um mecanismo de segurança baseado em *software* que pode

ser facilmente implantado, não acarretando grandes custos e, portanto, sendo principalmente voltado para plantas que continuam utilizando equipamentos obsoletos. A abordagem escolhida é o uso de um algoritmo de criptografia com autenticação das mensagens, uma abordagem que não havia sido estudada de acordo com a pesquisa bibliográfica conduzida neste trabalho.

O problema foi definido como sendo garantir a segurança das mensagens de entrada e saída dentro da rede, e foi subdividido em duas partes: a comunicação entre PLCs e suas estações de I/O e a comunicação entre PLCs e o *software* SCADA. Um algoritmo foi escolhido com base em suas propriedades para a implementação da prova de conceito, embora a proposta seja aplicável para outros algoritmos de criptografia que garantam a segurança das mensagens. O algoritmo escolhido foi o Hummingbird-2.

Para avaliar o desempenho do mecanismo proposto, foram propostas métricas para cada uma das entidades envolvidas, i.e., as estações de I/O, os PLCs e o *software* SCADA. Para as estações de I/O, a métrica proposta é uma métrica de latência baseada no tempo de ciclo do *firmware*. No caso dos PLCs, foram propostas uma métrica de latência baseada no tempo de ciclo da aplicação e uma de *jitter*. Finalmente, para o *software* SCADA, duas métricas baseadas na latência e na taxa de comunicação dos dados foram propostas.

Os experimentos foram conduzidos considerando o tamanho de uma aplicação de médio porte, usando a rede da universidade para simular uma situação de tráfego que se aproxime da realidade. Para avaliar as estações de I/O, foram conduzidos experimentos envolvendo encriptação e decriptação de suas portas de I/O. No caso dos PLCs, foram conduzidos experimentos aumentando-se o número de variáveis sendo encriptadas e decriptadas. Por fim, para avaliar o *software* SCADA, foram conduzidos experimentos onde o *software* precisava decriptar um total de 20 mil variáveis inteiras de 16 bits cada, o que equivale a uma aplicação de médio porte.

Analisando-se os resultados obtidos, pode-se afirmar que a solução proposta é viável para a maior parte das aplicações. Isso é válido tanto para aplicações pequenas e com tempos de ciclo bastante pequenos, como aplicações de *motion control*, quanto aplicações de até médio porte. O aumento percentual no tempo de ciclo da estação de I/O foi 7,5% no pior caso. No PLC, para um grande conjunto de 1024 variáveis analógicas (16 384 bits), houve um aumento de 20 ms, o que é considerado um bom resultado dado que aplicações grandes tendem a ter tempos de ciclo maiores. Finalmente, os experimentos com o *software* SCADA apresentaram aumentos de 5,6% para o tempo de sincronização e 8,5% para o tempo de resposta, no pior caso, para uma aplicação de 20 000 variáveis

inteiras de 16 bits (totalizando 320 000 bits).

A principal vantagem da abordagem proposta em termos de desempenho aparece justamente quando se considera a escalabilidade do método. A abordagem utiliza um método de computação distribuída onde cada PLC e cada estação de I/O criptografa suas próprias variáveis. Isso significa que o envio destas variáveis criptografadas para o *software* SCADA causa um impacto muito baixo em termos de desempenho, o que torna a solução proposta altamente escalável. Notavelmente, o aumento de tempo em função do aumento do número de variáveis apresenteu crescimento linear.

O método proposto pode, então, ser utilizado para se aumentar o nível de segurança de plantas que não possuem interesse em realizar altos investimentos em equipamentos novos no momento.

Sumarizando, as principais contribuições do trabalho são: projeto, proposta e estruturação de um novo mecanismo de defesa baseado em software que apresenta alta escalabilidade e apresenta um baixo impacto no desempenho, com crescimento linear; implementação e análise de viabilidade do uso de um algoritmo de criptografia com autenticação no contexto de redes industriais e sistemas de tempo real; e, por fim, incentivo a discussões e estudos no âmbito da segurança em redes industriais.