Practical Encryption Gateways to Integrate Legacy Industrial Machinery

Dissertation

zur Erlangung des akademischen Grades
Doktoringenieur (Dr.-Ing.)

vorgelegt an der
Technischen Universität Dresden
Fakultät Informatik

eingereicht von

Dipl.-Inf. Tim Lackorzyński

Betreuender Hochschullehrer:	Prof. Dr. rer. nat. Hermann Härtig
	Technische Universität Dresden
Zweitgutachter:	Dr.-Ing. Andreas Bluschke
	Signify Netherlands B.V.
Fachreferent:	Prof. Dr.-Ing. habil. Martin Wollschlaeger
	Technische Universität Dresden

Statusvortrag:	27. 11. 2020
Abgabe:	25. 01. 2022
Verteidigung:	24. 05. 2022

Contents

List of Figures

List of Tables

1 Introduction

The manufacturing of goods on an industrial scale is a very resource intensive and complex process. Huge investments in machinery and equipment as well as skilled personnel are necessary. Hence, there is always the incentive to increase efficiency and drive down costs by automating these processes. Automation, in that sense, means to reduce necessary human intervention as much as possible, while at the same time increasing the speed and reducing the costs of the production process. The term industrial automation (IA) subsumes all efforts in that direction, may they be concerned with the direct manufacturing process or more auxiliary processes, like quality control and handling of materials. Industries comprise of vastly different branches, that can be roughly classified into two groups. The first comprises of all types of manufacturing of discrete goods from cars and household appliances to food items and beverages. The second group is concerned with the execution of some kind of continuous process, like wastewater treatment, steel production or the chemical refinement of oil and gas. Yet, IA technologies are also applied in fields that exist mainly outside of classic factory setups, like transportation, building automation and utility distribution.

Industrial machines have an immediate influence on the physical world. They work directly with raw materials or discrete physical objects, that have continuously changing states. Human operators work very close to these processes. Therefore, the machinery as well as the goods and materials being processed must be monitored and controlled. Any deviation from defined behavior might have a direct negative impact on the safety of the workers or might result in damage of the machinery, the buildings or even the surroundings of the factory.

So-called industrial control system (ICS) are employed for that purpose. The term ICS comprises systems that control and steer industrial processes. Today, these are mainly digital and distributed systems that consist of nodes that range from simple I/O modules to powerful general purpose computers that are connected by a variety of different network protocols.

Currently, the field of IA finds itself in a phase of change. Control systems that were formerly fixed and isolated within the factories are now being extended with new technologies that implement new use cases. This trend is variably called Industry 4.0, the industrial Internet of things (IIoT) or more generically smart manufacturing. Factories are becoming so-called smart factories.

What this actually means in practice, is still somewhat fuzzy and is mainly described with a lot of ideas on a conceptual level. Efforts are generally aimed at increasing the efficiency as well as the flexibility of the production process and to open the field of IA up for new business models that are mainly transferred from the realm of service-based and data driven information technology (IT). This includes concepts like virtualization of factory infrastructure (cloud-based or edge computing) as well as the goods themselves that are being produced (so-called digital twins).

Factories will also be augmented with a vast number of new sensors that produce lots of data that then is supposed to be used for big data, cloud and machine learning applications. More concretely, applications like so-called predictive maintenance are proposed, where from sensor data the health of individual machines can be derived, so that maintenance breaks can be scheduled before the machine actually breaks. Another more concrete idea is mass customization, where each produced good is made to customer specifications. This involves the constant reconfiguration of machines as well as the notion of a machine, which good is being manufactured on it in that moment. In summary, industrial networks will transform to very heterogeneous, partly virtualized and service-based networks where a variety of use cases is executed in parallel.

The transition from the old model of IA to the new smart manufacturing model will present certain challenges. These arise mainly from the fact that IA infrastructures will continue to exist and will not entirely be replaced but instead be augmented with IT technologies and applications. This combination then constitutes a kind of clash of world views insofar that while ICSs and IT systems are comparable, they yet stem from wholly different ecosystems. ICSs were designed by mechanical and electrical engineers, not computer scientists. This led, for example, to full stack network protocols starting from the physical layer and including an application layer for specific tasks and use cases. IA in general focuses on purely functional and safety aspects and once a system is up and running inside specifications, it is usually never changed. This is in stark contrast to the classic IT world view, where technologies and especially networking is mainly thought in layers of abstraction. Layered structures of different protocols build on top of each other to offer the highest degrees of interoperability. Lower layers are designed to be as generic as possible, while specialization to use cases only happens in the upper layers. Furthermore, the IT world is marked by short life-cycles and constant change. Software as well as hardware is frequently updated, upgraded or exchanged.

This transitioning phase will be long and slow, as industrial machinery and equipment typically has life-cycles of multiple decades and generally high investment costs. Additionally, due to the ever present safety concerns, factory operators are a very conservative crowd, that only replace components when absolutely necessary.

While new built factories will feature up-to-date and well integrated systems, old ones will still exist aplenty. These old factories, to stay competitive, will nonetheless have to be upgraded in some way that allows their legacy IA infrastructure to integrate with the new services and components from the IT world so that the old factories too can benefit from the new approaches. Hence, it will be a challenge in the future to manage the concurrency of these vastly different systems within a factory.

One of the big challenges within this space will be the security of such a mixed system where legacy and new components interact. As already described, IA and IT are underpinned by different fundamental assumptions and especially the notion of security is different. IA is primarily concerned with the safety of the physical processes, meaning the focus is on the functional correctness of all involved components as well as defined and rapid behavior in case of malfunction. On the other hand, the world of IT is mainly a software-based world, where IT security is concerned with the protection of data in transit and at rest.

The big challenge will be in balancing out these different world views, as currently, industrial equipment has serious IT security-related shortcomings as many studies and successful attacks have shown in the recent past [41, 54, 103, 88, 59]. This is not limited to legacy components but is also prevalent in new components, that are still designed with the old IA assumptions in mind [105].

This thesis wants to contribute by proposing a security solution that adheres to the assumptions of both worlds and negotiates a viable compromise. It is based on the concept of industrial gateways. These network boxes are separate pieces of hardware that are placed in front of a machine that needs protection from network-based threats. This may be the case, because

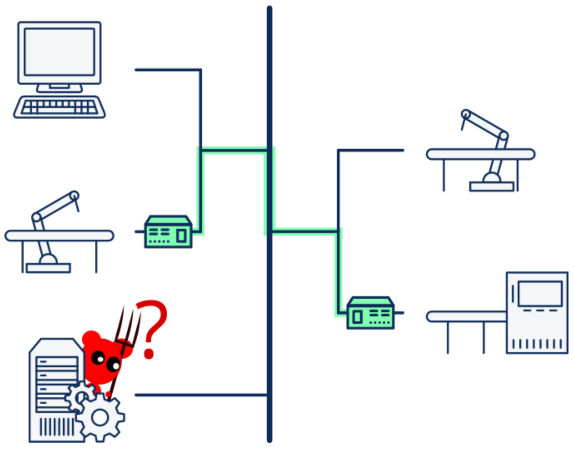


Figure 1.1: Industrial network with two gateways enabling secure communication.

that legacy machine was identified to be a possible attack target and needs to be isolated from the rest of the network or because the machine itself is not trusted and the rest of the network needs to be separated from that device. Since gateways are put between machine and network, they can control all communication flowing from and to the device, effectively separating it from the network. They can also establish encrypted tunnels between each other to enable secure end-to-end communication between devices. Fig. 1.1 shows a small example network with two gateways attached to two machines. A secure channel isolates their communication and a potential attacker within the network cannot observe the exchanged communication data.

The following chapter will introduce the concept of industrial gateways further. It will also introduce in more detail the old model of IA as well as the new model of smart manufacturing. Based on that, challenges that arise from the difference in security-related assumptions between IA and IT will be discussed. The chapter will also discuss related work. Finally, it will introduce the specific research question of this thesis and will give its scope.

2 Background and Related Work

This chapter first introduces the field of industrial automation (IA) and then discusses important IT security-related challenges that result from the shift from the old industrial control systems (ICSs) to the highly integrated smart factory systems of the future in Section 2.1. From these discussion, the research questions of this thesis is then derived in Section 2.2.

2.1 Industrial Automation

IA is currently in a state of transformation. Paradigms and approaches from the information technology (IT) world are currently being introduced into the ecosystem of IA to create new applications and services. This includes networking technologies and approaches towards IT security that are sometimes at odds with the paradigms from the established legacy ICSs.

In order to discuss this in detail, this section will first give a short introduction to the history of ICS. It will then follow up with an overview on the general trends and envisioned innovations that mark the somewhat vague goals ICSs are supposed to transition towards. Finally, IT security challenges that arise during this transformation process will be discussed in detail.

2.1.1 Past and Present

Systems to control industrial processes began with discrete electric controllers attached to machines that steered one single control loop. A sensor monitored a specific physical parameter and a corresponding actor regulated the physical process based on predefined limits of that parameter. As discrete controllers grew in number within the factory, they were integrated first into bigger control panels and then into mostly one facility-wide control room. This allowed for centralized control. Yet, every control panel still had its own connection to each physical device it monitored and controlled. This meant a lot of cabling and consequently led to further integration. As a result, distributed control systems (DCSs) evolved, that decoupled signal processing and signal transport. Control units were distributed within the factory and connected to so-called fieldbus networks that shared the networking infrastructure, greatly reducing necessary cabling. Thus, intermediate communication networks were created. They also made it possible to integrate process control with other computer systems that had different tasks, like production control or more sophisticated event logging. Vendors started to design proprietary communication protocols for command and control within these networks. Many fieldbus protocols were developed in time by different vendors individually, like Controller Area Network (CAN) bus [66], Modbus [104] or PROFIBUS [64]. Over the years, these were gradually standardized and opened up to other vendors.

As complexity within factories grew, control was more and more automatized. Programmable logic controllers (PLCs) were introduced, that could control and steer physical processes based on software they were programmed with. These controllers can be put close to the physical process and manage the control loops automatically, while themselves being managed remotely

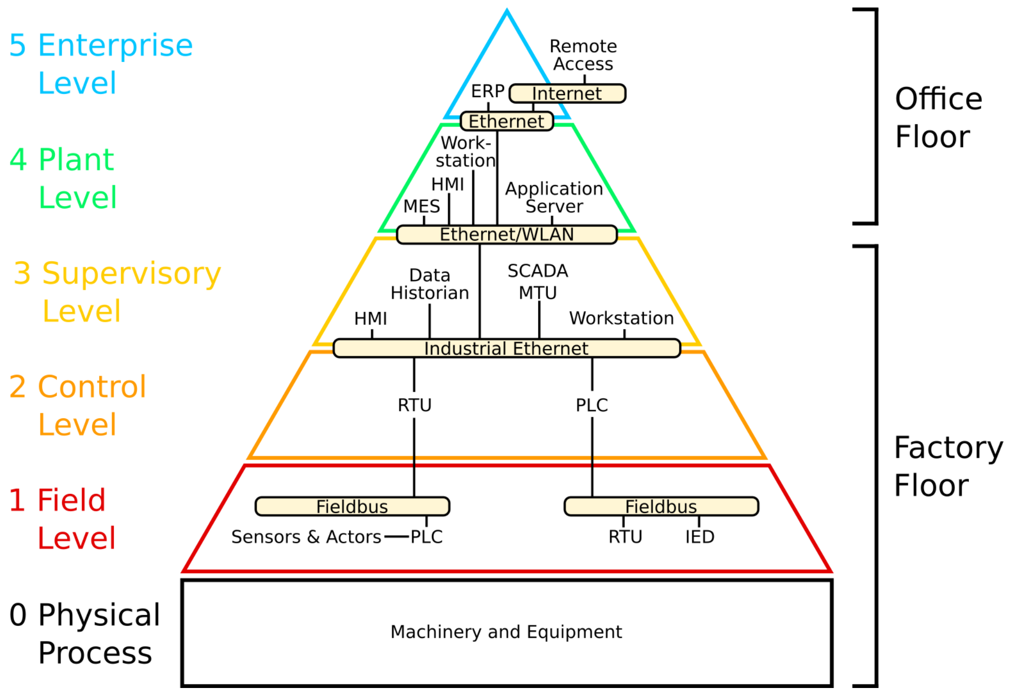


Figure 2.1: Pyramid model of information systems within a contemporary factory.

over the factory network. Then, further abstraction layers for supervision, control and scheduling were introduced and combined into a holistic control system architecture called supervisory control and data acquisition (SCADA). SCADA systems incorporate computers, network protocols and controllers and create a high-level process control and factory management system.

Today, the lines between DCSs and SCADA systems are blurring and the general technology landscape within a contemporary factory is even more diverse. Factories incorporate a heterogeneous set of information systems, which differ in aspects like requirements, intents and purposes or applied technologies. These systems can be broadly separated into the classic ICSs, which control and steer the physical production processes and which include the aforementioned DCSs and SCADA systems, and the from an IA standpoint more modern IT systems that provide additional software-based services that focus on data processing.

The automation pyramid is a way of modeling the complexity of these different systems as well as a unifying view on factory information systems in general. There are many different approaches on how to structure such a pyramid. A good overview can be found in [94]. All models have in common, that they try to classify and group the different technologies found in a factory into a certain number of levels, which constitute different grades of abstraction of processed data or information services provided.

Siepmann proposed a model that represents a canonical view on the matter [107]. His automation pyramid consists of 6 levels, as shown in Fig. 2.1. These are briefly introduced in the following:

Level 0: Physical Process Layer 0 does not represent any information system but the actual physical production processes as well as the machinery that is being steered by the information processing systems in the pyramid hierarchy above. This layer is the source of data that needs to be processed as well as the ultimate target of all controlling efforts. It is generally included to give a complete picture or rather to give the model a holistic view.

Level 1: Field Level Devices on this level have actual contact to the physical process. Sensors monitor the process by measuring physical quantities, e. g. temperatures, pressures or filling levels, while actors actively steer the physical process in the form of valves, conveyor belts or tools processing a workpiece. Fieldbus networks link these actors and sensors to controllers upstream on the next level.

Level 2: Control Level Control devices on this level implement closed control loops, where sensor data is evaluated and subsequent commands are issued to actors. These control devices can have different names, yet basically perform the same tasks. The most often used term is PLC. At first, PLCs were mere integrated circuits (ICs) designed to execute specific logic functions necessary to implement the control loops using electronic components, via e. g. relays, switches, and mechanical timers and counters. Today, these controllers have evolved to fully embedded platforms with the capability of controlling very complex processes and as such resemble more and more ordinary computers or computing nodes from the IT world. They are used substantially in SCADA systems and DCSs. In SCADA terminology these controllers are called remote terminal units (RTUs) or remote telemetry units. RTUs are mainly used to control remote stations as this was the initial use case for SCADA. Yet, PLCs at the local control level are also called RTUs in the SCADA context.

Level 3: Supervisory Level The third level is populated with various devices used for supervision and management of the networks on the lower levels. Control servers, like master terminal units (MTUs) in SCADA terminology, host the software that manages the lower level control devices. Human-machine interfaces (HMIs) serve as points for visualization of as well as interaction with the managed networks for factory operators. They may be stand-alone or integrated with control servers. Workstations offer further services. For example, logging is done by a so-called Data Historian. Devices are connected using some industrial Ethernet protocol. Industrial Ethernet is an umbrella term for communication protocols in the industrial environment that are all based on standard Ethernet and which have various use case-specific extensions. While this can sometimes mean merely standard Ethernet with ruggedized connectors, the term is generally understood to subsume various protocols extending Ethernet to provide low latency as well as real-time guarantees. Standard Ethernet, on the contrary, does not guarantee timely delivery of datagrams. It is probabilistic, meaning that data is written opportunistically to the bus and datagrams are handled in a best effort manner. This results in an unbound upper worst case delivery time, which is unacceptable for many industrial use cases, where defined reaction times to events are important to avert damages in the physical world. Today, many different commercial protocols exist, e. g. EtherCAT, EtherNet/IP, Modbus TCP, Profinet, SERCOS III [64], Ethernet Powerlink [48] or TTEthernet [29]. These protocols are all wire-bound. Wireless protocols are not generally employed on this level, because they lacked reliability in the past. Yet, with new research in the direction of real-time wireless communication and Industrial 5G, this is beginning to change slowly [116].

The systems on levels 1, 2 and 3 comprise what is typically referred to as ICSs. They manage the operational aspects of a factory. Other terms often used to describe these three levels are *factory floor* or *shop floor*. In contrast, the following two levels 4 and 5 consist of systems that are concerned with the general management and planning of a plant. These systems are common IT systems and both levels are often subsumed under the term *office floor*.

Level 4: Plant Level The most important and directly factory-related system on this level is the manufacturing execution system (MES). It tracks and monitors the production process or, in more general terms, the transformation of raw materials that enter a factory to the finished product leaving it. Other systems found on this level are the typical systems from the office IT environment, like various application servers including email or print and further staff operated workstations and HMIs. The networks on this level consist of ordinary standard wired and wireless Ethernet.

Level 5: Enterprise Level Factories are typically run as enterprises. There are many more auxiliary assets that need to be managed, and which are not directly connected to the production process. Enterprise resource planning (ERP) tools manage whole business processes. They include software tools for gathering, managing and interpreting various sources of data within the whole enterprise. This does not only include data from machinery and equipment but also for example human resources, logistics and finance. ERP systems may connect factories and offices at various sites and are hence connected to the Internet. They constitute the outermost layer of the automation pyramid. In this model, remote access to assets within the factory is only granted on this level, resulting in many layers of abstraction between an individual industrial machine and the Internet.

This short overview can only give a general feeling about the complexities of the various computer systems and networks within factories. More in-depth introductions to IA are plentiful, for example [52] or the respective Sections 2 in the "Guide to Industrial Control Systems (ICS) Security" from the United States National Institute of Standards and Technology (NIST) [112] and the German "ICS-Security-Kompendium" of the German Federal Office for Information Security (Bundesamt für Sicherheit in der Informationstechnik) (BSI) [50].

To summarize, even "old" or non-smart factories are already very complex entities. ICSs from the world of IA and common IT systems already run in parallel in the same setting, albeit on different levels. This means that different assumptions, requirements and use cases have to be satisfied at the same time. For example, there are very heterogeneous networking technologies present. On the factory floor small data packets only consisting of a couple of bytes must be transmitted in real-time, while on the office floor, big data streams are processed in a best effort manner. Risk assessment is done separately as well. As a result, corresponding protective measures must also have different focus. On the factory floor the main goal is to ensure continuous execution of the production process, while on the office floor the main goal is to protect data at rest and in transit. These existing conflicts will only aggravate in the future and are discussed in more detail below.

This short overview is a general description of how factory systems used to be organized and many factories still operate like that today. However, this will change for future factory systems. The next section will give a short introduction into how the field of IA will develop according to current conceptions.

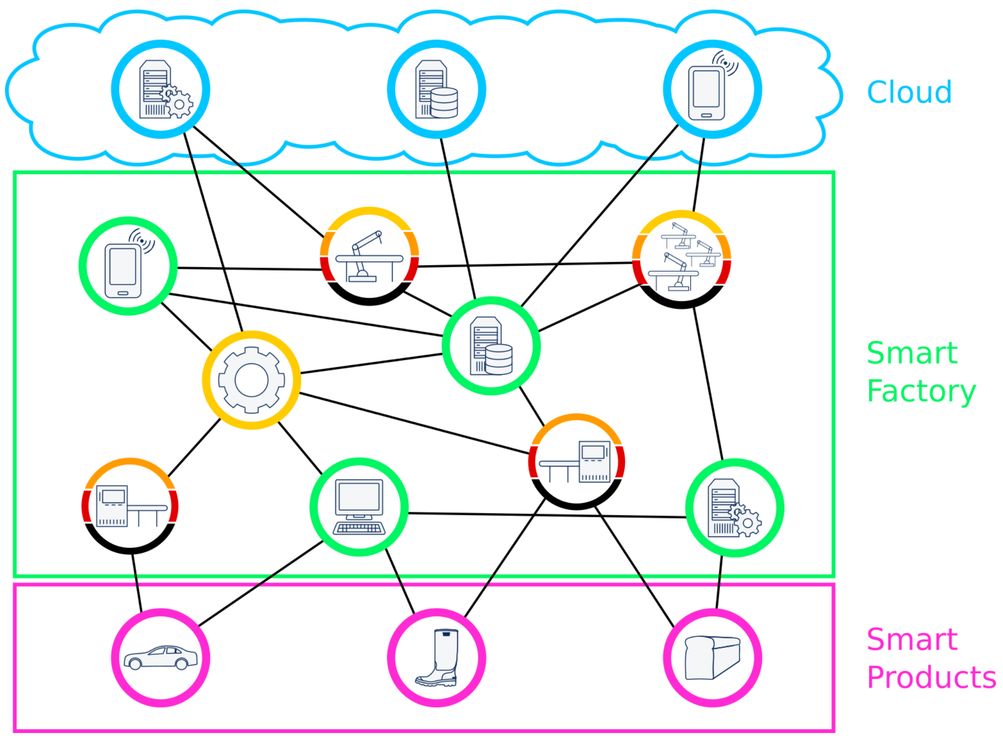


Figure 2.2: Model of a highly integrated smart factory.

2.1.2 Future

Future trends of IA are variably conflated under terms like Industry 4.0, the industrial Internet of things (IIoT) or smart manufacturing. They basically mean a higher grade of digitalization within the classic manufacturing and process industries beyond the state, that was described above [125]. The general aim is to further integrate ICSs with technologies from the data-driven and service-oriented IT world, so that new paradigms, use cases and modes of operation become possible. Focus is put on an increased interconnectivity of components and formerly isolated systems so that more information can be collected and analyzed, e. g. from the production process, the involved machinery or the goods that are being produced. For example, it shall become possible for subsystems of the factory to automatically diagnose and optimize themselves. To that end, machines, equipment, logistics and products shall be able to communicate directly to each other without a centralized command and control structure. Human operators shall be given more advanced technical assistance by providing real-time data using augmented reality interfaces. Fully virtualised factories as well as the products themselves shall exist as data models and as so-called digital twins. More interconnection of distributed production facilities and a decentralization of planning and controlling shall become possible across the whole value chain, spanning different stakeholders and the whole life-cycle of a product. More information as well as a more precise control over the production facilities shall enable functions like predictive maintenance (scheduling repair intervals for machines before they break) and mass customization (each product is produced to custom specifications).

As of yet, these trends and ideas are mostly buzzwords and academic prototypes. Concrete commercial concepts, technologies or products do not exist. However, they show a general direction, where innovation efforts are heading and some more general probable characteristics of future industrial networks can be derived from that.

Factories will shift from the monolithic automation pyramid model introduced in the previous section to a dynamic model with a flat hierarchy, where the lower levels of the pyramid are abstracted away into autonomous function blocks. This is a process that has already been going on for some time. Even contemporary production machines subsume the first 2 to 3 levels and only expose graphical or remote management interfaces that can be accessed via a directly attached terminal or via network. They execute programs written in high-level languages or derive orders of work steps from model files. The control loops between sensors and actors are managed automatically.

Fig. 2.2 shows how such a future industrial network will look like. Function blocks are connected to each other within a flat, general purpose network and more or less offer their services independently. These blocks can represent physical entities like individual machines or whole production cells, that subsume one or more manufacturing steps where multiple machines work together and which share certain characteristics and requirements, like for example the need for real-time communication. But, these blocks can also represent IT systems as well as purely virtual entities, like the aforementioned digital twins or even cloud-based services. In short, factories will interconnect physical, virtual and remote entities.

In reality, however, the full realisation of these ideas is far away. The near future will be marked by a transitioning phase from the old strictly hierarchical pyramid model to the new dynamic and heterogeneous Industrial 4.0 model. This phase will, due to the typically high investment costs and long life-cycles in the industrial environment, be very long. As the focus shifts from mainly isolated systems to highly integrated ones, challenges will arise from the fact, that assumptions and requirements from the involved legacy components and the new systems will have to be mediated. For example, legacy production systems will persist for the foreseeable future, while networking technologies and paradigms will evolve faster. As a consequence, active steps must be taken to integrate the old systems into future networking environments. The next section will discuss some of the security-related challenges that are important in this context.

2.1.3 Transitioning and Challenges for IT Security

The field of IA finds itself in a transitioning phase from the old and rather fixed automation pyramid model to the new dynamic and heterogeneous world of Industry 4.0. At the same time, the networks of these factories as well as the IT security measures within these networks must evolve as well. The fundamental underlying problem of this transition is, that old ICSs will not be instantly replaced by new service-based IT systems. Instead both worlds will rather merge together and form a hybrid environment where requirements from both sides need to be fulfilled at the same time. And as the effectiveness of IT security measures in general highly depends on the sets of assumptions and requirements they are based on, the concrete measures rolled out in future factories must also evolve. Therefore, conflicting requirements create concrete IT security challenges, that must be faced so that this transition may be successful.

These transitional challenges have been extensively studied in the past [46, 96, 52, 37, 61, 31]. This work will only highlight selected aspects as they are important for the remainder of this thesis. Additionally, the BSI as well as the NIST have published comprehensive guides on security in ICSs that also detail these challenges [50, 112].

Many of the challenges discussed below are either raised in the first place or aggravated by the fact that industrial machinery typically has very long life-cycles. Machinery and equipment has high investment costs and their general application does not change. Lathes, presses or conveyor belt systems remain useful independently of the current product, that is being produced on them. Anecdotal evidence suggests, that devices have lifetimes of many decades. Yet, concrete numbers are hard to find, probably due to the fact that these are treated as business secrets by the factory operators. NIST states in their guide the average lifetime of ICSs to be 10 to 15 years.

A study by Krejčí and Mazouch in 2015 tried to measure the age of machinery and equipment in the Czech Republic indirectly by inferring from publicly available bookkeeping data of companies [76]. They found an average age of around 8 years and a maximum age of about 15 years and noted that lifetime is generally increasing. In an older study from 2008 Erumban estimated machine and equipment age for the Dutch manufacturing industry [47]. He found an average lifetime of Dutch machinery of around 30 years, while also noting that other estimates from other countries arrived at lower numbers of around 12 to 21 years. To summarize, industrial machines and equipment have indeed long lifetimes. Averages of 10 to 15 years mean individual devices can be much older. This fact must always be considered in the following discussions.

Threat Model From a security standpoint, ICSs used to be designed following two principles or basic assumptions. The first was *perimeter security*, which means that systems only run in areas that are physically separated from the outside world and virtual access is only possible remotely and indirectly via systems on the Enterprise Level of the automation pyramid. The second was the principle of *security by obscurity*. This meant that attackers could not launch attacks because they missed necessary information. On the one hand, ICSs used to be implemented using specialized hardware and proprietary protocols, that were largely inaccessible to the “leisure time” hacker, that was assumed to be the prevalent attacker type in the past. On the other hand, the inner workings of individual factories, like for example how the network was structured, was also assumed to be a secret because only authorized personnel could enter the factory as perimeter security also included physical boundaries around the buildings and entrances protected by security guards. This resulted in the assessment, that attacks on ICSs were largely impossible. Consequently, no IT security measures were taken and no IT security management was implemented. The possibility of insider attacks, meaning malicious acts by factory workers, was largely ignored.

However, the transformation towards heterogeneous smart factory systems makes this view obsolete and even dangerous. As factory networks are being augmented with new devices and service-based IT systems, connectivity within the factory across hierarchy levels as well as with the outside world will also increase greatly. Additionally, this development transforms the previously proprietary industrial networks to more generic and low-cost IT networks based on standardized Ethernet and the Internet Protocol (IP). The results of this transition are highly heterogeneous networks where devices from different vendors coexist. At the same time, legacy ICS components will still be part of those networks. Yet, as the assumptions that underpin their security-related design principles do not hold anymore, this effectively opens them up to attacks. As a result, new types of threats and attackers arise. One class of attacks is active attacks on the network level and as a consequence, the internal network cannot be trusted to be secure anymore. Traffic therein must be protected as if it was flowing through the Internet. This concept is called zero trust networks and broadly states, that security measures should be introduced inside local networks to counter these types of threats [106].

Protection Goal Hierarchy With the evolving threat model, new security measures must be designed and implemented. Any design of a security measure always begins with the definition of protection goals, meaning against what type of threat a system shall be secured. In the following, we will focus on the main protection goals of confidentiality, integrity and availability. More protection goals exist, that allow more sophisticated descriptions of security goals, yet the three will suffice to describe the shifting trends we want to focus on here.

Classic ICSs focused mainly on safety, meaning risks were mostly seen in systems malfunctioning or rather behaving outside of specifications. As ICSs directly manage and control processes in the physical world, their malfunction can have direct impacts on the health and safety of humans and the environment or it can have direct financial impact through production loss or damage of property. From that thinking measures were derived that focused on ensuring the correct and constant functioning of systems, while, as described in the previous segment, adversarial behavior was not factored in at all.

The only concern for the systems and networks of the factory were with the reliability with which they provided their services. This can be translated into a very strict hierarchy of protection goals, where the most important protection goal is that of availability.

Formally, the goal of availability describes the aim that necessary services in a network shall be available to authorized entities when needed and in sufficient quality. In a more practical sense this means, that ICSs can only ensure the safety of the physical processes they manage, if the network infrastructure transmits control data and commands reliably. When real-time applications are concerned, this requirement extends to the timeliness of message delivery. This is especially true for e. g. emergency messages.

The other two protection goals are integrity and confidentiality. They respectively mean that data shall not be tampered with, or that this can be detected, and that data is only readable to authorized entities. Both goals are absolutely subordinate to the first, as their implementation typically makes it necessary to add functionality to the network and create management processes, for example by adding message authentication codes to messages or by creating processes to authorize devices as well as staff members operating those devices. Additionally, these functionalities in themselves carry a risk of malfunction, which makes them into an additional source of risk towards the primary protection goal of availability. Also, integrity and confidentiality protection mechanisms address threats by attackers which, as described above, were not considered in the design of ICSs, anyhow. These reasons led to a clear cut protection goal hierarchy, where availability comes first, with integrity and confidentiality being a distant second and third. This is abbreviated by the term “A-I-C”.

For IT systems, on the other hand, there is no strict general hierarchy assumed and the protection goals rather form a triad, where the individual goals must be weighted according to the specific use case. Yet, from a networking perspective, the biggest threats are typically active attackers that try to gain information or manipulate data. Consequently, confidentiality and integrity are weighted higher. Availability is ranked least important, simply due to the fact that threats to the network infrastructure typically cannot be met on the protocol level anyhow. This results in a less strict, but still existing protection goal hierarchy abbreviated with “C-I-A”.

Since industrial networks are not replaced, but augmented with IT components and services, both hierarchies necessarily result in protection goal conflicts that must be reconciled when designing and operating either wholly new systems or mere extensions for legacy systems.

As an example for these conflicts, the need to protect data in transit within the local network might serve. Standard procedure for IT systems is to encrypt data in transit so that attackers who may listen on the line may not gain valuable information. Yet, this introduces additional soft- or even hardware components to the existing system. This also results in new requirements for the participating end nodes, like increased demands in processing power as well as new necessary security management processes. All increase the complexity immensely. For example, the employment of security-related software makes patch management necessary. Processes must be introduced to manage and exchange encryption keys, including staff that need to be trained and authorized to handle these systems. Additionally, on a technical level, encryption necessarily increases transmission delays as computational steps are added to each send and receive operation. All these considerations stand in direct contrast to the primary ICS protection goal of availability, as risks and delays to the most vital processes within a factory are increased. How such a conflict is handled, strongly depends on the specific use case and cannot be answered in general terms.

Usability As security measures will be part of future factory management systems, staff must be able to handle the corresponding IT security processes and must be able to implement security policies in practice. Yet, factory operators are not classically trained in IT security matters and prior knowledge on, for example, encryption protocols or software management processes cannot be assumed. Today's corporate IT security tools on the other hand are designed to be used by professionals.

This fact or rather its consequences were demonstrated in a study by Dahlmanns et al. in 2020. The authors found that only 8% of modern OPC UA[1] servers are configured securely although the servers provided state of the art security measures [41]. The insecurity of the tested systems stemmed mainly from wrong configuration choices by the administrators. Additionally, even when factory operators were informed on the vulnerability of their systems, they did not patch their servers. The authors scanned the whole IPv4 address range, so their results hint at an industry-wide problem. As a result, it must be concluded that factory staff today lacks necessary IT security knowledge and that these deficiencies must be confronted on multiple levels. On the one hand, special training in IT security is necessary, but also the security tools must be designed with untrained or non-professional users in mind. Special emphasis must be given on usability concepts that integrate well with existing interfaces and work-flows of the industrial environment. Otherwise, as the study showed, even available security measures will not be implemented in practise.

Static End Points As described above, ICSs are safety critical and malfunctions can lead to direct physical harm. Therefore, after being installed, systems are extensively tested before being used productively and often a certification process is conducted by the equipment provider to proof that the newly installed system works as intended. This may include measuring quality parameters, like thresholds for certain work steps, vibration behavior or signal round-trip times. The systems can even be certified according to standards like IEC 61508, that describe tiers of requirements a system must meet to achieve a certain Safety Integrity Level (SIL) [65].

The goal of all these measures is to reduce the risk of malfunction as best as possible and sometimes they are a prerequisite so that the factory as a whole or the respective organizational unit can get insurance. As a consequence, ICSs in general tend to be very static or fixed

[1] OPC UA - Open Platform Communications Unified Architecture, is a widespread communication protocol used in IA [99].

and factory operators cannot make changes to a deployed system, as this would void said certification or operational guarantees by the vendor.

This is especially true for software. In practise this leads to systems that run software which is not updated at all. From an IT security standpoint, these machines must be considered as very vulnerable end points in the network, especially when considering the average age of machinery and equipment. Recent studies have extensively shown the how vulnerable these systems are [54, 103, 88, 59, 105]. In the past, this was not considered a problem, as ICSs were run in isolation. Yet, with the transformation towards very heterogeneous networks, the conflict between the desired low risk approach of not changing a running system and the IT security approach of constant improvement through security processes must be reconciled.

Performance Somewhat related to the challenge described in the previous paragraph, is the fact, that signal processing times within ICSs are often time-critical and deterministic. This means that the systems run under the assumption, that signal travel times have fixed upper bounds. The introduction of additional security mechanisms to signal processing necessarily increase this delay, as additional computation steps are added to the transmission process. Hence encryption must be optimized for performance as much as possible, so that functional requirements can still be met. Yet, additional security measures will always incur overhead, how efficient they otherwise may be. Whether the gains in security are actually worth the performance penalty is subject to the specific use case and, again, cannot be answered in general terms.

To summarize, the transition of IA is marked by the convergence of very different security paradigms, where requirements of the one may contradict assumptions of the other. Hence, although industrial networks are being augmented with IT systems, IT security logic and approaches cannot be readily applied to ICS networks. And while mature security mechanisms and processes exist in the world of IT, these must be adapted with care to the industrial sphere so that they can be effective.

2.2 Research Question

This section first describes the specific challenge this thesis wants to meet. This is followed by an introduction to so-called industrial gateways and a discussion, why this concept is most suited for the challenges we propose. This is followed by a discussion on previous approaches that also used industrial gateways for network and security-related purposes. Finally, this section will give a scope and introduce the specific topics that are investigated in this work.

2.2.1 Challenge

The previous section showed that the transition of the field of IA towards a dynamic and heterogeneous future offers many challenges. There are many concrete problems that need to be solved. This thesis wants to provide a solution to one specific problem, that revolves around the security of legacy machines.

As explained, industrial machines have long lifetimes and these old devices will still be part of future factories. Yet, from an IT security standpoint, operating these old machines must be regarded as a severe security risk. The software running on-top of them has, with high probability, a comparable age and must be considered outdated. As factory networks are being opened up more and more to Internet-based services, they become participants in a worldwide interconnected network. This makes them vulnerable and a prime target for attackers from all over the world.

Since updating those devices is not an option, other approaches are necessary that make it possible to securely integrate them into future networks. This thesis wants to investigate one such approach by building on the concept of gateways as a way to retrofit legacy machines. Gateways are network components that mediate between different network zones and can control and modify data traffic that flows between them. The concept will be introduced in more detail in the next section.

Furthermore, the here presented solution shall in effect provide an interface between legacy devices firmly rooted in the classic IA world and the IT-centric world of future service-based networks. Consequently, the transitional challenges discussed above must be acknowledged and factored into our design. They are translated in the following into requirements that our solution shall fulfill.

These requirements are:

1. **Threat Model** The heterogeneity of smart factory networks makes the assumption of perimeter security obsolete. Therefore, the local factory network must be assumed insecure and even malicious.

2. **Protection Goal Hierarchy** Although our solution will mainly revolve around protecting network data of legacy machines, the importance of the protection goal of availability must be respected.

3. **Usability** As our solution is to be used in an industrial environment, where staff may not be trained in IT security matters, it must be easy-to-use.

4. **Static End Points** For the various reasons stated above, the legacy machines we want to protect cannot be assumed to be modifiable in any way.

5. **Performance** As networking performance is important for the correct functioning of factory systems, our network-based solution must incur as low overheads as possible.

6. **Transparency** As industrial network protocols are very diverse, our solution shall aim to be as generally applicable as possible.

The last requirement is only indirectly derived from the discussions above. As explained, past industrial networks were based on many different proprietary fieldbus protocols. Providing solutions for each of those protocols is not an option. Instead, a generally applicable solution shall be pursued, that can secure devices irrespective of what protocol they use.

Industrial protocols are now in the process of being replaced by updated versions that are all based on Ethernet technology. Fig. 2.3 shows the distribution as well as the growth of different protocols within the industrial landscape. While legacy fieldbus systems still have a significant share, Ethernet-based systems represent the growing majority. And although wireless technologies also grow considerably, they do not seem to become dominant for the foreseeable future. The simple reason for this is that Ethernet technology is cheaper, very reliable and more versatile compared to the old fieldbus systems. Ethernet-based protocols all have in common that they are compatible to and hence can be integrated with standard

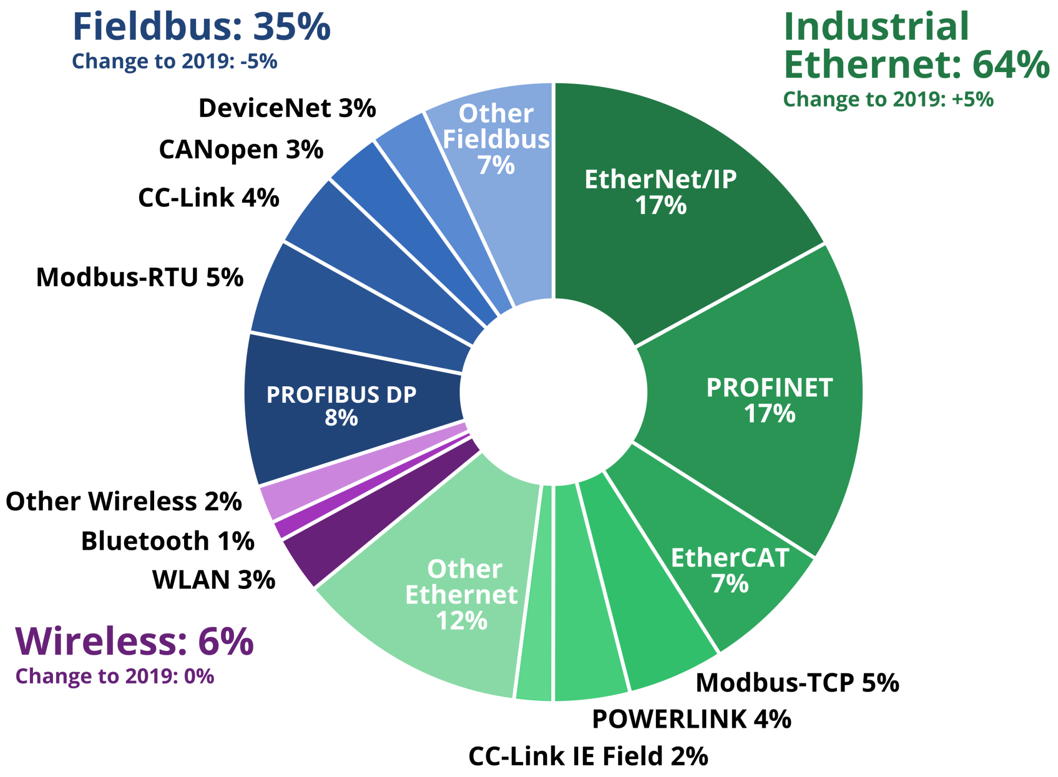


Figure 2.3: Market shares of communication protocols in IA in 2020 according to [58].

Ethernet equipment. The requirement for our solution to be as generally applicable as possible then effectively translates to being required to be based on Ethernet and be transparent to higher layer protocols that build on top of it.

2.2.2 Industrial Gateways

This part introduces the concept of industrial gateways and discusses, why gateways are the most suitable starting point to face the challenges we proposed above.

Gateways are independent and self-contained compute nodes. They are introduced into networks to serve specific use cases on the network level that cannot be implemented directly on the networked devices (end nodes) themselves. Gateways in general are used to serve as interfaces between two network zones or domains that differ in certain aspects. Another type of device used to introduce new functionalities to a network is the middlebox. Such devices provide for example firewalling or deep packet inspection. The approach presented in this thesis will make use of the concept of so-called encryption gateways. These devices effectively embody both characteristics as they on the one hand modify network traffic by encryption and on the other hand separate the network into secure and insecure domains. Hence, the terms gateway and middlebox will be used interchangeably.

In our approach, gateways are put in front of a specific industrial machine or a whole network segment. This effectively separates these entities from the rest of the network (Requirement 1). Fig. 2.4 shows a small example network. The gateways also have the ability to encrypt data traffic flowing from the devices. Other gateways can decrypt this traffic, creating secure network tunnels. As a result, data traffic is protected from potential malicious entities in other

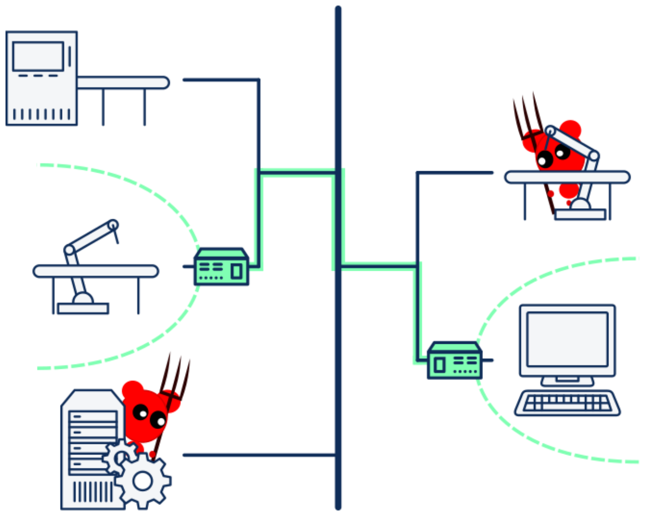


Figure 2.4: Gateways protecting end nodes from attackers inside an industrial network.

parts of the network. In effect, vulnerable legacy machines are only connected to other end nodes that are essential for their functioning and their data traffic is being protected while in transit. As gateways can be deployed anywhere within the network topology, it is possible to arrange the overlay tunnels very precisely, reducing the number of nodes that can connect to the vulnerable machine to a bare minimum.

In effect, this property fulfills our Requirement 4. As industrial machines and equipment cannot be upgraded directly with additional functionality, gateways make it possible to retrofit security functions by providing them without modifying the machines themselves. This is especially true for when the basic state of the machine is certified and any manipulation might void an operating permit. The gateways on the other hand can be easily maintained and updated on a regular basis.

In fact, gateways are the only concept that allows to add functionality to otherwise unchangeable nodes. This is why this otherwise elaborate and in practise rather costly concept was chosen as the basis for this thesis.

The next section will provide a look on the state of the art of gateways in industrial environments.

2.2.3 Related Work on Industrial Gateways

The research efforts on using gateways in industrial settings can be grouped in different broad use cases. Gateways are frequently used to establish connectivity between different types of networks. For example, they are used to bridge networks of different Internet of things (IoT) protocols [90, 72]. These approaches are not applicable to this thesis' research question, as they only focus on connecting end nodes that speak differing IoT protocols.

Another frequently implemented scenario is to use gateways to connect remote stations to a main network. This ranges from connecting very specific legacy energy grid installations [71], to bridging remote fieldbus networks [109], to SCADA-specific solutions [60], to more generic approaches to connect the factory network to cloud services [32]. As these types of solutions only target specific applications, they cannot answer the research question either.

A concept closer to our research question is that of so-called data diodes. There, gateways are employed as interfaces between two networks that have different security ratings. The gateways allow data packets carrying payload to only flow from the low security network to the high security network to make sure, that sensitive data does not leak [42, 57, 68]. Yet, this approach necessarily is very protocol-specific as each packet must be inspected and differentiated into containing management or payload data. It is not generally applicable as each protocol has to be handled individually. Finally, the use case is different. Data diodes are supposed to protect one high security zone from the rest of the network, while this thesis tries to create a system that can enforce many security zones independently of each other.

Yet, there are gateway approaches that specifically try to provide security tunnels for factory networks. Harada proposed a concept, where production cells are protected via a gateway that provides a firewall and acts as a virtual private network (VPN) end point [56]. This approach builds on the Scalance ecosystem, a range of industrial automation products from the vendor Siemens, and is therefore not generally applicable. Wright et al. used custom cryptographic protocols to secure SCADA data flows between Master Terminal Units and controllers using two gateways [118]. This approach again only works within special types of networks and excludes other ecosystems than SCADA.

Conklin used gateways to tunnel traffic between different devices [38]. Instead of encrypting the data flows, each gateway interprets the information within the data and only sends perceived state changes of the protected device to the other end. It assumes knowledge about traffic models and state machines of the devices. This is a highly specific solution only applicable to machines that run that type of software. Finally, the claim that security could be better served by masking traffic data in that way compared to encrypting it, is at least dubious.

Yun et al. also put gateways in front of production machines in order to protect them [122]. The problem description of their approach strongly resembles the approach of this thesis, yet their execution differs greatly. They make strange design decisions, that approach the realm of security voodoo. For example, they try to improve the security of the tunnels by constantly changing the encryption algorithm mid stream. While something like that may make sense to the uneducated, it nonetheless disqualifies this approach from further consideration.

Finally, Schleupner also proposed a gateway-based approach to secure data flows between industrial machines [108]. He uses pseudo random number generators to generate key streams that encrypt the traffic data and proposes a special source of randomness that improves the quality of the resulting cryptographic key stream. The random numbers are then transmitted to the stations using a separate fiber optic network the gateways also must be connected to. This approach is also marked by strange design choices. For example, the one-time encryption key used to encrypt a datagram is sent in a second datagram directly afterwards. This second packet is obfuscated with additional masking operations to achieve sender anonymity. This approach is deemed secure as the encryption and decryption functions are not publicly established. This directly opposes Kerckhoff's principle, a base rule of cryptography, that states that the secrecy of a cryptographic message must only be dependent on the secrecy of the key that was used to encrypt it. Furthermore, the separate optical network is deemed secure, because data is sent using quantum entanglement, where read operations on the line can be observed. Yet, if such a network is available, why not use it to send the encryption keys or even the payload in the first place. Additionally, it seems hardly economical to establish a second fiber based network within a factory just for that special purpose. It rather seems that this whole approach acts more as a use case to show the initially mentioned random number generation and distribution and has otherwise little practical applicability.

In summary, there exists to the knowledge of the author no general retrofitting approach to protect legacy industrial machines as stated as the challenge of this thesis.

2.2.4 Scope

To summarize the above, this thesis wants to propose a solution to integrate legacy industrial machines into future smart factory networks through the use of encryption gateways, while taking into account further requirements that resulted from discussed conflicts between the worlds of IA and IT.

On a conceptual level, gateways consist of a hardware platform and the software that runs on top of it. And although the author took part in research on how the hardware platform of a security-centric industrial gateway could look like [33], the platform's design is not part of this work.

Instead, this thesis is centered around providing a mainly software-based system that could run on any gateway platform and which provides the secure tunneling infrastructure that facilitates the transparent encryption of the end nodes' data to satisfy the stated Requirements 1, 5 and 6. Additional auxiliary works try to work towards Requirements 2 and 3. Requirement 4 is fulfilled by employing gateways in the first place.

In detail, the following topics are investigated:

- Selection of a suitable encryption protocol for tunneling (Chapter 3).
- Necessary modifications to enable and optimize the performance of the chosen protocol for the industrial setting (Chapter 4).
- Extension of the chosen protocol to increase general applicability (Chapter 5).
- Hardware extension to increase the availability of the connections protected by the encryption gateways (Chapter 6).
- Simple and intuitive concept for the configuration of the gateways (Chapter 7).
- Considerations around managing data flows in more complex tunnel topologies (Chapter 8).

3 Study on Contemporary Encryption Protocols

The fundamental building block of any network tunneling approach is its encryption protocol. Creating a custom protocol from scratch that satisfies all described requirements would be a very complex task that would provide its own challenges. Time and effort-wise, this would be a thesis in its own right and its focus would necessarily be a different one. It is rather more constructive to use an existing protocol and extend its functionality to serve our purpose.

Therefore, this chapter studies existing software-based encryption solutions. Their performance as well as further non-functional aspects revolving around security, manageability and ease of use are investigated. This survey looks beyond the mere applicability for our purposes and also evaluates the protocols on their general fitness to be implemented in the industrial environment.

Since this thesis is not based on any specific hardware platform or even a certain class of platforms, the performance is evaluated on multiple hardware platforms. These range from resource-restricted embedded platforms to powerful server platforms. As a result, we identify one protocol, that will form the basis for further considerations.

This work was previously published in [80].

The remainder is organized as follows: Section 3.1 reviews the related work in the field of surveying encryption protocols. Section 3.2 gives some technical background and introduces the investigated software-based encryption solutions. Section 3.3 introduces our experimental settings, the tools used for measurement, as well as the hardware platforms used for testing. Section 3.4 presents, discusses and compares results, while Section 3.5 concludes with lessons-learned and identifies further research questions.

3.1 Related Work

In the past, many studies were published, that investigated network encryption protocols. Yet, those studies generally are either old and outdated, small in scope, only tested few hardware setups, or were geared towards certain specific use cases. Even fewer specifically dealt with industrial or embedded environments.

For example, Czybik et al. did investigate security schemes for industrial communication, but were only concerned with integrity protection for very small embedded platforms for real-time applications [40].

Numerous studies that do performance comparisons only do so for few encryption protocols and then only test a small number of the optionally available cryptographic algorithms within the protocols [75, 39, 115, 123, 97]. Hardware platforms, if mentioned at all, consist of standard PC hardware and in some cases, the involved nodes are not even identical.

Another group of studies evaluates protocols not from a performance standpoint, but from a standpoint of management with a focus on operational questions [124, 89, 84, 27].

Khanvilkar et al. did an extensive study on open source protocols and investigated among performance also security properties and operational concerns [73]. Yet, this research was published in 2004 and is therefore quite outdated. Much of the discussed software is already deprecated and new approaches are not mentioned.

To the best of our knowledge, there is no large scale study comparing multiple recent network encryption protocols on different hardware platforms.

3.2 Investigated Protocols

Factory networks largely used to be flat layer 2 Ethernet-based networks. This means that all devices of the factory used to be directly connected within a single broadcast domain or local area network (LAN). Layer 2 stands for the data link layer according to the Open Systems Interconnection model (OSI model), standardized as ISO 7498. Yet, in the future these networks will be transformed into more complex, hierarchical and modular Internet Protocol (IP)-based layer 3 networks [125, 26]. In order to conquer this new complexity, separation and virtualization of new as well as legacy components, networks and whole infrastructures will be employed [86].

Especially in the case of legacy machinery, old devices might not be able to use the IP protocol. This makes it necessary for the encryption gateways of our approach to act as interfaces between the legacy layer 2 Ethernet of the machines and the layer 3 network of the envisioned smart factories. This technique is called bridging and means that incoming layer 2 Ethernet frames are packed into layer 3 IP packets. The data is sent to the other network and is then unpacked and reconstructed. The protocol had to support bridging in order to be selected for this study.

There are many different protocols available for that purpose. This study concentrates on those, where an implementation is freely available on Linux as open source. The access to source code is very important, as it makes modifications possible, in case the most fitting protocol still does not satisfy all requirements. Furthermore, Linux is the standard operating system for embedded platforms that are found in the industrial environment.

In the following, the investigated candidates are presented:

OpenVPN OpenVPN is a de facto standard virtual private network (VPN) protocol [12]. It runs as a user space application and is used in bridge mode (transporting layer 2 frames) for this study. OpenVPN supports many cryptographic algorithms for encryption and the generation of message authentication codes (MACs). All ciphers were tested, that were available for each hardware platform. OpenVPN allows to setup the secure tunnel using either the User Datagram Protocol (UDP) or the Transmission Control Protocol (TCP) as underlying transport protocol. We tested both options.

IPsec StrongSwan/IPsec is another de facto standard VPN protocol [15]. In contrast to OpenVPN, it is integrated into the Linux kernel. IPsec adds a header to IP packets and encrypts their payload. It does not offer bridging. Therefore, the Layer 2 Tunneling Protocol (L2TP) was used to be able to actually transmit layer 2 frames over the secure connection provided by IPsec [85]. This procedure is called L2TP/IPsec and is also standardized [101]. IPsec also supports multiple cryptographic algorithms for encryption and MAC generation and all available on each platform were tested.

Tinc Tinc is a protocol that allows to establish meshed VPNs between multiple participants. It is freely available for the Linux operating system [16]. It runs as a user space application and allows to bridge Ethernet segments. All available cryptographic algorithms for encryption and MAC generation were tested.

Freelan Freelan is a VPN protocol available for many operating systems [4]. It runs as a user space application and offers Ethernet bridging. It uses different cryptographic algorithms compared to the other protocols. All of those were tested.

SSH VPN Secure Shell is a cryptographic network protocol used for protecting network services, mainly known for providing remote command-line login and remote command execution [11]. Yet, it can also provide Ethernet bridging. It runs as a user space application and all available cryptographic schemes were tested.

MACsec MACsec is an encryption scheme, that was relatively recently introduced to the Linux kernel [45]. It works only on layer 2, meaning that it encrypts whole Ethernet frames. It is the first widely available free and open source implementation of a security scheme for layer 2 communication and its integration into Linux will make it become very widespread. MACsec can be assumed to be available on many hardware platforms and systems in the future.

Since MACsec does not offer bridging, we used L2TP to transmit the MACsec-encrypted frames over layer 3. While the payload is still encrypted and authenticated, we acknowledge, that this is not a protocol that can be implemented in practice as is, as it has some security-related drawbacks. Yet, we wanted to explore this scheme as it promised high performance because of its kernel integration and lean and new implementation.

While all previously mentioned protocols offer multiple ciphers for encryption and authentication, the only available cryptographic scheme within MACsec is 128 bit AES[1] in Galois Counter Mode, used for authenticated encryption.

Wireguard Wireguard is a very new VPN protocol, that was recently integrated into the Linux kernel and aims to replace IPsec. Main design goals are easier configurability and higher performance [43]. Just like MACsec, it only supports one algorithm, namely ChaCha20/Poly1305. This cipher provides efficient authenticated encryption and is standardized [98]. Like IPsec, Wireguard only works on IP packets. Therefore, an L2TP tunnel was set up on top of the Wireguard connection, so that actual layer 2 frames could be transmitted.

3.3 Methodology

The aim of this study is to compare the protocols identified above. We tested their performance, but also investigated non-functional properties and aspects. We wanted to first find the most suitable protocol for our use case but also to get a general understanding of the applicability of these protocols in the industrial environment, where practical considerations concerning usability and ease-of-use also play and important role.

The performance parameters measured were throughput and latency over the established secure tunnels. While the throughput measurements produced a single value, which made

[1]AES - Advanced Encryption Standard

Frame size (bytes)	64	128	256	512	1024	1400	1518
Weight	0.35	0.3	0.1	0.1	0.05	0.05	0.05

Table 3.1: Frame sizes and weights for calculating the weighted latency for the evaluation of VPN protocols.

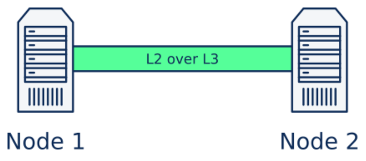


Figure 3.1: Basic experimental setting for performance evaluation of VPN protocols.

them easily comparable, the latency measurements were done for a variety of different frame sizes starting from the smallest possible up to the standard Ethernet maximum frame size. These individual values then resulted in a curve where the x-axis denoted the frame size and the y-axis the latency. The evaluation section of this chapter includes multiple such curves, e. g. Fig. 3.4b.

Then, in order to be able to rank different latency curves, weighted arithmetic means over the measured values on the curves were calculated. Since the focus of this study is the industrial environment, smaller frame sizes were weighted higher, as this reflects actual industrial traffic patterns more closely, compared to a mere average. Therefore, we favor protocols, that perform better with small payloads. The chosen frame sizes and their weights are shown in Table 3.1. We call this final value the *weighted latency* and we used it to compare the different experiments latency-wise.

We also studied non-functional aspects of each protocol, that were concerned with the security of certain parameter choices, how they have to be configured and managed, as well as the overall handling of each software solution.

We conducted experiments in two steps. First, we measured each protocol separately. Some allowed for choosing different algorithms for encryption and for authentication (called Message Authentication Codes (MACs) or digests), so we tested all available options. The individually best performing option was then chosen to compare protocols.

Testing all possible combinations of supported encryption schemes and MACs would have been prohibitively complex, so encryption algorithms were all tested with SHA-1[2] as MAC, while all MACs were tested with AES-128-CBC (meaning AES with 128 bit block size in the Cipher Block Chaining mode of operation). CBC was chosen, because is was available as an option on all protocols. Furthermore, baseline measurements were taken on each hardware platform without any cryptographic protection in place in order to find the upper boundaries for the maximum achievable performance.

As we wanted to reduce side effects as much as possible, we chose a simple and basic setting to conduct the measurement experiments. Two nodes were connected by wire, as depicted in Fig. 3.1. Layer 2 traffic was generated at one node and sent over a layer 3 connection to the other node. This setting was also used to evaluate the non-functional aspects of each protocol.

We ran tests on different hardware platforms using this setting in order to gain insights

[2]SHA-1 - Secure Hash Algorithm 1, using a 160-bit hash function

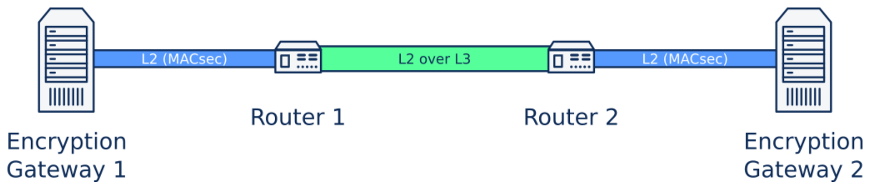


Figure 3.2: Extended experimental setting for performance evaluation of MACsec tunneling strategies.

on how the protocols behaved in different environments and how they would interact with various hardware specific properties. Platforms include very resource-restricted embedded platforms up to very powerful server machines. They are listed in Table 3.2. No special tweaking of hardware or software was done and default settings (on the operating systems as well as the protocols themselves) were kept as much as possible to reach the highest degrees of comparability between the platforms. The two respective nodes consisted of identical hardware and software (versions). The Freescale LayerScape platforms were provided by a project partner as part of the *fastVPN* research project that investigated the concept of industrial encryption gateways [49]. As these were prototypical platforms, not all protocols were available and could hence not be tested there.

Only after finding the individually best performing cipher settings for each protocol on each hardware platform could we compare them among each other. The discussion of the overall performance results independently of the underlying hardware platform as well as the non-functional aspects of the individual protocol can be found in Section 3.4.1. Section 3.4.2 reports on the concrete performance measurements and rankings of each protocol per hardware platform. Certain particularities, that are necessary to understand the results for a platform, are also examined. The overall discussion and comparative evaluation of the protocols is given in Section 3.4.3.

In a second step, we specifically investigated the potential of MACsec in our encryption gateway use case. We were not only motivated by the results (see below), but also by the fact that MACsec offers a characteristic, that other available protocols do not provide. MACsec is the first widely available open source software protocol to offers protection of entire layer 2 frames. As it encrypts the whole Ethernet frame, the upper layer protocols that the end nodes actually use for communication, are of no consequence and can be disregarded in our design.

MACsec is insofar transparent to the end nodes as long as they communicate using Ethernet technology. This corresponds nicely with our assumptions about the industrial environment in which our encryption gateway scheme resides and which was set as a requirement for this thesis (Requirement 6: Transparency).

To investigate MACsec further, we tested different possible strategies how to protect the communication flows between end nodes in separate networks beginning at layer 2. We used the experimental setup shown in Fig. 3.2. It consists of two (physical) routing devices which are connected by wire and have each an encryption gateway connected also by wire. An encryption gateway and a router constitute a LAN and both communicate on layer 2, while the two routers communicate over layer 3 with each other. HP MicroServers (see Table 3.2) were chosen as platform for the routers and the HP ProDesks were chosen as the gateways. These platforms were chosen so that it was ensured, that the gateways could produce a MACsec-protected data stream, that would fully saturate the available bandwidth of the router's network interfaces.

#	Platform	Processor	RAM	Network Interfaces	Crypto Hardware Acceleration	Operating System
1	HP ProDesk	Intel Core i5-4590 Quad-Core @ 3.3 GHz	16 GB	1x Gigabit Ethernet	AES	Linux 4.16.0/ Debian Buster
2	Raspberry Pi 3	ARM Cortex-A53 Quad-Core @ 1.2 GHz	1 GB	1x Fast Ethernet	-	Linux 4.14.32/ Raspbian Stretch
3	HP ProLiant MicroServer Gen7	AMD Athlon II Neo N36L Dual-Core @ 1.3 GHz	1 GB	3x Gigabit Ethernet	-	Linux 4.16.0/ Debian Buster
4	Xeon Server	Intel Xeon D-2146NT 16-Core @ 3 GHz	64 GB	4 x 10 Gigabit Ethernet	AES	Linux 4.19.4/Arch
5	Freescale LayerScape LS1020A	ARMv7 (v71) Dual-Core @ 1 GHz	1 GB	2x Gigabit Ethernet	-	Linux 4.9.98/ Custom

Table 3.2: Hardware platforms used for the evaluation of VPN protocols.

Furthermore, the HP MicroServers offered the necessary multiple Ethernet interfaces.

The protection of the data flows between the gateways can be organized differently, depending on the assessment of certain trade-offs. We chose three differing approaches. These and their reasoning are presented in the following:

MACsec over L2TP If the gateways already protect the communication payload using MACsec, the routers would only need to relay the already secured Ethernet frames. L2TP was again chosen for that purpose. As stated previously, this is no proper solution and in this scenario even opens up new vulnerabilities. For example, the MAC addresses of involved nodes leak and reveal meta data about the communication streams and denial-of-service (DoS) attacks become possible against the routers, as integrity of the data is only checked at the gateways. Additionally, MACsec would now have to be configured for nodes in different LANs, not just in the same LAN. As this is not a considered use case, tools for secure remote configuration of MACsec are not available. We did it by hand. Yet, this approach offers the least amount of additional computational overhead and we wanted to evaluate how a possible (properly designed) protocol could perform.

MACsec over Wireguard The security-related disadvantages of the previous prototypical approach can easily be ameliorated by setting up an additional secure layer 3 tunnel between the routers (a VPN). To set up VPN connections between different networks is standard procedure and this approach can be considered state-of-the-art. This approach would protect the meta data of the individual data flows between LANs and would provide integrity-checking at the routers. Wireguard was chosen for that purpose. Data is now encrypted twice, but this approach easily and with conventional means protects the whole data flow in and between both LANs. Additionally, not only MACsec but also Wireguard now have to be configured globally.

MACsec plus Wireguard Another solution is to protect each LAN with MACsec individually and then establish a VPN connection between the routers. The routers now first decrypt locally arriving MACsec frames and then re-encrypt them using Wireguard before relaying them to the other LAN. The configuration in this case is conventional, meaning intra-LAN configuration of MACsec is done locally and only the tunnel between the routers has to be configured globally.

The performance of these three approaches plus a baseline without encryption were tested and the results are presented and discussed in Section 3.4.4

Finally, we used ping to measure the latency, while we used iperf3 [7] for throughput measurements. Either dstat [3] or mpstat [9], depending on availability, were used for CPU utilization measurements. Dstat was available on each platform except for Freescale LayerScape and was also used to measure the raw Ethernet throughput directly from the Ethernet devices. All tools are freely available and can be considered standard.

Each experimental run consisting of a certain protocol with a certain set of encryption and/or MAC algorithm on a certain hardware platform was conducted by sending 10000 ping packets for each Ethernet frame size. The weighted latency was calculated from the resulting round-trip times. Iperf3 was run in TCP mode for 10 seconds. All measurement results shown in the following are to be understood to be the means of the results of all the individual runs.

3.4 Evaluation

This section presents and discusses our results. First, each of the investigated protocols, that were presented in Section 3.2, is discussed individually in Section 3.4.1. This includes how different cipher options performed as well as more general non-functional aspects. Then, Section 3.4.2 discusses how the protocols performed on each hardware platform. Performance rankings for each are given. Next, in Section 3.4.3 an overall summary and comparison of the results is given as well as lessons learned. Finally, Section 3.4.4 presents and discusses the results of our specific MACsec-related investigations.

3.4.1 Protocols

This section discusses the individual results for all protocols that were introduced in Section 3.2. For each, first, non-functional aspects are analyzed and secondly, when available, the performance of different cipher options is summarized and evaluated independently of the underlying hardware platform.

OpenVPN OpenVPN can be applied to a variety of use cases and is configurable via configuration files as well as parameters added to the start command. It is a very established and proven software and well documented, but still requires some knowledge to leverage it properly. It offers many different ciphers for encryption and MAC generation, among those, many old, outdated and broken (e. g. DES[3], Blowfish, RC2[4]). While these ciphers are clearly described as such, Blowfish, at the time of creating this study, was still set as default. This was changed in a later release. While for compatibility reasons it might make sense to support broken ciphers, it was a very bad design choice to still use it as a default.

The comparative performance of the different ciphers was similar with respect to latency and throughput. The UDP mode achieved a little bit more throughput compared to TCP mode. Differences in latency were not discernible.

The biggest impact to performance was contributed by the mode of operation, and not so much by the actual encryption algorithm. CFB1[5] and CFB8[6] showed (on all platforms) abysmal behavior and should not be used. Other modes had no discernible impact. The best performing encryption scheme was AES. After that came Camellia and SEED.

The most efficient MACs were MD5[7] and SHA-1. MD5 is also old and considered broken, and should therefore only be used against unintentional corruption. Since SHA-1 also shows signs of age (as collisions have been found, for details refer to [111]), it would be more wise to use SHA-2[8]. It only incurs a minor additional performance penalty. Certain MACs (Whirlpool, BLAKE2, MDC-2[9]) performed very bad on some or all of the platforms and performance of the MACs generally varied greatly. Latency was less impacted by the choice of MAC compared to throughput.

IPsec IPsec is also a proven and standard VPN protocol and as such well documented. Yet, is it more complicated to configure (via configuration files) compared to OpenVPN and needs more effort and expert knowledge. The strongSwan IPsec suite offers old and known to be broken ciphers for compatibility reasons, but describes them as such. The default encryption scheme is AES-128 and the default MAC is SHA-2. Both choices are up to date and can be considered secure.

IPsec runs in the Linux kernel and encrypts IP packets on the fly, if the destination address was previously configured as an end point. It does not create special virtual devices (compared to all the other protocols), which have to be used for routing. This makes it hard to detect, whether outgoing packets are actually being protected. If IPsec is wrongly configured or stops working for any reason, IP packets are still being sent and no loss of connectivity is observed by an upper layer. Instead, IPsec fails silently and packets are sent in plaintext.

The kernel integration makes IPsec very fast, but also more complex to use, as IPsec needs to be monitored constantly. In case of failure, in most cases the applications and services using this channel would probably not be informed about the lack of protection.

Furthermore, not all available cipher options actually worked (for comparison among protocols see Table 3.3). Encryption algorithms could be chosen with or without a mode of operation. Choosing an algorithm without mode always worked. AES with a selected mode worked sometimes on some platforms. Camellia with a chosen mode never worked (without

[3]DES - Data Encryption Standard
[4]RC2 - Rivest Cipher 2, after its author Ron Rivest
[5]CFB1 - cipher feedback mode 1-bit
[6]CFB8 - cipher feedback mode 8-bit
[7]MD5 - Message-Digest Algorithm 5
[8]SHA-2 - Secure Hash Algorithm with either 256 or 512 bit block size
[9]MDC-2 - Modification Detection Code 2

Platform	Raspberry Pi 3	HP ProLiant Micro-Server Gen7	HP ProDesk	Freescale Layer-Scape LS1020A	Xeon Server
OpenVPN	61/0/61	57/0/57	57/0/57	59/0/59	94/0/94
IPsec	23/32/55	21/33/54	31/23/54	—	28/27/55
Tinc	27/0/27	21/0/21	31/0/31	—	33/0/33
Freelan	6/0/6	6/0/6	6/0/6	6/0/6	6/0/6
SSH	16/13/29	19/10/29	16/10/26	29/0/29	15/10/25
MACsec	1/0/1	1/0/1	1/0/1	1/0/1	1/0/1
Wireguard	1/0/1	1/0/1	1/0/1	1/0/1	1/0/1

Table 3.3: Comparison of available cipher options for each VPN protocol and hardware platform. The triples specify the amounts of working cipher options, failed options and total of tested options.

one, it did).

Two MACs always worked (AES-XCBC[10], SHA-1). MD5 and AES-CMAC[11] worked sometimes on some platforms and all different variants of AES-GMAC[12] never worked.

Camellia generally showed best performance for throughput, yet worst for latency (while worst means 15% higher compared to respective best). AES performed slightly worse in throughput but showed less increase in latency. Best MACs were MD5 and SHA-1. SHA-2 was only marginally worse and, as previously explained, should therefore be preferred.

Tinc Tinc is a less common VPN protocol with a smaller user base. It is configurable via configuration files and is slightly more complex to use, compared to OpenVPN (as it offers more functionality). The documentation states that all ciphers from LibreSSL or OpenSSL in CBC mode are supported. And while this is true, when inquiring the options (via `openssl list -cipher-algorithms`), OpenSSL only lists all possibilities and does not offer an assessment about the security of these algorithms. While old and outdated ciphers are available, Tinc defaults to up-to-date AES-256 and SHA-2.

ChaCha20/Poly1305 showed best performance. On the platforms where it was not available, AES was best. After that came Camellia and SEED. If hardware acceleration was available, AES always performed best.

SHA-2 was on the same level as SHA-1 and MD5 and should therefore be preferred as MAC. Whirlpool showed the worst performance by a big margin and BLAKE2 was either among the best or among the worst, depending on whether there was a CPU bottleneck.

Tinc also showed some strange behavior. On the Xeon platform, the Shake-128 MAC showed the best throughput performance with a big margin and at the same time a very bad latency.

[10]XCBC - extended cipher block chaining
[11]CMAC - block cipher-based message authentication code
[12]GMAC - Galois message authentication code

ChaCha20 (without Poly1305) showed abysmal performance.

Freelan Freelan is a less common VPN protocol that offers much more functionality and serves different use cases compared to the standards OpenVPN and IPsec. It is therefore also more complex to configure, which is done also via configuration files.

It does not offer outdated or broken ciphers and defaults to the strongest available one. It uses elliptic-curve cryptography and users may choose different curves and whether AES should be run with a key size of 128 or 256 bit.

All options were tested and differences in performance were minimal. So the strongest option should be considered.

Yet, compared to the other discussed protocols, Freelan performed very poorly and should hence only be used in use cases, the other protocols cannot accommodate.

SSH VPN Secure Shell is a standard tool, yet probably not for the use case of network bridging. Therefore, configuration tends to be less intuitive and is done via parameters on the command line at startup.

It offers outdated and broken ciphers without warning or discussion. Yet, it defaults to ChaCha20/Poly1305 and other strong and up-to-date ciphers.

Not all offered cipher options actually worked (see Table 3.3). Independently of the platform, AES-CTR[13], AES-GCM[14] and ChaCha20/Poly1305 always worked and encryption in CBC mode always failed. Two MACs (UMAC[15], HMAC-SHA-2[16]) always worked, while HMAC-SHA-1 worked on most platforms. Other MACs only worked on a single platform or never.

Performance results were very volatile, probably owing to the fact, that Secure Shell operates in user space and VPN is not its intended use case. The most efficient encryption schemes were AES and ChaCha20/Poly1305. The results for the MACs showed no clear picture. They were pretty random and characteristics like tag size, ETM (Encrypt-Then-Mac) or not, UMAC or HMAC did not help to differentiate or group the results. Yet, MACs had major influence on the performance (on some platforms). Sometimes choosing ETM increased latency considerably.

MACsec MACsec runs inside the Linux kernel and can be statically configured via the iproute tool set. Since being new, more comfortable options are not yet available. In stark contrast to the previous protocols, the only cipher it offers, is AES-128-GCM. This cipher is an authenticated encryption algorithm and hence no extra MAC or digest needs to be specified. It is considered up-to-date and no misconfiguration in this respect can happen.

Wireguard Usability being a design goal of Wireguard, configuration was the easiest and least complex. It does much of the configuration automatically and does not need to be constantly monitored, in contrast to IPsec, as it provides its own virtual interfaces. Like MACsec, it does not offer different cipher choices. It only uses up-to-date ChaCha20/Poly1305.

[13]CTR - counter mode
[14]GCM - Galois counter mode
[15]UMAC - universal hashing-based message authentication code
[16]HMAC - hash-based message authentication code

3.4.2 Hardware Platforms

This section presents the performance results of each protocol on each hardware platform. For protocols where different cipher algorithms could be selected, the highest performing options were chosen to represent the protocols in performance rankings. The performance of the cipher options is further discussed for each hardware platform and, if necessary to understand the results, certain technical particularities of the hardware platforms are discussed as well.

HP ProDesk The performance results for this hardware platform can be found in Tab. 3.4. The protocols were ranked separately on their throughput and latency results. Where ciphers could be chosen, the choices are denoted in brackets behind the protocol name in the form of `<encryption algorithm>/<message authentication code>`. "Goodput" describes the amount of payload that the secure channel could transmit, while "throughput" denotes the total amount of data transmitted including overheads introduced by the protocols. Baseline measurements for both performance measures were taken without any security protocol present to estimate their upper bounds on the platform. The "% of Baseline" measure describes the relative distance of each protocol to that theoretical maximum. During throughput measurements, as much data as possible is generated on the nodes, which must then be processed (encrypted) by the nodes. This makes the CPU an important influence factor during these measurements. Hence, CPU usage was also measured. This does not hold for the latency measurements, which is why the CPU usage was ignored there. Identical tables are used below to present the results of the other hardware platforms.

On this platform, the CPU was powerful enough, so that nearly all protocols (except Freelan) approached line speed. The small differences in achieved goodput stem from the individually different protocol overheads. Also, the selection of ciphers (where applicable) did have less to no impact on the results. AES was generally the most efficient choice for cipher, due to the CPU providing AES hardware acceleration. On OpenVPN, some of the cipher options behaved very badly (apart from the previously discussed CFB1 and CFB8 modes of operation). MACsec again showed best latency performance. On this platform, no protocol outperformed the others. All, except for Freelan, can be recommended. If low latency is necessary, then MACsec or IPsec should be preferred.

Raspberry Pi 3 The performance results for this hardware platform are depicted in Tab. 3.5.

The results are generally dominated by the less powerful CPU of the Raspberry Pi 3 and the limited Ethernet device, which is connected over the USB interface. While the system achieves line speed when only sending and receiving data unencrypted, the speed is considerably reduced, when data is also protected. Network flows had high volatility, suffered frequent random outliers and showed erratic behavior. Furthermore, only Freelan used more than one CPU core. MACsec (uncharacteristically) performed worst for throughput, yet this may be explained by the way, the network interface is attached. Overall Wireguard showed the best performance, achieving the highest throughput and good latency.

HP ProLiant MicroServer Gen7 The performance results for this hardware platform are referenced in Tab. 3.6.

In the absence of hardware-based acceleration for AES within the CPU, ChaCha20/Poly1305 showed the best performance on multiple protocols. Wireguard, using the same cipher, showed a vastly better throughput performance compared to all other protocols. MACsec showed the

	Protocol	**Goodput in Mbit/s**	**% of Baseline**	**Throughput in Mbit/s**	**CPU Usage**
	Baseline	935.0	100%	979.6	2.5%
1	**Tinc** (AES-128-CBC/SHA-1)	902.0	96.5%	981.1	25.6%
2	**OpenVPN** (AES-128-GCM/SHA-1)	891.0	95.3%	981.4	22.1%
3	**Wireguard**	862.0	92.2%	982.0	21.9%
4	**SSH** (AES-128-GCM/HMAC-SHA-1)	861.0	92.1%	979.3	21.3%
5	**IPsec** (AES-256-GCM16/SHA-1)	846.0	90.5%	966.1	3.6%
6	**MACsec**	833.0	89.1%	968.3	3.6%
7	**Freelan** (AES-128-GCM/SHA-256)	395.0	42.2%	430.1	49.5%

(a) Throughput

	Protocol	**Weighted Latency**	**% of Baseline**
	Baseline	0.14	100%
1	**MACsec**	0.17	118.0%
2	**IPsec** (AES-256-GCM16/SHA-1)	0.17	120.2%
3	**OpenVPN** (AES-128-GCM/SHA-1)	0.23	157.1%
4	**Tinc** (AES-128-CBC/SHA-1)	0.25	170.5%
5	**Wireguard**	0.32	219.4%
6	**SSH** (AES-128-GCM/HMAC-SHA-1)	0.33	229.9%
7	**Freelan** (AES-128-GCM/SHA-256)	0.66	453.2%

(b) Latency

Table 3.4: Performance of VPN protocols on the HP ProDesk platform.

	Protocol	Goodput in Mbit/s	% of Baseline	Throughput in Mbit/s	CPU Usage
	Baseline	94.2	100%	98.2	5.0%
1	**Wireguard**	87.4	92.8%	98.4	21.4%
2	**Tinc** (ChaCha20/Poly1305)	72.3	76.8%	76.2	25.6%
3	**OpenVPN** (AES-128-CBC/SHA-1)	69.2	73.5%	76.1	26.5%
4	**Freelan** (AES-128-GCM/SHA-2)	68.2	72.4%	74.7	66.4%
5	**IPsec** (Camellia-128/SHA-1)	66.9	71.0%	74.5	18.5%
6	**SSH** (ChaCha20/Poly1305)	59.8	63.5%	73.9	26.2%
7	**MACsec**	42.8	45.4%	51.3	24.8%

(a) Throughput

	Protocol	Weighted Latency	% of Baseline
	Baseline	0.55	100%
1	**IPsec** (Camellia-128/SHA-1)	0.86	155.3%
2	**MACsec**	0.89	161.2%
3	**Tinc** (AES-128-Wrap/SHA-1)	0.91	165.7%
4	**Wireguard**	0.91	165.8%
5	**OpenVPN** (AES-128-OFB/SHA-1)	0.97	176.4%
6	**SSH** (ChaCha20/Poly1305)	1.28	231.6%
7	**Freelan** (AES-128-GCM/SHA-256)	1.92	348.4%

(b) Latency

Table 3.5: Performance of VPN protocols on the Raspberry Pi 3 platform.

	Protocol	Goodput in Mbit/s	% of Baseline	Throughput in Mbit/s	CPU Usage
	Baseline	941.0	100%	1010.6	14.1%
1	**Wireguard**	797.0	84.7%	944.4	88.0%
2	**IPsec** (Camellia-128/SHA-1)	287.0	30.5%	322.9	54.0%
3	**MACsec**	247.0	26.2%	294.8	57.3%
4	**Tinc** (ChaCha20/Poly1305)	197.0	20.9%	220.8	62.8%
5	**SSH** (ChaCha20/Poly1305)	178.0	18.9%	211.5	60.5%
6	**OpenVPN** (AES-128-CBC/SHA-1)	151.0	16.0%	174.6	52.1%
7	**Freelan** (AES-128-GCM/SHA-256)	98.6	10.5%	111.3	88.3%

(a) Throughput

	Protocol	Weighted Latency	% of Baseline
	Baseline	0.20	100%
1	**MACsec**	0.41	201.8%
2	**IPsec** (AES-128-CTR/SHA-1)	0.44	215.7%
3	**Tinc** (ChaCha20/Poly1305)	0.53	259.9%
4	**OpenVPN** (AES-128-GCM/SHA-1)	0.80	396.2%
5	**Wireguard**	0.81	399.3%
6	**SSH** (AES-256-GCM/HMAC-SHA-1)	0.95	470.5%
7	**Freelan** (AES-128-GCM/SHA-256)	1.32	653.1%

(b) Latency

Table 3.6: Performance of VPN protocols on the HP MicroServer platform.

lowest latency. Neither protocol can be clearly preferred. Instead this must be decided on the basis of the specific use case. If latency is more important, MACsec should be used. Yet, if high throughput is the dominating requirement, Wireguard should be preferred. Overall, in case no specific use case can be identified, Wireguard should be preferred, as it is the only protocol which achieves almost line speed while still showing in absolute terms acceptable levels of latency.

Xeon Server The performance results for this hardware platform can be found in Tab. 3.7.

The baseline performance measurement shows that this system can achieve the line speed of 10 Gbit/s. But, it uses frame sizes of up to 64k between the nodes and this is no standard behavior. This only works when traffic flows directly between the physical Ethernet network devices of these servers. It does not work over tunnels and also probably not over ordinary networking hardware (switches, routers), that would be expected in a real industrial setting. Therefore, already only using an L2TP tunnel (without additional encryption scheme) reduces throughput to 4.2 Gbit/s. Generally, there were big differences between different ciphers within the same protocols, yet the bottleneck on this platform clearly was not the CPU. Hence, the available AES hardware acceleration did not have an impact on the results. It must rather be attributed to something else, and we suspect the memory interface. Accordingly, the best performing protocols for both throughput and latency were MACsec and IPsec, probably because they run in kernel space. Wireguard, which runs also in the kernel (yet was still a prototype at the time of conducting the study) was close in performance, while the protocols running in user space performed very bad.

Freescale LayerScape LS1020A The performance results for this hardware platform are recorded in Tab. 3.8. This platform is an embedded platform and was available as a demo board. It did not run a full-fledged of-the-shelf operating system. Instead, it ran a special embedded Linux version with a reduced function set. Therefore, not all protocols could be tested and total throughput could not be measured as necessary system tools were not available. The goodput was measured with the usual ping and iperf3 tools.

CPU performance showed to be the main limiting factor on this platform as well and while line speed was achievable for unprotected traffic, encryption reduced the performance considerably. Again Wireguard showed the best throughput performance, while MACsec showed the lowest latency. Overall Wireguard should be preferred, as it achieves considerably more throughput than the other protocols, while recording second best latency.

3.4.3 Summary and Comparison

This section summarizes and compares the performance results as well as important non-functional aspects that distinguish the studied protocols.

Performance Fig. 3.3 summarizes the individual performance rankings for each platform. Since the platforms contained Ethernet devices of different speeds, the throughput measurements were normalized to the same order of magnitude (hence no units on the y-axis of that figure), in order to make their relative distance from the baseline (i. e. the theoretic upper bound) comparable. An analogous depiction of the latency results would not result in further insights, hence measurement results were not normalized there. Instead, absolute values were

	Protocol	**Goodput in Mbit/s**	**% of Baseline**	**Throughput in Mbit/s**	**CPU Usage**
	Baseline	9380.0	100%	9843.8	4.5%
1	**MACsec**	2880.0	30.7%	3366.5	8.1%
2	**IPsec** (AES-256-GMAC/SHA-1)	2830.0	30.2%	3134.4	8.2%
3	**Wireguard**	2030.0	21.6%	2361.1	24.5%
4	**SSH** (AES-256-GCM/HMAC-SHA-1)	823.0	8.8%	1037.3	8.8%
5	**Tinc** (AES-256-CBC/SHA-1)	675.0	7.2%	735.0	8.6%
6	**OpenVPN** (AES-256-GCM/SHA-1)	656.0	7.0%	725.7	7.7%
7	**Freelan** (AES-128-GCM/SHA-256)	159.0	1.7%	176.7	21.7%

(a) Throughput

	Protocol	**Weighted Latency**	**% of Baseline**
	Baseline	0.04	100%
1	**IPsec** (AES-256-GMAC/SHA-1)	0.06	161.5%
2	**MACsec**	0.06	161.7%
3	**OpenVPN** (AES-256-GCM/SHA-1)	0.12	300.0%
4	**Tinc** (AES-256-CBC/SHA-1)	0.13	343.4%
5	**SSH** (AES-256-GCM/HMAC-SHA-1)	0.22	550.0%
6	**Wireguard**	0.24	622.3%
7	**Freelan** (AES-128-GCM/SHA-256)	0.85	2200.0%

(b) Latency

Table 3.7: Performance of VPN protocols on the Xeon Server platform.

	Protocol	**Goodput in Mbit/s**	**% of Baseline**	**CPU Usage**
	Baseline	943.0	100%	74.0%
1	**Wireguard**	164.0	17.4%	81.6%
2	**MACsec**	51.0	5.4%	55.0%
3	**OpenVPN** (AES-128-CBC/SHA-1)	42.5	4.5%	46.9%
4	**SSH** (AES-192-CTR/HMAC-SHA-1)	42.5	4.5%	51.3%

(a) Throughput

	Protocol	**Weighted Latency**	**% of Baseline**
	Baseline	0.08	100%
1	**MACsec**	0.32	407.4%
2	**Wireguard**	0.36	459.6%
3	**OpenVPN** (AES-128-OFB/SHA-1)	0.40	514.2%
4	**SSH** (AES-192-CTR/HMAC-SHA-1)	0.69	894.5%

(b) Latency

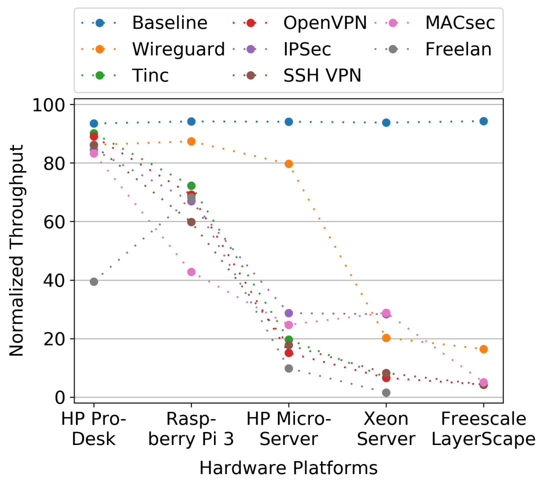
Table 3.8: Performance of VPN protocols on the Freescale LayerScape platform.

kept. The evaluation showed a clear trend towards the newer protocols MACsec and Wireguard. While MACsec (together with IPsec) was consistently best or second best performing protocol for latency, Wireguard showed the highest throughput achievable (or even line speed) on 4 of 5 platforms.

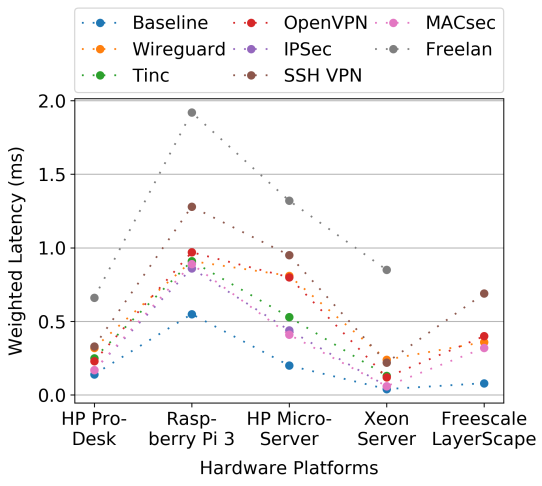
For 10 Gbit/s links, the equation seems to change considerably. In order to saturate these links, hardware support for encryption and powerful CPUs are not sufficient anymore and the bottleneck moved somewhere else. Where to, we can only speculate.

Non-functional Aspects The customary and established protocols (OpenVPN, IPsec, Tinc, Secure Shell) offer a multitude of ciphers to choose from. And, while variety is ostensibly a good feature, it has detrimental effects as well.

Some protocols offer ciphers in their documentation, but once configured just do not work (see Table 3.3) and furthermore, the sets of working algorithms change between platforms. We could not find conclusive evidence as to why this is the case. It is at least puzzling, as all platforms ran an up-to-date Linux kernel, with, in most cases, a current software distribution on top (see Table 3.2). Some ciphers even worked on none of the tested platforms. Within the ciphers that did work, some individual ciphers (e. g. Whirlpool, MCDC-2) always showed abysmal performance. The modes of operation CFB1 and CFB1 also performed very badly, independently of the configured cipher. Other ciphers showed very good and very poor performance depending on the platform (e. g. BLAKE2). Furthermore, some ciphers are so old,



(a) Throughput ranking.[9]



(b) Latency ranking.

Figure 3.3: Performance rankings of VPN protocols over all tested platforms.

that they have been broken by now, and must be considered insecure. Blowfish was proposed in 1993 and was even still the default setting for OpenVPN at the time of creating this study. Legacy support cannot be used as an argument here. Performance of the ciphers between platforms also differs widely. If performance actually is an issue, tweaking of the individual system becomes necessary and as we have showed, this is a non-trivial task.

This wealth of options, that probably accumulated over many years of development and maintenance of each software, seems to make it hard to manage it. In our minds, users would be better served, if the configurable cipher sets would be drastically reduced.

In contrast, the new approaches MACsec and Wireguard go in that direction and do not offer the user multiple ciphers, thereby eliminating the chance for misconfiguration. Additionally, this gives the software developers the chance to address performance and compatibility issues, that may arise on different hardware and operating system architectures. Therefore, we clearly recommend the use of those two protocols, wherever possible.

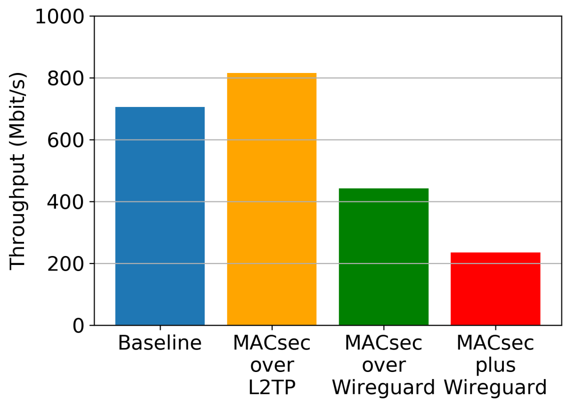
3.4.4 MACsec-based Gateway Setting

This section discusses the performance results for each of the routing strategies, we introduced above for our extended MACsec-based gateway setting, where four nodes are involved instead of two (see Fig. 3.2). The outer nodes constitute our MACsec-based encryption gateways, while the inner two are routers that only tunnel the traffic from the gateways. Fig. 3.4 shows the achieved throughput and latency performances for each of the approaches.

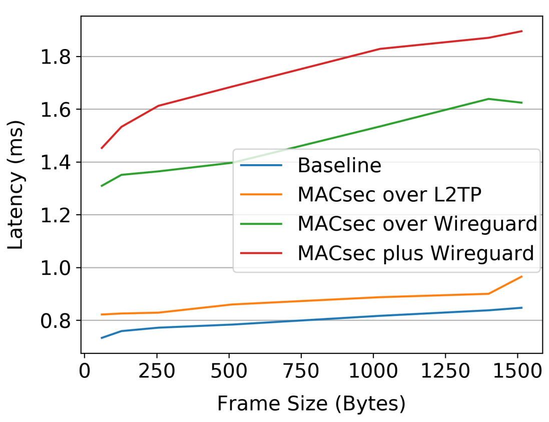
The throughput of the baseline measurement is lower than the measurement for the 'MAC-sec over L2TP' approach. This behavior is strange and counter-intuitive, but was reliably reproducible. It stems from the measurement tool we used.

Iperf3 is the only freely available standard tool for measuring network bandwidth. Yet, it is optimized for measurements across the Internet and is notorious for sometimes giving strange results especially in non-standard measurement scenarios (e. g. [67]). And indeed, our

[9]Normalization factors used for harmonization of each platforms results were 0.1, 1, 0.1, 0.01, 0.1



(a) Throughput comparison.



(b) Latency comparison.

Figure 3.4: Performance comparison of MACsec tunneling strategies.

extended setting constitutes an edge case, as we measure in a local network setting, while there is a source for delays that is untypical for such a local connection (the repackaging of the traffic done by the routers).

Iperf3 works by initially estimating the available bandwidth and then setting the sending rate accordingly for the remainder of the experimental run. At the start of the run, it sends as much packets as possible and looks at the return times of the acknowledgments. In our setting, the routers in the middle both had to process this initial batch of packets. This incurred delays, which iperf3 perceived as a bottleneck on the communication link. It factored this in and reduced its sending rate. The amount of reduction is based on a heuristic, which overestimated the delays incurred by the routers in our setting, resulting in the eventually lower measurement results. Again, this is due to the fact that iperf3 is optimized for measurements across the Internet and not for measurements in our local network setup.

The additional encryption step introduced in the 'MACsec over L2TP' approach, on the other hand, lead to the bottleneck shifting to the encryption gateway itself. Iperf3 could only send the initial batch of packets as fast as the CPU allowed. This lead to an overall straightened out data flow as the initial batch of packets could be better processed by the intermediate routers. The rate at which the acknowledgments arrived now matched the sending rate, leading to iperf3 not reducing the sending rate for the remainder of the measurement run.

The counter-intuitive results of the first two tested scenarios can hence be explained by iperf3's rate adjustment mechanism overestimating the incurred delay from the intermediate routers in the first scenario. Yet, this does not take away from the results for the latter three scenarios, where we measured our different tunneling strategies. There, MACsec was always enabled, meaning, the bottleneck was fixed and hence factored in the same way by iperf3. The result of the first scenario was merely meant to give an upper limit, to what bandwidth may be achievable. Yet, this was evidently not possible to show, due to limitations of the measurement tool. Instead, we will focus on the latter three results, as these are comparable.

The performance of the 'MACsec over L2TP' approach shows highest performance of the three, as the routers only relay already encrypted frames. The performance drops considerably with the remaining two approaches. The additional encryption steps performed on the routers, have big impact. The additional Wireguard tunnel within the second approach halves achieved throughput and almost doubles latency. The further step of the third approach of de- and encrypting the MACsec frames on the routers halves the achievable throughput yet again.

From these results, we can derive, that the conventional state-of-the-art approaches of adding a VPN tunnel ('MACsec over Wireguard' and 'MACsec plus Wireguard' in the figures) to protect data between LANs seems to be unfeasible for resource-restricted environments, where performance is nonetheless an issue. Therefore, the aforementioned trade-off between configuration complexity and performance should be answered individually depending on the use case.

These results motivated us to investigate the 'MACsec over L2TP' approach further and Chapter 5 will discuss our research in that direction.

3.5 Conclusion

This chapter investigated different network traffic encryption protocols and how to efficiently interconnect devices from different local area networks. Non-functional aspects as well as performance was analyzed, discussed and compared.

The classic and well established protocols, like OpenVPN and IPsec, were found to exhibit significant drawbacks in comparison to the newer approaches of MACsec and Wireguard. Generally and independently of a specific use case, these protocols should be preferred in the future.

Concerning the topic of this thesis, this study revealed valuable findings. MACsec turned out to be the most suitable protocol for our specific use case of encryption gateways as its pure layer 2-based approach fits best to our requirements. Its integration as a Linux kernel module gives it instantly widespread availability and for us the chance to modify it according to our needs. Furthermore, the encryption and authentication cipher ChaCha20/Poly1305 performed best in resource-restricted environments, where AES hardware acceleration within the CPU is typically not available. As this is precisely the type of environment our encryption gateways exist in, the next chapter will investigate (among other things) a possible integration of ChaCha20/Poly1305 with MACsec.

Furthermore, the MACsec-based experiments in the extended gateway setting showed potential for big performance increases compared to state-of-the-art approaches. Yet, the discussed drawbacks with the tested naive approach ('MACsec over L2TP') make further research in that direction necessary. Chapter 5 will discuss this question in detail and propose a possible solution.

4 Enabling and Optimizing MACsec for the Industrial Environment

The previous chapter surveyed different network encryption protocols and we identified the new encryption protocol MACsec as the best choice for our encryption gateway approach. Nonetheless, still certain challenges present themselves, when applying this general purpose tool to our use case set in the industrial environment. Legacy networking technology, which is typically rolled out in existing factories as well as performance requirements for latency-critical applications must be taken into account. Therefore, in this chapter, we introduce and discuss two modifications that enable and optimize the MACsec protocol for the industrial environment.

First, we investigate a modification that enables MACsec to be used within legacy networks and, therefore, make it possible for MACsec to run on industrial gateways in the first place and secondly, we improve MACsec's efficiency further, especially with industrial use cases in mind.

This work was previously published as an extended abstract in [78] and as a full article in [79].

The rest of the paper is structured as follows. Section 4.1 will provide technical background and motivate our proposed modifications in detail. Related work will be discussed in Section 4.2. Section 4.3 will present a modification to MACsec that enables it to be applied in scenarios, where the legacy networking technology Fast Ethernet is deployed. Section 4.4 investigates on how to improve the performance of MACsec especially in industrial use cases. Section 4.5 will end with concluding remarks and an outlook on further research questions.

4.1 Background

MACsec is a relatively new security protocol. It was standardized as IEEE 802.1AE in 2006 [22]. An implementation in the Linux kernel was introduced in 2016, making it available as open source for the first time [45]. Previously, MACsec was only implemented in a proprietary and closed source fashion by commercial companies. It offers encryption on OSI layer 2[1], meaning it protects whole Ethernet frames by encrypting the payload and adding additional headers (a so-called SecTAG and an integrity check value (ICV)). Fig. 4.1 depicts a MACsec frame together with a simple Ethernet frame for comparison.

Virtual private network (VPN) protocols, like IPsec [70] or the newer Wireguard [43], operate on OSI layer 3, meaning they transmit data using the Internet Protocol (IP). However, the assumption that IP is rolled out inside the network cannot be guaranteed to hold within the industrial environment, especially when considering certain industrial Ethernet solutions, that only work on layer 2, e. g. EtherCat [64]. Supporting these and other even older legacy installations makes the use of layer 2 encryption protocols a necessary tool to secure those

[1]According to the OSI layering model, standardized as ISO 7498.

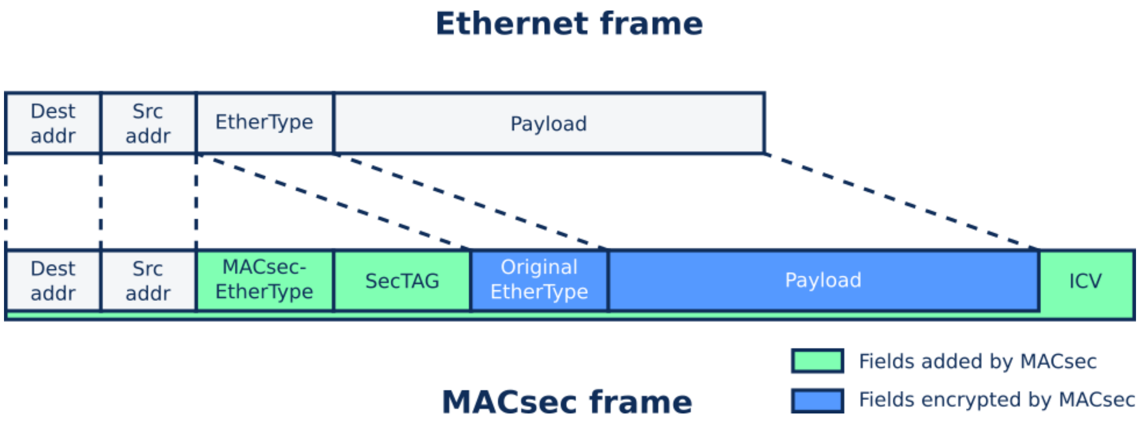


Figure 4.1: MACsec applied to an Ethernet frame.

networks [50]. Furthermore, the previous chapter has shown, that MACsec ranks among the highest performing encryption protocols, making it most suitable for performance as well as security-critical environments.

Additionally, Fast Ethernet is still the prevalent networking technology in industrial settings. It was introduced in 1995 and, because of the very long life-cycles of factory equipment, it will remain the backbone networking technology in many factories for many years. Furthermore, cabling is not easily replaced in factories and on industrial installations. It is often an integral part of a building, e. g. being cast in foam insulation as cable ducts need to be sealed for fire safety reasons. Also, replacing cabling that connects multiple buildings on a factory ground would typically be very expensive. Moreover, Fast Ethernet is the only allowed networking technology for certain legacy industrial communication protocols, like e. g. Profinet [64].

Fast Ethernet only allows for frames of a maximal frame size of 1518 bytes. After subtracting Ethernet headers, this results in a frame being able to transport at maximum 1500 bytes of payload in a single transaction. This value is called the maximum transmission unit (MTU). If MACsec is applied to an Ethernet frame that is already at maximum size, the addition of the necessary MACsec headers would make the resulting frame exceed this MTU. The frame would then simply be dropped by the involved networking devices (e. g. network cards or interfaces and switches) as it would not conform to the specifications of Fast Ethernet. Certain services and applications that require transmissions of large MTUs would fail silently, meaning without emitting error or failure codes and messages as they typically are not prepared to deal with transmission errors. These services and applications include e. g. software updates, supervisory video streams or simple SSL handshakes necessary for remote connections for maintenance purposes.

Jumbo frames, frames with a vastly increased MTU, were only introduced with Gigabit Ethernet and cannot be assumed to be available in these environments. The path MTU discovery (PMTUD) technique was invented as a tool to detect unknown MTUs on the transmission path of IPv4 connections. Yet, as it works on layer 3, it cannot be expected to be available either.

These considerations make it necessary to fragment frames, i. e. to split too large frames into smaller ones, which fit a particular MTU. However, according to the OSI layering model, fragmentation is a feature provided by layer 3. Yet, we cannot assume layer 3 networking to be present and hence we cannot assume fragmentation to be available. Therefore, in order to be able to use MACsec in industrial environments, it is necessary to implement a fragmentation scheme directly into MACsec. Section 4.3 presents and discusses our approach.

Furthermore, MACsec is specified to use a single cryptographic algorithm, AES[2] in Galois Counter Mode (AES-GCM). This is a so-called authenticated encryption with associated data (AEAD) algorithm, which means that it encrypts and authenticates data in one step. It is generally a good choice for generic information technology (IT) environments, as modern general-purpose CPUs provide AES hardware acceleration, which means that certain AES computation steps are directly implemented as hardware instructions on the CPU. They can therefore be computed very fast compared to implementations in software. Yet, this assumption does not hold for the majority of CPUs found in embedded systems for industrial environments. Results from the previous chapter showed, that on platforms which do not possess AES hardware acceleration features, other cryptographic algorithms can be faster.

As there is active research going on in the direction of high performance cryptographic algorithms, we investigated a set of promising ciphers that had the potential to improve the performance of MACsec. Results are presented and discussed in Section 4.4.

4.2 Related Work

Many protocols exist that can in principle encrypt traffic on layer 2. VPN software solutions like OpenVPN [12] or IPsec [15] offer so-called tunneling modes, where Ethernet frames are encrypted and transported over layer 3 networks. Yet, these are generally slower compared to MACsec and additionally have certain security-related drawbacks, as was shown in the previous chapter. As already mentioned in the previous section, layer 3 networking cannot be assumed to be available in industrial networks, anyway. The MACsec Linux kernel module is the first and freely available implementation of a genuine, well integrated and fast pure layer 2 encryption scheme, making it a de facto standard tool. Out of the available candidates evaluated in the previous chapter, it is best suited for our use case.

virtual local area network (VLAN) could satisfy some of our proposed requirements [20]. It offers separation of network flows by tagging frames with an identifier. As VLANs are standardized as IEEE 802.1Q, even unaware networking equipment would be able to process these frames. Yet, traffic would still be unencrypted and be sent in plain text.

IEEE 802.3br is an amendment to the Ethernet standard IEEE 802.3 and introduces a so-called MAC Merge sublayer which allows fragmentation of lower priority frames to support the timely delivery of high priority express traffic [21]. Fixed time slots are allotted for different priorities. Low priority frames get preempted, when their time slot ends. This means that those frames are fragmented in a way, that utilizes the remainder of their time slot. Once the high priority time slot is finished, the remainder of the low priority frame is sent. Fragments are buffered at the receiver side and eventually reassembled. The fragmentation mechanism operates on timing schedules and not MTU sizes. It also cannot be instrumented by higher layers (such as MACsec). Finally, it operates by modifying the preamble of the frame so that the receiving side can distinguish fragments from complete frames. This fragmentation scheme hence only operates on a link-by-link basis, as each intermediate networking device would buffer and reassemble the fragments. It does not offer end-to-end fragmentation, which is our intended use case. All these properties make this scheme unsuitable for our needs.

[2]AES - Advanced Encryption Standard

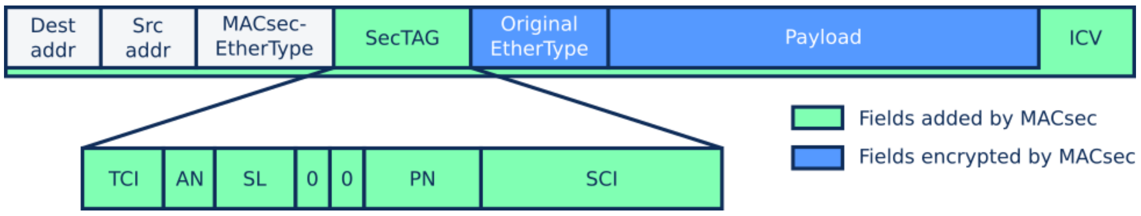


Figure 4.2: Frame structure of a MACsec frame. Irrelevant fields were omitted for clarity.

4.3 Fragmentation

This section presents our fragmentation scheme for MACsec. We first introduce the design of our approach and then present an implementation. We conducted a series of tests to evaluate the performance of our implementation. The testing methodology is presented, followed by a discussion of the results.

4.3.1 Design

The structure of a MACsec frame is depicted in Fig. 4.2. Simple Ethernet frames consist of fields for source and destination address, EtherType (indicates the protocol of the payload), and the payload itself. MACsec adds two fields as its own headers. One is the Security TAG (SecTAG), which contains fields and flags that are necessary for the handling of each frame. It consists of 14 bytes, of which two bits are not yet defined (marked as "0" in the figure) by the MACsec standard. In the following, the fields and flags will only be further discussed, if they are important to understand our fragmentation scheme. Detailed descriptions can be found in the MACsec standard [22]. The second field added by MACsec is an ICV that is formed over the whole frame and then appended to it.

Fig. 4.3 illustrates our fragmentation scheme. It starts by first checking, whether an incoming frame needs to be fragmented. This is done by comparing the payload size of the incoming frame to the MTU of the transport channel. If the frame exceeds the channel MTU, the payload is split accordingly to fit the smaller MTU. When splitting, the minimal frame size for Ethernet frames is respected. In theory, this might result in more than two fragments. Yet in practice, two fragments is the most likely result. New Ethernet frames are created for each payload fragment, copying the headers from the initial incoming frame. Then, each of these new frames is individually encrypted by MACsec.

In order for the receiver to be able to reassemble the fragments back into the original incoming frame, frames that carry fragments must be marked. For this purpose, we use one of the two as of yet undefined bits in the SecTAG, see Fig. 4.4 for reference. We call one of those bits

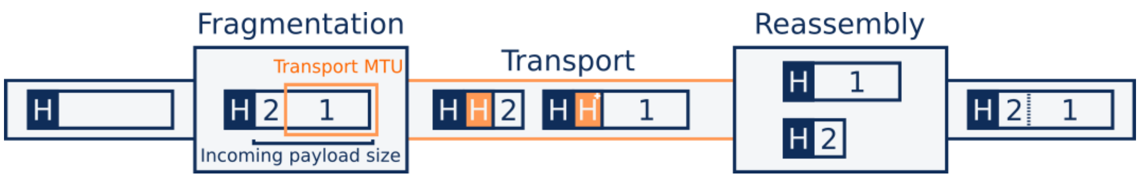


Figure 4.3: Fragmentation and reassembly of a MACsec frame exceeding the MTU of the transport channel.

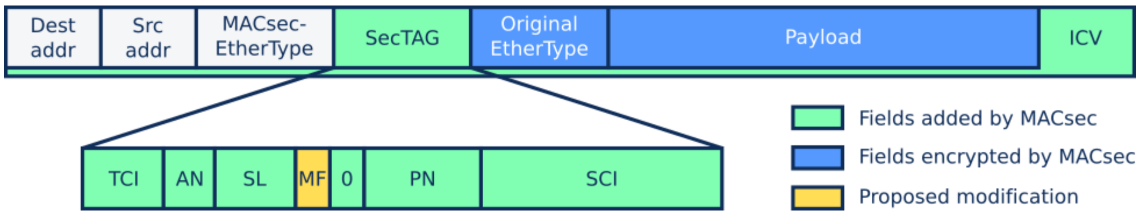


Figure 4.4: Frame structure of a modified MACsec frame, including the MF bit in the SecTAG.

the more fragments (MF) bit and it is set in the header of every fragment except for the last one. The receiver decrypts each incoming frame and if the MF bit is set, buffers the fragment and waits for more. When the last fragment is received (MF bit not set), it reassembles the fragments into a single frame and outputs it. The order of fragments is always kept via the packet number (PN) field in the headers.

This modification does not reduce the security of the MACsec protocol, if we assume standard MACsec to already work correctly. Our approach first fragments the incoming oversized frame and repackages the payload into new frames and only then inputs these new frames into MACsec's encryption function. Similarly, all frames are first decrypted in standard fashion and only then reassembled. Hence, our approach has no influence on the cryptographic properties of MACsec, it merely changes how inputs and outputs are processed.

Our approach only deviates in one way. During the creation of the fragment frames, one bit (the MF bit) within the MACsec header is modified. The whole header subsequently still gets integrity-protected by the ICV, meaning that also the MF bit is protected. Therefore, it cannot be spoofed by an attacker to make the recipient reassemble fragments in a wrong way. Although being visible to a potential attacker, as the bit is not encrypted, just integrity-protected, it does not reveal anything about the contents of the frame. And although it shows that the original payload was larger than the current MTU of the transport channel and that subsequent frames belong to one original payload, this could also just as easily be derived from the absolute sizes of each frame. A series of MTU-sized frames followed by a smaller frame under the presence of a fragmentation scheme clearly gives the fact away, that a bigger payload was fragmented. Yet, to obfuscate this, a different set of measures would have to be taken. This would include generating dummy traffic and using time sliced sending. This is completely out of scope for this work, as it would vastly increase the complexity of the protocol and reduce performance noticeably.

This approach, on the contrary, is very efficient in that it does not increase the overhead of the frame, as the undefined bits are already always transmitted within each frame. Also, increased processing times of each frame (checking MTU size at sending, checking MF bit at receiving) are negligible. Buffering should also not introduce significant overheads, as the buffered fragments are still subject to MACsec's replay protection mechanism. The replay window is set according to the capabilities of the device and if too many frames are received out of order, old ones are dropped.

4.3.2 Implementation and Test Methodology

We implemented our approach by modifying the existing MACsec Linux kernel module.

To understand the performance behavior of our fragmentation scheme, we focused on corner cases by testing multiple scenarios where frames of different sizes were transported, resulting

#	Scenario	Incoming payload size	Transport MTU
1	No Fragmentation	1468 byte	1468 byte
2	Small and Big Fragment	1500 byte	1468 byte
3	Two Big Fragments	2936 byte	1468 byte

Table 4.1: Configured MTU sizes for all scenarios to evaluate our fragmentation scheme. 1468 bytes are the effective MTU over Fast Ethernet, when MACsec overheads are factored in.

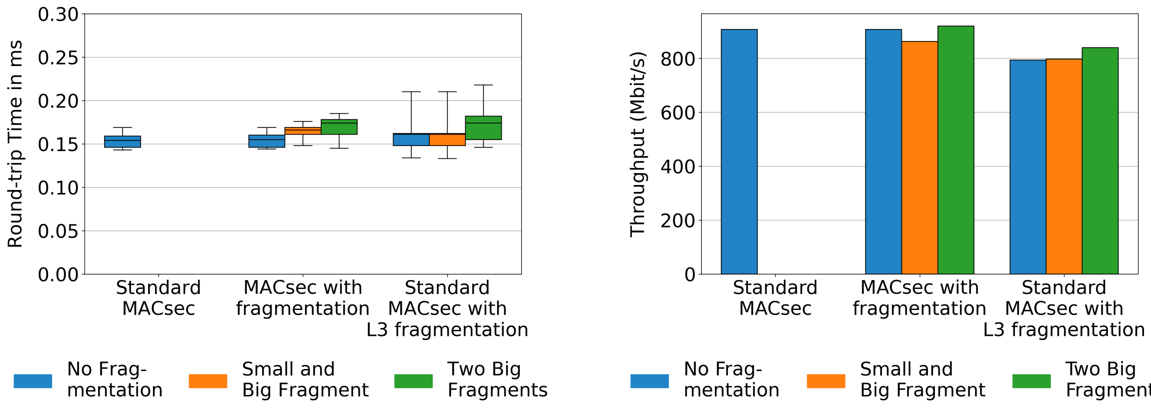
in different amounts and sizes of fragments each time.

To establish a baseline, in the first scenario, we chose the largest size possible of the incoming frame so that still no fragmentation would have to happen on the transport channel. In the second scenario, we slightly increased the size of the incoming frame, so that fragmentation would happen. This resulted in two fragments, one big and one small, that would be sent over the transport channel. In the last scenario, we chose a very large incoming frame that would be fragmented into two big fragments. The scenarios were implemented by injecting incoming frames with different payload sizes (denoted as "Incoming payload size" in Fig. 4.3). Tab. 4.1 shows the respective payload and transport channel MTU sizes for each scenario.

As there is no other layer 2 encryption scheme available that can also fragment Ethernet frames, we compared our approach to a standard MACsec tunnel with a layer 3 fragmentation protocol running on top. We used the Layer 2 Tunneling Protocol (L2TP) implementation from the iproute2 tool collection for that purpose [6]. Additionally, we measured the unmodified standard MACsec in scenario one to assess the efficiency of our implementation.

The test setting used to assess the performance of our implementation consisted of two nodes connected via Ethernet cable. This simple test setting was chosen, because we wanted to focus on the performance overhead our approach added to the transmission process compared to the standard solution. Therefore, we chose a setting with reduced complexity so that side effects from e. g. cross flows or intermediate networking equipment could be omitted as much as possible. The nodes' characteristics can be found in Tab. 4.3 under 1. We chose a platform with a powerful CPU and properly integrated Ethernet hardware in order to eliminate possible bottlenecks from these directions. This platform was equipped with Gigabit Ethernet hardware. Yet, this should not pose any problems for the validity of the experiments, as the for these experiments important characteristic, the MTU, is identical between Fast and Gigabit Ethernet (1518 bytes). Gigabit Ethernet is in principle capable of transmitting bigger frames, so-called jumbo frames. Yet, this has to be specifically configured and this was not done throughout the experiments.

The obvious choice to mimic legacy Fast Ethernet devices in a factory scenario would have been the Raspberry Pi 3 platform, which we also used (among others) in the evaluation in Section 4.4. However, it is equipped with a weak CPU and its networking interface is internally connected via USB. This slows the whole networking stack down by a big margin and also increases variance considerably. The two approaches we wanted to compare (MACsec with fragmentation and MACsec with an L2TP tunnel on top) use the networking stack to a different extent and together with a strained CPU, this would lead to measurement results that could not be attributed clearly to the difference of the approaches, but rather to their usage of this



(a) Latency. (b) Throughput.

Figure 4.5: Performance of our implementation of MACsec fragmentation compared to standard MACsec with and without using layer 3 fragmentation.

Ethernet-via-USB bottleneck configuration under a busy CPU. Therefore, we chose a platform without these drawbacks.

The performance measures tested for were throughput and latency. We used iperf3 [7] for throughput and ping round-trip times for latency measurements. While iperf3 automatically selected the biggest possible frame size, ping was configured for each scenario to produce frames with sizes according to Tab. 4.1. The iperf3 experiments were conducted in TCP mode. For each scenario, 1000 iperf3 runs (10 seconds each) were conducted, while 50,000 ping round-trip times were measured. All measurement results shown in the following are to be understood to be the means of the results of all the individual runs.

4.3.3 Evaluation

Fig. 4.5a shows the results for the round-trip time measurements of the experiments described above. Each bar shows minimal, mean and maximum values, as well as 25% and 75% quantiles.

Our approach shows expected behavior. In case of no fragmentation, it shows identical behavior compared to the standard implementation of MACsec. This shows, that our implementation is as efficient as the standard and does not create unnecessary overheads when no fragmentation is performed. The second scenario shows an increased mean round-trip time of 7% compared to the first scenario. The relatively big increase of our implementation stems from the fact, that fragmentation takes place and one frame is transformed into two. This means that an additional frame needs to be queued and sent, resulting in an increase of the total amount of bytes that need to be transmitted over the transport channel during one round-trip time measurement. The third scenario shows yet a further increase as even more payload is transmitted per measurement in relation to scenario two. Yet, since still only two frames are transmitted, the increase is only 3% from scenario two.

The standard approach with layer 3 fragmentation overall shows comparable behavior. Yet, while the mean round-trip times for all scenarios are roughly the same compared to our approach, the variance is much higher. This stems from the fact, that the layer 3 fragmentation scheme (L2TP) is its own protocol and works with its own queues and networking code. This results in extra steps, where frames are being stored and read from memory and where the operating system might preempt the whole transmission process. Additionally, L2TP adds another header. This results in an additional fragment in scenarios one and three, as the respective MTUs in those scenarios were chosen so that the channel MTU would be maxed.

Fig. 4.5b shows the results of the throughput measurements of the same experiments. The bars show mean values over all experimental runs for each scenario as described above. Extreme values and quantiles are not shown, as the variance within the results was comparatively low and not significant for the remainder of the considerations.

Again, our approach shows identical behavior compared to the standard MACsec implementation when no fragmentation takes place, as it does not add any overheads. The second scenario shows a decrease in performance of ~50 Mbit/s. Here, the second fragment is small but still has its own fully sized header. For this small frame, the header to payload ratio is very bad and effectively reduces the overall throughput by that margin. In the third scenario, two big frames are being transmitted over the transport channel. The performance increases by ~60 Mbit/s compared to the second scenario, achieving roughly the same performance as in the first scenario. The reason being the better ratio between header and payload, as was just described.

Compared to our approach, standard MACsec with layer 3 fragmentation always performs worse. As described above, L2TP requires its own header, which reduces the achievable throughput over the transport channel. This additional header already makes fragmentation happen in scenario one, explaining the small difference to scenario two. The throughput then increases slightly in scenario three although a third fragment is sent, because of the dynamics explained above.

In summary, the performance results show, that our proposed modifications are efficient and do not introduce additional amounts of overhead. Furthermore, compared to a standard solution, our approach can reduce the variance of latency and increase the throughput by some margin.

4.4 Performance Improvements

This section presents our findings in improving MACsec performance by replacing the default cipher AES-GCM with ciphers, that may be more suited to industrial use cases. In the following, first those ciphers are presented and discussed. Then, results of performance evaluations based on our implementation are reported.

4.4.1 Cipher Selection

There is much research going on in the direction of efficient high performance cryptography. The CAESAR competition (Competition for Authenticated Encryption: Security, Applicability, and Robustness) tries to steer this research [23]. It calls for AEAD algorithms applicable to one of three different use cases. The second use case "High Performance Applications" is concerned with efficiency for 32 and 64 bit CPU architectures and corresponds best to our use case. Industrial gateways fall into that category. From the three finalists of this second use case, we chose two. The third finalist was at the time of conducting this study still partly patented and hence not considered further. Patents typically hinder the dissemination of free and open source implementations and we did not want to invest in a solutions that might not be available to the general public in the future. Instead, we chose AEGIS and MORUS. A previous study showed AEGIS to be the most performant candidate among the finalists of the

#	Cipher	Key Length	Relevant Optimizations
1	AES-GCM (default)	128 bit	AES-NI
2	AEGIS	128 bit	AES-NI
3	MORUS	128 bit	-
4	ChaCha20/Poly1305	256 bit	-

Table 4.2: Ciphers investigated to optimize the performance of MACsec. AES-NI stands for Advanced Encryption Standard New Instructions.

CAESAR challenge, while MORUS proved to be the most efficient [77]. AEGIS uses the AES round function and is therefore compatible to AES hardware acceleration features [120]. It tries to be less computationally expensive compared to standard AES, while retaining a comparative security level. MORUS, on the other hand, only uses simple mathematical operations (AND and XOR) for encryption, which can be efficiently implemented independently of certain features of the underlying CPU architecture [119].

Yet, other studies showed partly different results. While AEGIS always showed high performance, the performance of MORUS was volatile and highly dependent on the implementation [69, 100, 95]. We selected both, as our use case was different from all these studies.

Additionally, we selected the ChaCha20/Poly1305 AEAD system, which is standardized in RFC 7539 [98]. It is not related to AES and designed to be fast and efficient independently of the hardware platform it runs on. We specifically chose it, because it has shown in the previous chapter to be considerably faster on platforms, where no AES hardware acceleration features were available. Tab. 4.2 lists the selected ciphers. We also measured the default MACsec cipher AES-GCM for comparison.

4.4.2 Methodology and Implementation

The test setting consisted of two nodes, which were connected via Ethernet cable. All ciphers were tested on four different hardware platforms. These are listed in Tab. 4.3. Additionally, an unencrypted plain Ethernet connection between the nodes was measured to gain an upper bound of what is theoretically possible on each platform (denoted as “Baseline”). The performance parameters tested for were latency and throughput. The latency was measured for different frame sizes, as it typically varies between small and big frames. The frame sizes were chosen within the boundaries of standard Ethernet, so that no fragmentation would take place.

All of the selected ciphers were available within the Linux Crypto API. MACsec had to be modified slightly to work with each different cipher as the length of the ICV field had to be adapted to the respective block size.

We used iperf3 for throughput and ping for latency measurements. Iperf3 was used in TCP mode. For each experiment, 100 iperf3 runs (10 second duration each) were conducted, while 50,000 ping round-trip times were measured for each frame size. All measurement results shown in the following are to be understood to be the means of the results of all the individual runs.

#	Platform	Processor	RAM	Network Interfaces	Operating System	Class
1	HP ProDesk	Intel Core i5-4590 Quad-Core @ 3.3 GHz	16 GB	Gigabit Ethernet	Linux 4.16.0 Debian Buster	Desktop/ Generic (AES-NI available)
2	HP ProLiant MicroServer Gen7	AMD Athlon II Neo N36L Dual-Core @ 1.3 GHz	1 GB	Gigabit Ethernet	Linux 4.16.0 Debian Buster	Embedded system
3	Raspberry Pi 3	ARM Cortex-A53 Quad-Core @ 1.4 GHz	1 GB	Gigabit Ethernet (via adapter)	Linux 4.14.32 Raspbian Stretch	Embedded system
4	Raspberry Pi 4	ARM Cortex-A72 Quad-Core @ 1.5 GHz	4 GB	Gigabit Ethernet	Linux 5.0.21 Raspbian Buster	Embedded system

Table 4.3: Hardware platforms used to evaluate ciphers to optimize the performance of MACsec.

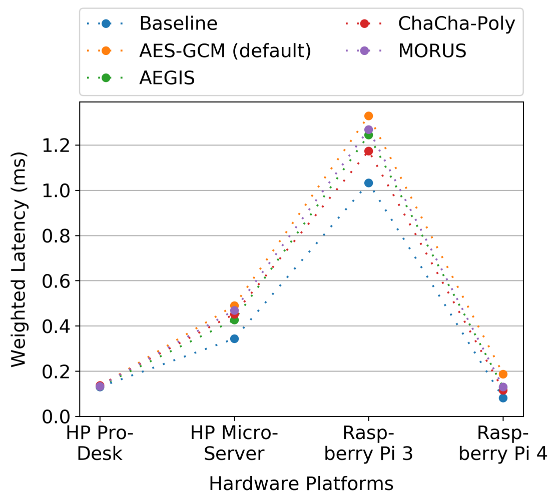
4.4.3 Evaluation

The ciphers were tested for latency and throughput. Fig. 4.6a shows a summary of the latency measurements for each cipher on all tested platforms. To create this compact view of the results, the measurement series for each cipher (on each platform) consisting of values for different frame sizes had to be condensed into one value (the *weighted latency*). This was done by calculating a weighted arithmetic mean over the values for each frame size. These values for each frame size were themselves formed by creating the mean over 50,000 individual ping measurements. We increased the weights of smaller frame sizes as this reflected industrial traffic patterns better than giving equal weight to each frame size. Tab. 4.4 lists the measured frame sizes and assigned weights. The detailed results for each platform including the measurements for each frame size can be found in Fig. 4.7.

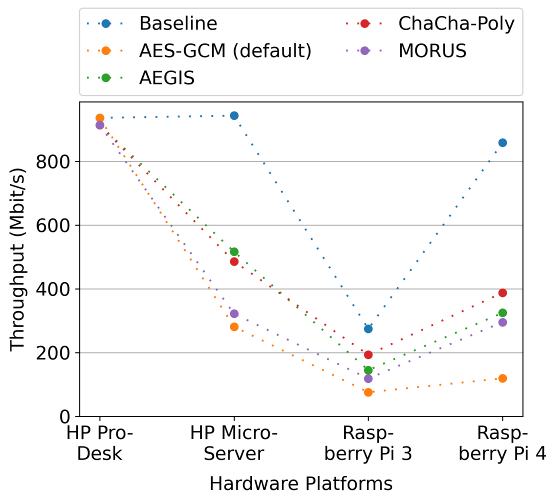
The results for latency show no clear picture. No cipher consistently outperforms the others. Yet, they generally manage to reduce the latency compared to the default cipher and bring it closer to the baseline.

Frame size (bytes)	128	256	512	1024	1400	1518
Weight	0.5	0.2	0.1	0.1	0.05	0.05

Table 4.4: Frame sizes and weights for calculating the weighted latency for the evaluation of ciphers to optimize MACsec.

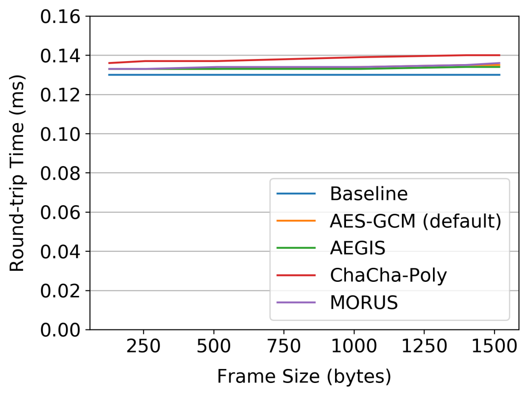


(a) Weighted latency measurements.

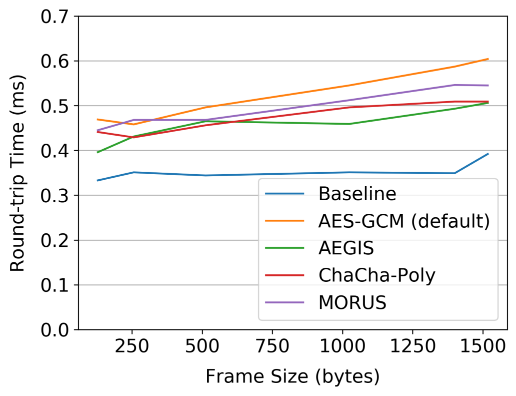


(b) Throughput measurements.

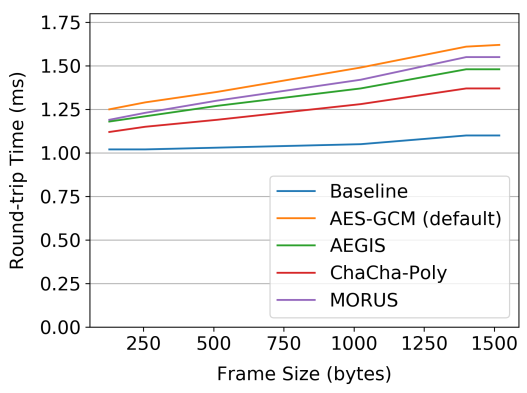
Figure 4.6: Comparison of ciphers to optimize the performance of MACsec over all platforms.



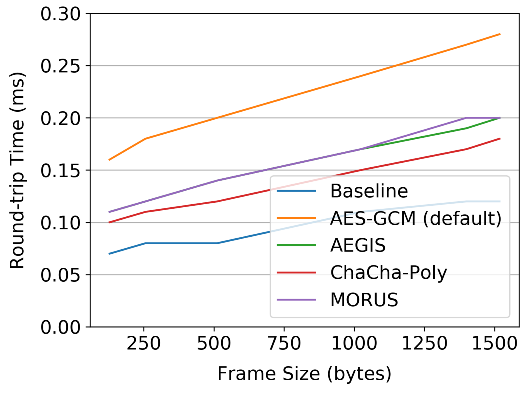
(a) HP ProDesk.



(b) HP ProLiant MicroServer Gen7.



(c) Raspberry Pi 3.



(d) Raspberry Pi 4.

Figure 4.7: Latency measurements of ciphers on all platforms to optimize the performance of MACsec.

The results on the first platform showed only minimal divergence. Fig. 4.7a gives more detail. The encryption step was clearly not the bottleneck, as the difference between the ciphers and the baseline is minimal. In this case, the CPU was powerful enough to always saturate the networking interface independently of the cipher. The small differences probably stem from each cipher's individual implementation on this platform. The AES-based ciphers profit from the available AES hardware acceleration.

The second platform showed more divergence in results. All ciphers performed slightly better compared to the default cipher. Yet, no cipher clearly outperformed the others, when looking at the results for each frame size (see Fig. 4.7b). Nonetheless, AEGIS showed the best overall performance. This is insofar surprising, as AEGIS' argument for efficiency rests on its more efficient use of AES hardware primitives compared to the standard AES-GCM. Yet, these hardware acceleration features were not available on this platform and from a theoretical standpoint, ChaCha20/Poly1305 or MORUS should have performed better. On the other hand, we cannot rule out implementation issues on this platform, as it is comparatively old and the cipher implementations may not be optimized for this CPU architecture.

The results of the third and the fourth platform are similar, as the latter is an update of the former with a faster CPU and a better networking interface. Both show consistent behavior over different frame sizes (see Fig. 4.7c and Fig. 4.7d), giving higher confidence to the results, compared to the previous platform. AEGIS and MORUS showed increased performance compared to the default cipher, while ChaCha20/Poly1305 achieved the best improvements. The fourth platform showed a bigger relative distance between the performances of the baseline and the default cipher compared to the other platforms, giving the most space for improving the performance with selecting a different cipher. In this case, ChaCha20/Poly1305 managed to decrease the round-trip time for all frame sizes by ~40% compared to the standard implementation.

Fig. 4.6b shows the combined results of the throughput measurements for each cipher on all platforms. No cipher performs consistently better than the others and the results generally mirror the latency results discussed above. The rankings between the ciphers stays the same. Only the relative gains, when compared to the default cipher, are bigger.

Again, the performance of the first platform cannot be improved by selecting a different ciphers. On the contrary, they reduce the throughput by 2.5%, probably due to deficiencies in our prototypical implementations.

The second platform shows vast divergence between the default cipher and the baseline, hinting that the CPU of this platform is clearly lacking necessary computing power to saturate its own networking interface when also encrypting the traffic. This opens up big potential for more efficient ciphers and consequently, AEGIS and ChaCha20/Poly1305 manage to almost double the achieved throughput (184% and 173% respectively), compared to the default.

The third and the fourth platform again showed similar behavior only differing in the relative distance of the results because of their different CPUs. All ciphers significantly improved the performance. ChaCha20/Poly1305 achieved a throughput of 255% and 326%, respectively, compared to the default.

Summarily, our experiments showed that resulting improvements depend on the underlying hardware platform. Significant throughput improvements can be achieved, while improvements for latency were, in absolute terms, small. Especially, the results from the second and the fourth platform showed, that when the computing power of the CPU in relation to the attached networking interface is low, much can be gained from choosing a more efficient cipher. AEGIS and ChaCha20/Poly1305 achieved the biggest performance increases, which is in concordance with results from the previous studies cited above. Furthermore, AEGIS even worked efficiently in a scenario, it was not primarily designed for. Hence, it may even be considered for platforms without AES hardware acceleration features.

4.5 Conclusion

The previous chapter identified MACsec as the most suitable candidate for our encryption gateway scenario. Yet, some modifications are necessary, to enable and improve it for our use case.

We identified the need for a fragmentation scheme within MACsec due to certain particularities of industrial networks stemming from the frequent use of legacy networking equipment. Our proposed scheme works efficiently and does not incur overheads when fragmenting and even slightly improves performance compared to a standard solution.

Additionally, this work identified two highly efficient cipher algorithms that can significantly improve the performance of MACsec on embedded platforms, which are the type of platform industrial encryption gateways are typically built upon. Especially ChaCha20/Poly1305 showed vast potential on platforms without AES hardware acceleration features. As it is also by default included in the Linux kernel, in general, this cipher should be considered more in comparable use cases in the future and is selected for our approach.

Yet, some open questions remain. For example, further research could investigate related use cases, where the proposed fragmentation scheme might yield additional benefits. And while the two ciphers identified, AEGIS and ChaCha20/Poly1305, turned out to considerably improve the performance of MACsec, it could be beneficial to investigate further ciphers for even higher improvements.

5 Tunneling MACsec Frames over Layer 3

Chapter 3 identified MACsec as the best starting point for investigations in the direction of a tunneling protocol that may satisfy all the requirements for our encryption gateway scenario we discussed in detail in Chapter 2. Chapter 4 investigated the protocol further and proposed some necessary modifications to standard MACsec that enabled and optimized it for our use case in the industrial environment.

Yet, one drawback is still remaining and must be addressed, so that MACsec can actually be used for our scenario. This drawback is, that MACsec works only on layer 2. This means that it can only be used inside the same local area network (LAN). We hinted at this problem in Chapter 3 and showed that while a naive solution was insecure, secure state-of-the-art approaches were very inefficient. Hence, if MACsec was used as is as the security protocol on our encryption gateways, they could not protect traffic flowing beyond LAN borders.

As this would reduce the general applicability of our approach immensely, we investigate in this chapter how to extend MACsec with the ability to tunnel or bridge its traffic between multiple LANs in a secure and efficient manner.

This work is, at the time of writing this thesis, being published in [82].

The remainder of this chapter is structured as follows. Section 5.1 first outlines our general approach and then gives details necessary to understand our two proposals to enable tunneling for MACsec. This includes the header structure of MACsec frames, together with an analysis of how sensitive the individual fields in the headers are towards attacks. Secondly, as we are just as interested in efficiency as in the previous chapters, we will discuss the topic of fast packet processing to optimize the performance of our approaches as much as possible. Section 5.2 will discuss related work. Section 5.3 presents our two approaches and Section 5.4 details our implementation. Results are discussed in Section 5.5. Section 5.6 gives concluding remarks.

5.1 Background

The state-of-the-art approach of tunneling over insecure networks is to use a virtual private network (VPN) protocol. Yet, we have shown in a previous chapter (in Section 3.4.4), that by simply adding a VPN protocol, the overall tunneling performance between two encryption gateways is significantly reduced. The reduction stems from the VPN protocol re-encrypting the whole MACsec frame. This is actually not necessary, as the payload is already encrypted by MACsec. This results in overheads that do not deliver additional payoffs.

Following from that, the straightforward efficient but also naive approach would be to use a non-encrypting tunneling protocol like Layer 2 Tunneling Protocol (L2TP) [85], Virtual Extensible LAN (VXLAN) [87] or generic routing encapsulation (GRE) [55] and only rely on the security properties of MACsec. This, however, has two main drawbacks. First, a lack of confidentiality of the frame headers, which may contain sensitive information and, second, a missing authentication at the tunnel end points, which allows for arbitrary traffic injection and subsequent denial-of-service (DoS) attack opportunities against the internal LANs.

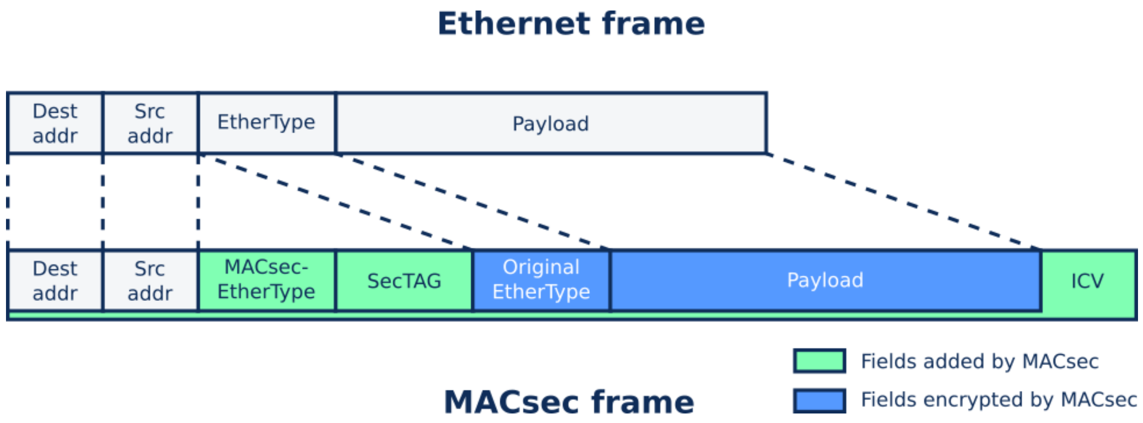


Figure 5.1: MACsec applied to an Ethernet frame.

Instead, we want to extend this naive idea by focusing on the unprotected headers and provide the missing security properties, while preserving its efficiency. In this work, we will propose two different strategies on how to protect these headers so that a MACsec frame can be securely tunneled without touching the already encrypted payload within that frame. The rationale hereby is, that by only working on the comparatively small headers, big increases in efficiency can be gained, compared to the state-of-the-art, that just encrypts the whole frame as is.

The two approaches investigated here are equally secure and differ in terms of complexity. Both will heavily focus on efficiency, as performance will be a key factor in future industrial networks, while at the same time industrial embedded platforms will still be rather resource-restricted entities. Therefore, in the following, first MACsec headers will be discussed in detail as well as their sensitivity for potential attacks. Then various approaches towards efficient network processing will be discussed.

5.1.1 MACsec Header

MACsec protects layer 2 Ethernet frames by encrypting the frames' payload and by integrity protecting the whole frame, as shown in Fig. 5.1. It adds at maximum 32 bytes of additional headers that get integrity protected as well. Destination and source addresses of the original frames are kept. The original EtherType is moved to the payload (data segment) and the MACsec exclusive EtherType is set instead. This encapsulation of whole Ethernet frames happens independently of any higher layer protocol that frame might be transporting. This makes MACsec an ideal protocol for the encryption gateways of our approach as it can transparently protect the data traffic of any Ethernet-based legacy industrial machines.

As the figure shows, the payload of the MACsec frame is already encrypted. Yet, the MACsec header fields remain readable. To be able to tunnel these frames in a secure way, we will now investigate in detail, which of the header fields is security sensitive and needs protection.

MACsec defines communication relationships as secure channels (SCs). These are configured unidirectional point-to-multipoint, meaning that a sender can send data to multiple receivers. A corresponding SC has to be configured on all receivers and when two MACsec entities want to communicate to each other, two SCs must be configured where both are respectively sender and receiver. A SC is identified by the 64 bit secure channel identifier (SCI). It consists of the 48 bit MAC address of the MACsec device plus a 16 bit port number. The configuration of ports allows to have multiple MACsec instances on the same device. The SCI can also be omitted if sender and receiver are directly connected. This reduces the overall MACsec header size, but it also is a special case that requires direct cabling between MACsec devices. This case is ignored here.

The 32 bit packet number (PN) is used to establish an order of sent frames and also acts as the counter for the AES-GCM cipher that is used for authenticated encryption. Galois counter mode (GCM) dictates that for security reasons a key may never be used with the same counter twice. Therefore, when the PN overflows, the old encryption key is discarded and a new key is selected. These keys are hence ephemeral.

MACsec allows for a certain amount of asynchronism within a defined replay window. To guarantee this window even when the PN overflows, the concept of security associations (SAs) is introduced. The actual encryption keys are not bound to an SC but to an SA. An SA is defined by the SCI plus a 2 bit association number (AN). When the PN reaches its maximum value, the next AN is chosen, resulting in a reset of the PN to 0 and a different encryption key being selected. To allow for smooth rollover two ANs can be active at the same time and a receiver can choose which key to use for decryption based on the AN number found in the header. Keys can either be configured by hand for a specific SA or be provided by the MACsec key agreement (MKA) protocol that manages MACsec stations automatically [25].

The 6 bit short length (SL) field is set to zero if the payload is larger than 48 bytes and represents the length of the payload otherwise.

The 6 bit tag control information (TCI) field sums up different single bit flags. The Version (V) bit is always (as per the standard) set to 0, while the End Station (ES) bit indicates that the sending MACsec station is also the source of the frame. In effect a set ES flag means that the frames' Ethernet source address is identical to the first 48 bits of the SCI. The SCI present (SC) flag shows presence of the SCI. As described above, there might be cases, when the SCI can be omitted. The Single Copy Broadcast (SCB) flag is used in fiber tree topologies to toggle single copy broadcasts without the explicit use of the SCI. This scenario is also not relevant to this work. The Encrypted payload (E) flag indicates that the payload is additionally encrypted instead of only being integrity-protected. In this work, we always assume that this is the case. The Changed text (C) flag indicates whether the data segment is simple payload or management information addressed to the MKA client running on the station. Two bits (0) exist in the header that are as of yet undefined in the MACsec standard and therefore are always set to 0. They are reserved for future use. A 16 bit integrity check value (ICV) is calculated over the whole frame and is appended at the end. This value provides integrity protection for all headers and the payload of the frame, irrespective if they originate from the original frame or additions by MACsec. This value is always calculated and, in contrast to the payload encryption, not optional.

After this short introduction of the MACsec header fields, we will now look at them individually on how sensitive towards security they are. Specifically, what information could an attacker derive when observing the headers when they are tunneled in an unprotected fashion, as described with our initial naive tunneling approach.

First, an attacker would see the Ethernet source and destination addresses of the tunneled frame. These are the addresses of the original machines that needed protection in the first place. This would allow the attacker to attribute communication partners and relationships as well as data flows between them. He would also learn about the traffic patterns of the communication, including volume and frequency. Additionally, Ethernet addresses (MAC addresses) typically include information about the vendor of the network card. This might give clues to identify the specific industrial machine or type of machine that is communicating.

An attacker would also learn the EtherType of the tunneled frame (MACsec) and hence would learn how to interpret the following byte fields (that may otherwise look random).

The SCI allows to identify the MACsec devices (the encryption gateways). This would give insights on the structure of the internal network by showing which industrial machines are grouped behind the same gateway. Recording observed SAs and PNs would reveal traffic patterns of the communication. These values on themselves would also make it possible to discern individual flows.

In contrast, the SL field does not offer information to the attacker that he cannot learn anyhow by just observing the length of the frames. The TCI field also does not leak any new information, as the attacker can infer flags anyway based on what is observable. The only interesting flag is the C flag, that determines whether ordinary payload or MKA traffic is transmitted. As MKA traffic will be handled differently in our design, it can be ignored here. The payload, including the original EtherType, is encrypted anyhow, so the attacker cannot learn anything, except the length. The ICV is a cryptographically strong message authentication code (MAC), that is for the attacker not discernible from random bits. He especially is not able to use this value to create bogus frames that would pass the integrity check or derive information about the encryption key from it. Fig. 5.3a summarizes the above and depicts which header fields of a MACsec frame we consider sensitive.

5.1.2 Fast Packet Processing

MACsec-tunneling will be facilitated by a single device that will sit at the edge of the local network. Hence, it must handle the traffic flows of all encryption gateways in that domain in parallel. And as network performance demands in general will only grow in future industrial networks, we also aim to be as efficient as possible. Therefore, we will now focus on the topic of fast packet processing. This includes the processing steps facilitated by the network stack and excludes the concrete encryption step of the payload (as we investigated that step at length in the previous chapter).

In contemporary general purpose systems, the performance is limited by the architecture of the network software stack and not by the data rates of the physical network interface [36]. Additionally, most transmission overheads typically stem from per-packet processing steps. This means that the size of a packet or frame has only a secondary influence on the processing times. It additionally means, that small packets, that are prevalent in industrial scenarios, can also profit greatly from optimizations from that direction.

The networking performance in general can be enhanced by various software- and hardware-based approaches. Hardware-based approaches include network cards that are equipped with special field-programmable gate arrays (FPGAs), so-called smart network interface controllers (SmartNICs). These are special integrated circuits that can be directly configured or programmed by the user. Single instructions or whole applications can be offloaded to them, meaning that they run directly on-top of the hardware without software layers in between. This increases their performance considerably, compared to a normal implementation that runs on-top of an operating system. Yet, these approaches depend on specific hardware extensions as well as interfacing software that integrates these features with the general purpose operating system that is still needed. Additionally, these techniques are, as of yet, only found in cloud computing environments and not in the sphere of industrial computing.

Therefore, we focussed on a software-based approach instead. There are different software frameworks available that implement various techniques that optimize networking performance, like for example reducing kernel interrupts and system calls, zero-copy and memory mapping or batch processing and parallelism.

The most prominent frameworks for this purpose are netmap, PF_RING_ZC and Data Plane Development Kit (DPDK). According to [51], DPDK shows the best performance of the three, while PF_Ring is a close second. It additionally provides better access to hardware features and offloading. Therefore, we based our design on DPDK.

DPDK provides an extensive software library for high-speed packet processing and tightly integrates with other functions that are useful in this context, like hash table management [1]. And while DPDK is mainly supported by and traditionally focussed on network interface controllers (NICs) from Intel, support for network controllers from other vendors is steadily increasing [2]. After loading a DPDK driver into the system and binding it to a specific NIC, the network controller is then invisible to the kernel and hence outside the standard Linux kernel stack. This makes it easily possible to dedicate the physical device to one use case and not suffer from overheads introduced by secondary services provided by the general purpose operating system. DPDK relies on huge pages. These make it possible to map 1 GB instead of the standard 4 KB of memory. Memory is also continuously managed, and hence never swapped, resulting in fewer memory operations. This then greatly increases the networking performance, because memory operations are typically the most costly operations in this scenario.

5.2 Related Work

The use case of bridging or tunneling networks, so that participants can connect to each other as if they were in the same network, is well researched. The Layer 2 Tunneling Protocol is a standard tool for that purpose [85]. It allows to transmit layer 2 Ethernet frames over layer 3 Internet Protocol (IP)-based networks. It encapsulates Ethernet frames into User Datagram Protocol (UDP) packets, but does not protect its payload in any way. No encryption or integrity protection takes place. Therefore, L2TP should only be used in public networks (the Internet) together with the IPsec protocol. IPsec adds the necessary data encryption and integrity protection. This approach is canonically called L2TP/IPsec [101]. Other standard VPN protocols, that could be used likewise, are OpenVPN [12] and Wireguard [43].

In contrast to our intended use case, these approaches only offer point-to-point connectivity. Multiple end points are not supported. VXLAN on the other hand offers such point-to-multipoint connectivity [87]. It was specifically designed for this purpose, but also does not offer any protection of the tunneled traffic data.

The bridging of industrial networks will only become a more important subject in the future, for the reasons given above. Many concrete scientific proposals have been published. Yet, even newer approaches lack security considerations (like [72]) and hence cannot be applied to our use case, where we assume zero trust networks. Other works consider security but employ new technologies and disregard the need for backwards compatibility for legacy components [90].

In contrast, we already enhanced and optimized MACsec for the industrial environment. What it is missing, is specifically the ability to bridge networks. Just adding a state-of-the-art VPN solution for that purpose, would incur unnecessary overheads, as already mentioned. Since no previous approach addresses all of our requirements, this work wants to add the

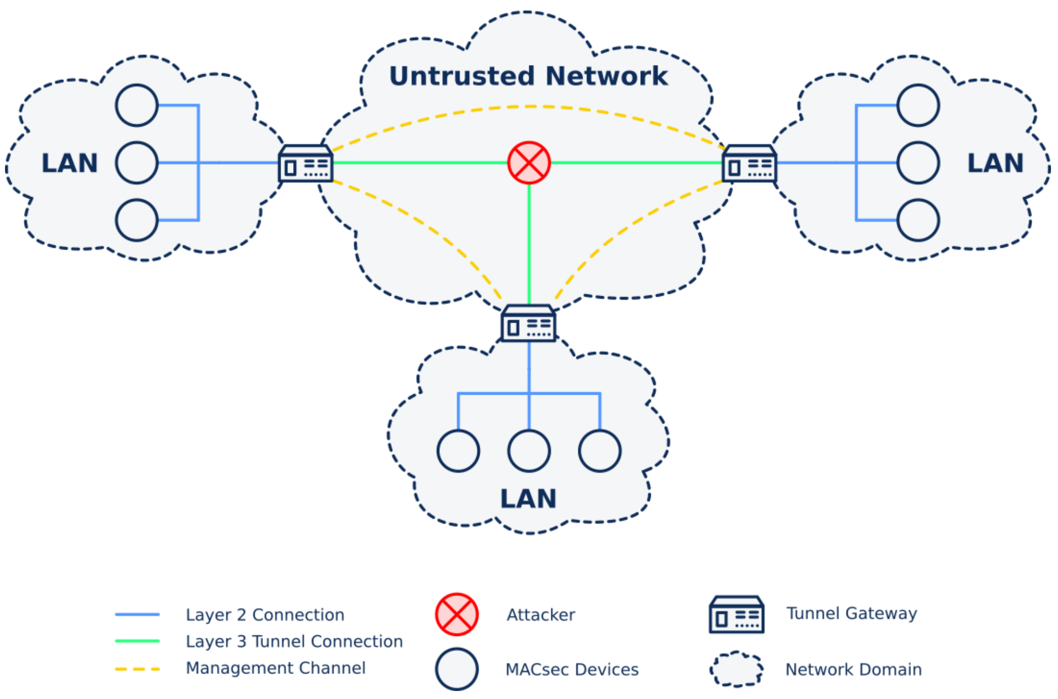


Figure 5.2: Scenario for tunneling of MACsec frames. Local networks are bridged over an untrusted network.

feature of network bridging to MACsec to make it a complete protocol that can be generally applied to our use case.

5.3 Design

To implement tunneling of already encrypted MACsec frames, we designed two approaches. One is a more simple one that only encrypts the headers, while the second one is more complex, but optimized towards performance as much as possible. In the following, we will first describe our scenario in more detail and discuss requirements as well as some further necessary concepts. Based on that, we will first discuss the more complex identifier-based approach, which is then followed by the encryption-based approach.

5.3.1 Scenario and Requirements

We based our design on the scenario shown in Fig. 5.2. We assume an untrusted layer 3 IP network, to which multiple LANs are attached. Tunnel gateways act as interfaces in between. The LANs are populated by devices that speak MACsec. These devices are, just as introduced above, actually encryption gateways protecting legacy industrial machinery. Yet, for the sake of clarity, we abstract from that and just define them as devices that emit MACsec frames.

The basic idea of tunneling is to enable the local devices (circles) to transparently communicate among each other, irrespective in which concrete local network both partners reside. A remote MACsec device should appear as if it was part of the own local network.

The tunnel gateways do not act as MACsec communication partners and only facilitate the tunneling. Only one tunnel gateway can be configured per LAN, as otherwise loops would be possible. We assume the tunnel gateways to be preconfigured, so that they have knowledge of each other and can transmit data between each other.

Tunnel gateways maintain two distinct channels. The first is the tunnel over which actual MACsec frames are transmitted (in green), while the second is a management channel (in yellow) that is used for the exchange of information that is necessary to synchronize the gateways. Since the management channel is not supposed to transport performance-critical traffic, we just assume it is protected using a standard VPN protocol, for example Wireguard. Although MACsec can in principle be configured to not encrypt, we assume encryption is always enabled.

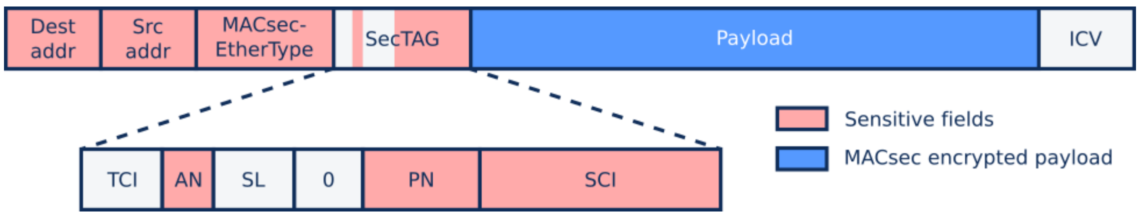
Based on this scenario, our designs are required to implement certain aspects. As discussed above, the security-critical parts of a MACsec frame are its headers. Hence, they must be protected, so that an attacker eavesdropping on the untrusted network cannot gain any information. Second, our designs must authenticate the tunnel traffic. This means that it becomes possible for a tunnel gateway to discern between real and fake traffic including replays. Without that requirement, an attacker could run a DoS attack on the MACsec devices inside the local networks, as the traffic would only be checked for integrity by them. We assume the MACsec devices to be of the type of typical industrial embedded platforms. This means, they are somewhat resource-restricted or rather on the lower end of the performance scale and hence easily overwhelmed by even moderate DoS attacks. Furthermore, we require our designs to allow for point-to-multipoint tunneling, meaning that it is possible for MACsec devices to communicate to remote devices in more than one other LAN, just as depicted in the figure. As a last requirement, we demand the tunnel connections that bridge the MACsec frames to be optimized for performance (in contrast to the management channel).

As already mentioned, we assume the attacker to sit on the network over which the MACsec frames are being tunneled. The attacker can read, modify or drop packets. We do not assume an attacker can drop all packets because this would result in completely different countermeasures and exceed the abilities of networking protocols in general. Yet, we of course assume that some packet loss is possible, as this must not necessarily stem from a deliberate attack but from some random temporary network failure. We also assume the attacker to be able to replay and forge bogus messages. Yet, in our model, the attacker cannot break strong cryptographic primitives.

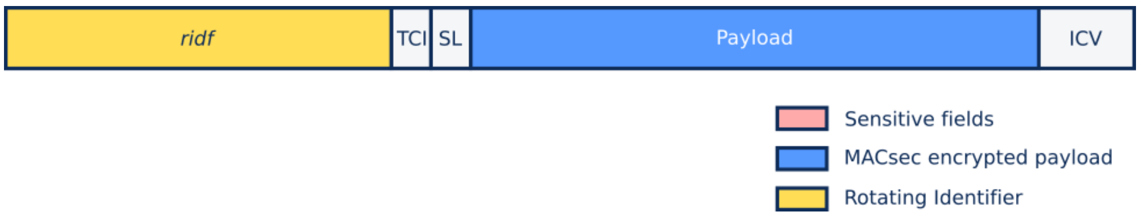
Finally, we will define the concept of flows more detailed, as they are necessary to understand the remainder of our designs.

The necessity to define flows in the first place stems from the fact that the MACsec frames being relayed between the tunnel gateways are not addressed from and to the MACsec devices (the encryption gateways) but the actual end nodes (industrial machines) that they protect. The MACsec devices as well as the tunnel gateways do not know the destination and hence the target LAN of any frame that it sees. When MACsec is rolled out only inside a single LAN, this is not a problem, because all stations are part of the same shared medium or broadcast domain. A MACsec station would see every frame and check the SCI, whether some action was necessary. But, since we do not want to stupidly broadcast all frames between LANs, we need to define some notion of data flows, so that once a frame is identified to belong to a certain flow, it can be relayed directly, analogous to how switches learn outgoing ports to reduce network load.

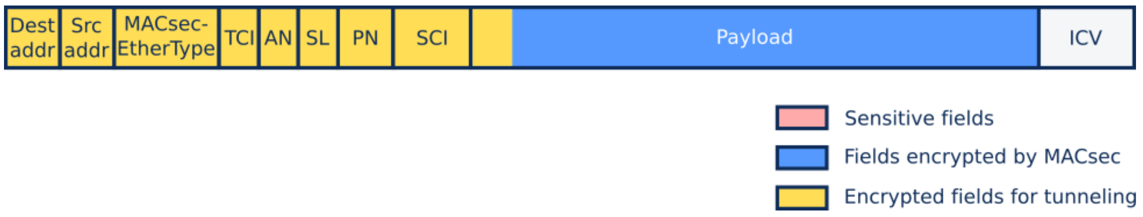
We define a flow as unidirectional traffic from one MACsec device sending to another one



(a) Standard MACsec frame.



(b) MACsec frame modified for identifier-based approach.



(c) MACsec frame modified for encryption-based approach.

Figure 5.3: Standard and modified MACsec frame structures for tunneling. Field lengths are not to scale.

receiving the flow. This is in contrast to the secure channel concept of MACsec itself, where a SC is defined between one sender and multiple receivers. We identify a flow by the SCI and SA fields in the MACsec header together with the destination MAC address. Hence, within a flow, all frames origin from the same source, have the same destination and are encrypted using the same key.

5.3.2 Identifier-based Approach

Both approaches build on the naive approach discussed in the introduction, where a simple and insecure tunneling protocol was used. Hence, in the following, we will assume an underlying protocol, that actually facilitates the tunneling, meaning a layer 2 Ethernet frame is repackaged into layer 3 IP packets and sent to the remote tunneling gateway. All modifications discussed below mean, that the resulting modified frame is transmitted over the untrusted network using some unprotected tunneling protocol, like L2TP or VXLAN.

The problem with the naive approach is, that security sensitive MACsec header information is sent in plain text. The identifier-based approach wants to solve this by replacing these parts of the headers by a random identifier. On first sight, the identifier would not reveal any information to a potential attacker and at the same time would not require any costly cryptographic operations. Simple table lookups suffice at the uplink (sending gateway) and downlink (receiving gateway) and the necessary mappings between identifier and header contents can be exchanged via the management channel.

We classify frames into flows according to the concept introduced above. This results in most fields being constant inside of a flow class. Subsequently, every flow can be represented by one identifier, effectively masking and compressing these fields. The PN is a field that is security-critical and which changes on each frame, but since it is predictable, there is no need to send it in each frame. The TCI and SL fields may change, but as discussed above, they are not critical towards security and hence can just be transmitted without additional measures. Fig. 5.3a gives an overview on the MACsec header fields that are deemed critical.

A tunnel gateway needs to register a new flow when it occurs and share that information with the remote gateways. Then each time a frame of a respective flow needs to be tunneled, the sensitive header fields are removed and the identifier is added instead. This can be done very fast and efficient and hence results in only a small overhead for the tunneling operation. All valuable information is either removed or masked (the payload is already encrypted by the MACsec devices).

Yet, the result of these first considerations is still insecure. The core problem is, that no strong form of authentication is included. All packets of the same flow share the same identifier, making it easy for an attacker to forge valid traffic as it is trivial to predict a valid identifier after having observed one, opening up the possibility of DoS attacks. This also makes it easy for an attacker to link packets to flows, which can be considered a confidentiality violation. Additionally, this approach does not assume packets getting lost or blocked in transit. The receiving gateway can not know if a packet is missing and therefore might reconstruct the wrong PN.

To overcome these problems, we improve on this idea by using rotating identifiers and by adding a receive window. Now, flows are still bound to a base identifier *bidf*, that is also sent to the remote gateways via the management channel. Yet, this identifier is not used directly to replace the header fields. Instead, it is used to derive a rotating identifier *ridf* using a secure derivation function F (e. g. a hashing function) that takes the respective PN as input: $ridf_{PN} = F(bidf, PN)$. The resulting *ridf* is different for each tunneled frame and looks random for an attacker. Additionally, all *ridfs* can be calculated independently of each other as they are only based on the *bidf* and the predictable PN. F should be a cryptographically strong function, meaning attackers should not be able to recover the original *bidf*. Further, it should be collision resistant, meaning it should be very improbable to arrive at the same *ridf* using different inputs. Fig. 5.3b shows the resulting frame.

Replacing sensitive header fields with rotating identifiers makes the protocol more complicated. The flows are managed in tables, shown in Tab. 5.1.The entries will be discussed in more detail below. The uplink (putting an incoming frame into the tunnel) as well as the downlink (receiving a frame from the tunnel) follow specific procedures. Both are depicted in Fig. 5.4. The uplink procedure is the following: the flow, an incoming frame belongs to, is checked based on the SCI, SA and destination Ethernet address and if a new flow is identified, a *bidf* is generated. This base identifier is a simple random number and is sent via the management channel together with the MACsec header fields, it is supposed to replace, to all remote gateways. This includes the PN. Then an appropriate *ridf* is calculated and used to replace the respective MACsec header fields in the frame. The new frame is then sent to the remote gateway.

The downlink procedure includes additional steps. First, when a tunnel gateway receives information about a new flow, it adds that flow to the so-called Flow Table. This table is a key-value list, where the key is the *bidf* and the value consists of other necessary state information,

Flow Table Entry Uplink

Key	SCI | SA
Values	- Unicast *bidf*
	- Broadcast *bidf*
	- Remote Gateways
	- Timeout

(a) Uplink.

Flow Table Entry Downlink

Key	*bidf*
Values	- Header Data
	- Window
	- Next expected PN
	- Bound

Identifier Table Entry

Key	*ridf*
Values	- PN
	- Seen (Replay protection)
	- Pointer to Flow Table Entry

(b) Downlink.

Table 5.1: Necessary tables for the Flow management of our MACsec tunneling approaches.

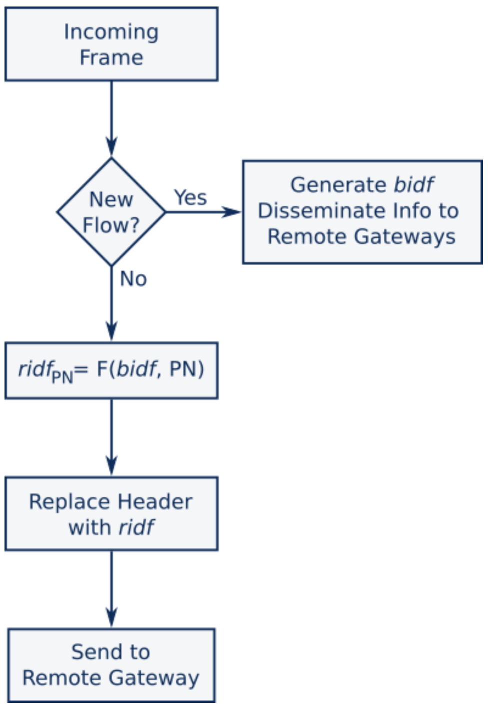
like header values etc. Additionally, the gateway will use the received PN to precalculate a window of next expected possible *ridf*, based on the received *bidf* and PN. These are added to the so-called Identifier Table, where the key is the *ridf* and the value is further information, like the PN. One of those values is a pointer back to the respective Flow Table entry, the *ridf* belongs to.

When a packet arrives from the tunnel, the *ridf* found inside the frame is looked up in the Identifier Table. If the identifier is found, the PN entry and the pointer into the Flow Table is used to reconstruct the original frame. Additionally, the old Identifier Table entry is removed and a new one is calculated and added. The next expected identifier is updated as well.

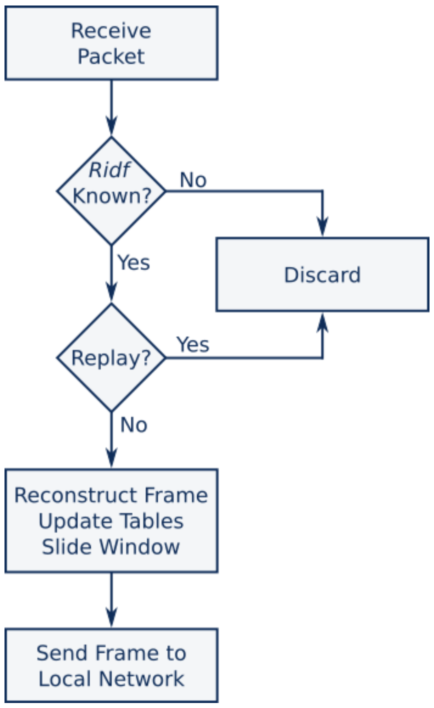
We allowed for multiple possible next *ridfs* to account for packet loss inside a defined sliding window. The Identifier Table is managed so that it always holds the appropriate amount of *ridfs* corresponding to the size of the window.

As described above, tunnel gateways initially do not know behind which remote tunnel gateway the destination of a new flow is. The gateways acquire this knowledge via a two step process. First, incoming frames from new flows are broadcasted to all remote gateways. When the destination MACsec device answers, that triggers the flow discovery process on the remote tunnel gateway. As a result the first gateway learns the destination of the new flow. It registers this information in its Flow Table and sends future frames from that flow only in this direction.

MKA automatically establishes one-to-one SCs with every participant. Consequently, a normal broadcast message is sent over all of the SCs separately. This behavior results in a problem for our flow-based design. Independently of whether a frame is sent to a specific destination address or to the Ethernet broadcast address, the PN is increased on the sending MACsec device all the same. Yet, our tunneling gateways would see different destination addresses (unicast or broadcast address) and would hence attribute these frame to different flows. The increase of the PN would then only be registered at either the unicast or the broadcast flow. As a result, when the next respective other frame arrives, it would be reconstructed using a


Incoming Frame
New Flow?
Yes
Generate *bidf* Disseminate Info to Remote Gateways
No
$ridf_{PN}$= F(*bidf*, PN)
Replace Header with *ridf*
Send to Remote Gateway

(a) Uplink procedure.


Receive Packet
Ridf Known?
No
Discard
Yes
Replay?
Yes
No
Reconstruct Frame Update Tables Slide Window
Send Frame to Local Network

(b) Downlink procedure.

Figure 5.4: Up- and downlink procedures of our identifier-based tunneling approach.

wrong (too low) PN.

To confront this issue, we need to bind both flows together. This is done by registering both flows separately on the tunnel gateways, but by updating the PN as well as the sliding window, when a new frame arrives also on the respective other flow to keep both synchronous.

Finally, we will discuss the fact, that in principle a DoS attack on the encryption gateways can be mounted by guessing correct *ridfs* by generating random packets including random identifiers and sending them to the target gateway. If the guess is successful, the *ridf* will be replaced by an existing valid header and the frame is sent down to the encryption gateway. Yet, there, MACsec's integrity check will detect and discard the bogus frame, as the ICV as part of the payload was also randomly generated and does not match the rest of the frame. Additionally, the attacker must always randomly generate *ridfs*, as he cannot use previously observed ones. Once an *ridf* is processed at a receiving tunnel gateway, it is deleted from the identifier table and the next expected PN of the respective flow is increased accordingly. Reusing an observed packet and only changing the *ridf* would also not lead to a successful attack, as then, if the *ridf* was guessed correctly, the reconstructed frame would still include an ICV that was calculated using an old PN, resulting in a rejection during MACsec's integrity check phase. Additionally, the attacker cannot learn from his guesses, as the tunnel gateways show no reaction in any way, when a frame gets dropped by either the tunnel or the encryption gateways.

Summarily, the success of a DoS attack depends on the possible rate with witch an attacker can guess correct *ridfs*. With a sufficient length of the *ridf*, such an attack should become very inefficient and hence improbable. We will investigate this in detail and evaluate the *ridf* length we chose to implement in the evaluation section below.

In general terms, using random identifiers like our *ridfs* is comparable to using cookies for DoS attack mitigation, like for example Wireguard does. This is an established approach and considered a good defence against this type of attack.

5.3.3 Encryption-based Approach

The previous approach tried to improve the performance by avoiding cryptographic operations like encryption and integrity checks and replacing them with a message-independent calculable single derivation function. Yet, it requires additional steps for flow attribution and binding, which increases complexity of the protocol.

The approach discussed in this section on the other hand will be much simpler by relying on encryption of the MACsec headers. We still assume a secure management channel between tunnel gateways. This allows to use symmetric encryption and we assume appropriate key material has been exchanged when the tunnel gateways were set up.

The sensitive header fields that need to be encrypted are source and destination address, PN, SCI and AN. These amount to 194 bits. As we chose the block cipher Advanced Encryption Standard (AES) for encryption and as it has a block size of 128 bits, this results in two cipher blocks of overall 256 bits length. This is more space than necessary and in fact allows to bluntly encrypt the first 256 bits of the frame. This includes all of the MACsec header and 32 bits of the payload segment, as is depicted in Fig. 5.3c.

We encrypt the second block first and then use its plaintext as well as the resulting ciphertext as additional input for the encryption of the first block. This corresponds to the propagating cipher block chaining (PCBC) mode of operation. We chose this mode in order to provide integrity and confidentiality without additional initialization vectors or integrity check values. This reduces the frame size and saves expensive cryptographic operations. This is possible due to the unique properties of our approach.

As described, the second block includes 32 bits of payload that is already encrypted by MACsec. These 32 bits are always unpredictable for the attacker. MACsec ensures this by using a different initialization vector (the PN) for each frame. These 32 bits are enough to make our second block unpredictable as well. This is due to the diffusion property of AES (which we use for encrypting our blocks). It means, that a change of one bit of the plaintext has impact on all bits of the ciphertext.

Consequently, the ciphertext of the first block also becomes nondeterministic, as we use the encrypted second block to construct it. As a result, both blocks of ciphertext are different for every frame, independently of the repeating header fields. The only requirement hereby is, that the symmetric encryption keys have to change, when any of the tunneled SAs changes, because then the ciphertext of the MACsec payload might repeat.

We assume the gateways to maintain Flow Tables similar to the identifier-based approach above. Hence, the tunnel gateway can send the frame to the remote gateway, which then decrypts it with previously exchanged keys. The remote gateway looks up the header information in the Flow Table and if the entry matches to the decryption result, the frame is put onto the internal network. Otherwise it is discarded.

Finally, we again do not explicitly add integrity protection to our encrypted headers. Instead, the lookup in the Flow Table facilitates that, as a single bit flip during transit would result in a completely different decrypted header. The attacker is not able to generate ciphertext that

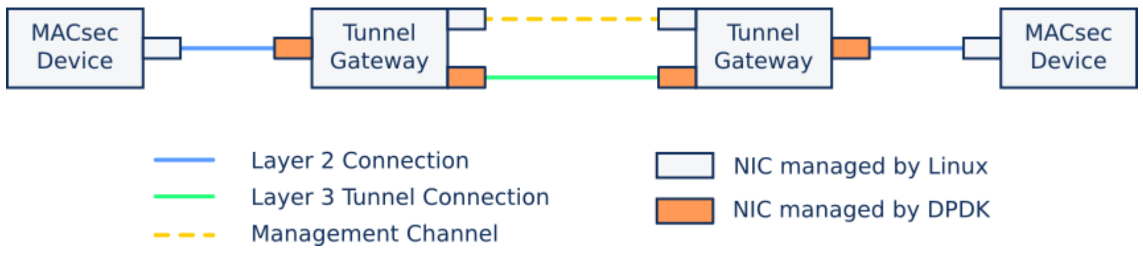


Figure 5.5: Setup for implementation and evaluation of our MACsec tunneling approaches.

would decrypt to a valid header resulting in a successful lookup, without knowing the secret key.

5.4 Implementation

The setup used to test and evaluate our implementation is shown in Fig. 5.5. All involved NICs are Gigabit Ethernet devices. The tunnel gateways were implemented using Netgate MBT-4220 appliances. They are equipped with an Intel Atom E3845 CPU with 1.91 GHz, 2 GB RAM and two on-board Intel I210 Gigabit NICs. The management channel was realised by attaching USB-Ethernet adapters. These interfaces are rather slow due to the USB interface, but as they only processed management traffic, the tunneling traffic, we actually wanted to optimize, was not affected. The tunnel traffic was exclusively transmitted through the on-board network interfaces. Both gateways ran CentOS Linux 8 with a kernel version of 4.18. The MACsec devices were implemented using ODROID-H2 minicomputers from Hardkernel. These come with a quad-core Intel Celeron J4105 with 1.50 GHz and ran Ubuntu 20.04.1 LTS with a Linux kernel in version 5.8. The performance-critical network interfaces in orange were bound to DPDK, in effect hiding them from the Linux kernel. No other system service had access to these interfaces and could interfere with the measurements. We used the igb_uio kernel module driver to manage the NICs.

We required a Transmission Control Protocol (TCP) stack for the Management channel for reliable transmission, but existing implementations on top of DPDK did not prove to be flexible enough. Instead, we decided to use DPDK for the tunnel connection only and add an additional interface through a USB-Ethernet adapter for the management channel. It was not bound to DPDK and therefore allowed to be managed by the Linux kernel. We used Wireguard as security layer.

We used DPDK in version 19.11.2, OpenSSL in version 1.1.1g and libsodium in version 1.0.18. Our DPDK application was compiled using GCC in version 8.3.1 and we always used the compiler flags, recommended by the DPDK manual. Note that our setup is a reduced version of the scenario introduced above. Yet, we of course implemented the steps of the protocol that are concerned with remote gateway management and selection.

To implement our tunneling approaches, we created a MACsec parsing library using DPDK that allowed for extracting information from incoming frames and to construct outgoing ones. To actually transport these frame, we also implemented a prototypical tunneling protocol in DPDK that mirrors the functionality of e. g. VXLAN. The Flow and Identifier Tables described in Tab. 5.1 were implemented using DPDK hash tables. While MACsec frames that transported payload were transmitted using the tunnel connection, MKA (management) traffic was transmitted over the management channel. MKA sends synchronization frames only every second and hence does not strain the channel much. Additionally, MKA traffic is generally not performance-critical.

The derivation function F is the central cryptographic primitive of the identifier-based approach. Hence, it had to be chosen with special care. The main requirement was that it must be keyed, meaning an additional value can be inserted to make it nondeterministic. Function classes that could fulfill this requirement, are encryption and hashing functions. We chose SipHash [30]. It describes a family of hashing functions that is optimized for performance and short inputs and has been designed for use in hash tables and message authentication codes. This corresponds perfectly to our use case and is considered state-of-the-art. We use SipHash-2-4 as these parameters provide maximum security according to the authors. For input, we use a base identifier ($bidf$) of 128 bit length, with the 32 bit PN as hash key. The output is our rotating identifier $ridf$ with a length of 64 bit.

The encryption-based approach was simpler to implement compared to the identifier-based one. For flow discovery, flow management and lookup of remote gateways, we used the same DPDK hash table implementation from above. The header encryption was implemented using AES provided by the OpenSSL library. The decryption step at the remote tunnel gateway was implemented equally straightforward. It includes a lookup to verify that the frame belongs to a valid flow.

During all experimental runs, MACsec was used in encryption mode and it was made sure that, although handling was implemented into the protocol, no MKA traffic happened during the experiments.

5.5 Evaluation

This section will first discuss our proposed protocol designs and then present results of performance measurements done on our prototypical implementations.

5.5.1 Protocol Designs

The main goal of the proposed approaches was to ensure the confidentiality of header information when MACsec frames are being tunneled over insecure networks. Both approaches accomplish that. The encryption-based approach by simply encrypting the whole header and the identifier-based approach by replacing the sensitive fields with a random identifier.

Timing information from traffic patterns are still observable. Yet, this is out of scope as it requires different countermeasures like buffering and sending in regular intervals. These measures would also destroy the performance and such a solution would then not be applicable in industrial environments.

Furthermore, our approaches show no outward reaction to detected wrong packets that an attacker could use to adapt its approach. Malicious packets are simply dropped. As a result, our tunnel gateways also cannot be abused to take part in any kind of reflection-based attack. Oracle attacks, where an attacker pieces information together from multiple communication rounds with the target, are impossible as well.

We did not consider the possibility of attacks from the inside, meaning an attacker that sits in one of the local networks. He could for example inject random MACsec frames that would each time create a flow entry on the tunnel gateway. This would eventually lead to memory exhaustion, because the gateways in our scenario do not know which flows are benign.

Yet, there are countermeasures available, that could be added to our protocols. For example, if MKA was available, gateways could analyze the MKA traffic and derive the benign flows. Another possible approach could be quotas for SCIs or source Ethernet addresses or shorter timeout times, when no remote gateway signals a successful answer to that frame.

Performance-wise, the approaches mainly differ in the cryptographic operations used. For each transport of a frame within the identifier-based approach, the cryptographic hash function has to be called three times. Once on the uplink to create the *ridf* and two times on the downlink to adjust the sliding window by calculating new expected *ridfs*. The encryption-based approach on the other hand needs to encrypt and decrypt two AES blocks. The first approach indeed works with the fewest amount of cryptographic operations possible and through the header replacement it even reduces the effective size of the packets, allowing for bigger payloads. Therefore, on paper, there should be some difference between the performance results.

The other operation which might impact performance, is the Flow Table lookup. Both approaches used DPDK hash tables for that purpose. A lookup takes constant time, independently of the table's size, assuming enough memory is available for them. In our experiments this was always the case, no swapping happened.

In any case, some kind of table lookup is not avoidable anyhow if the scenario considers more than two gateways, as the destination gateway has to be looked up, and considering that broadcasting to all remote gateways is not an option. Furthermore, the lookups realize authorization of incoming traffic on the downlink and provide DoS protection, as we described above.

Finally, we will discuss our choice for length of the *ridf*, as it is directly linked to the success rate of a possible DoS attack on the encryption gateways. We chose a size of 64 bits for the *ridf*, thus, the probability of guessing one *ridf* correctly is $1/2^{64}$. Yet, there are many *ridfs* valid at the same time due to the replay window mechanism and the fact, that a tunnel gateway probably services multiple flows in parallel.

To formalize the number of active identifiers per tunnel gateway, we will assume in the following, that there is a number n of networks that are connected to each other via tunnel gateways and that each of those networks contains a number m of encryption gateways which communicate to all encryption gateways in the other networks. We further assume a replay window of w, meaning that a gateway precalculates and holds w *ridfs* per flow. Finally, we take into account, that for each flow two separate sets of *ridfs* for both uni- and broadcast have to be kept (as described above). These consideration result in the following formula to describe the number of active identifiers per gateway a:

$$a = 2w(n-1)m^2$$

One encryption gateway in a network communicating to all other encryption gateways in the other networks results in $(n-1)m$ flows. Multiplied with m again (this time for encryption gateways within its own network) as well as 2 (unicast and broadcast flows), this results in the number of active flows, that a tunnel gateway has to manage for its network at the same time. Multiplied again with w, we arrive at the number a of active identifiers per gateway. The resulting probability of guessing an arbitrary active *ridf* correctly is thus $a/2^{64}$.

Finally, if we multiply this probability with the rate with which an attacker can inject packets

#	Scenario	RTT ± mdev	Throughput ± mdev
1	VXLAN	3 ± 0.13 ms	837 ± 9 Mbit/s
2	VXLAN + Wireguard	4.85 ± 0.55 ms	245 ± 16 Mbit/s
3	Identifier-based	1.3 ± 0.07 ms	842 ± 3 Mbit/s
4	Encryption-based	1.3 ± 0.05 ms	837 ± 3 Mbit/s

Table 5.2: Performance of MACsec tunneling approaches measured on the Netgate platforms including standard deviation (mdev).

g, we arrive at a model, that allows us to make estimations on the success rate s of the attacker:

$$s = g\frac{a}{2^{64}} = g\frac{2w(n-1)m^2}{2^{64}}$$

For a realistic setting, we will assume a window size of 1000, 10 networks and 50 encryption gateways per network ($w = 1000$, $n = 10$, $m = 50$). This results in 45,000,000 active identifiers per gateway and even an attacker, that could inject an unrealistic 10 million packets per second[1], would only arrive at a success rate of 0.000024 per second. In other words, it would take an attacker a mean time of 26 hours to guess one correct *ridf*. Again, this successfully guessed and wrongly reconstructed frame would be instantly dropped by the encryption gateway after MACsec's integrity check. This is an attack rate, that any encryption gateway should be able to deal with.

Attack rates stay manageable, even in widely overestimated scenarios. For example, if we assume n = 1000, m = 1000, with the same window size and injection rate, the success rate would still be only ~1 per second, showing that the 64 bits we chose as size for the *ridf* should be sufficient for all realistic scenarios.

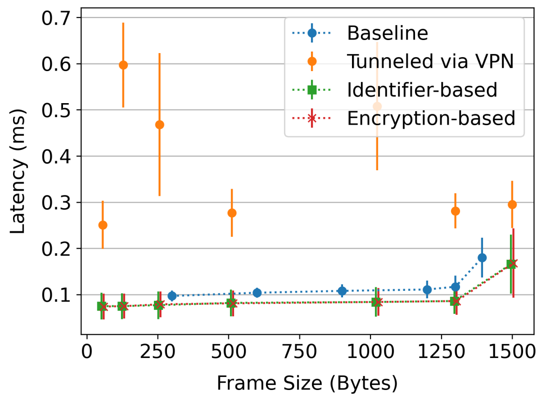
5.5.2 Performance

All of the measurements described in the following were conducted on the setup depicted in Fig. 5.5. The measurements were taken after initializing the connection with a simple ping. This includes the establishment of flows at the gateways as well as necessary ARP[2] resolution at the MACsec devices.

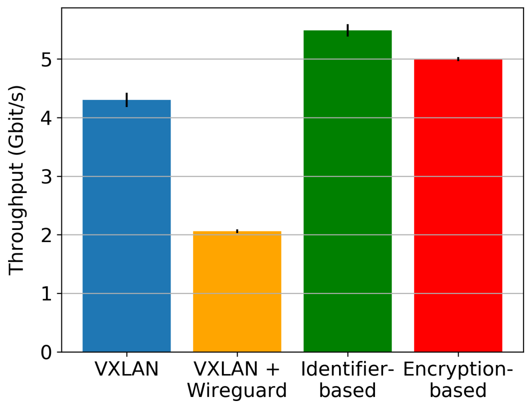
We measured different scenarios. First, we bridged MACsec frames using standard VXLAN. No protection of MACsec headers occurs, as VXLAN just repackages frames and transmits them using UDP. This state-of-the-art approach should incur the least amount of overhead and the measurement results of this scenario shall serve as an indicator of what performance is actually achievable. In the second scenario, MACsec frames were tunneled using a standard VPN protocol. This is the state-of-the-art approach to solve the problem this work is based on. We implemented the tunnel in this scenario using Wireguard, because it is considered highly efficient. Finally, we measured both the identifier-based and the encryption-based approaches as third and fourth scenario.

[1]The data stream would be at least 6.7 Gbit/s, if we assume the smallest possible packet size and ignore certain technical limitations like available networking bandwidth.

[2]ARP - Address Resolution Protocol



(a) Round-trip times with standard deviation.



(b) Throughput with standard deviation.

Figure 5.6: Performance of MACsec tunneling approaches measured on the Xeon Server platforms.

For each scenario, we measured the latency as well as the throughput of the tunnel between the two MACsec devices. Results are listed in Tab. 5.2. The latency was measured as the mean of 65,535 individual ping round-trip times (RTT) with a frame size of 64 bytes. The throughput was measured using the tool iperf3 using an interval of 10 seconds [7].

The results of the first two scenarios show expected behavior. The tunneling gateways are capable of fully saturating the line speed of 1 Gbit/s, when only relaying frames (scenario one), while they showed a huge decrease in performance, when the whole data stream had to be additionally encrypted (scenario two). Latency was increased by over 50%, while the throughput was reduced to one third.

Our approaches showed superior behavior compared to the first two scenarios. Round-trip times were reduced by half, even compared to the first scenario where no encryption took place. Both approaches managed to saturate the line. The slightly bigger achieved throughput of the identifier-based approach stems from the fact, that the replacement with the identifier reduces the absolute header size, making it possible to transport more payload per packet. This ultimately increased the achievable throughput. Yet in absolute terms, the measured differences were small and these performance numbers alone do not give strong indicators on which approach to prefer.

General learnings from the results so far are, that latency can be greatly improved by circumventing the Linux network stack and that cryptographic operations have big influence on the performance if applied to whole frames. Offloading functionality to DPDK improved the performance considerably. The results show, that our approaches greatly reduce the overheads of tunneling MACsec frames compared to the state-of-the-art (scenario two).

We expected the CPUs of the tunnel gateways to be the bottlenecks in our measurement setup. A CPU bottleneck would have led to clearer results concerning the differences in efficiency of our approaches. Instead, the bottleneck were the Gigabit Ethernet interfaces.

Therefore, we measured all scenarios a second time using different hardware for the tunnel gateways. This time we used two server platforms that were equipped with 16-Core Intel Xeon CPUs at 3 GHz, 64 GB RAM and four 10 Gigabit Ethernet NICs on-board. Fig. 5.6 shows the results. Experiments were conducted in the same fashion as before. We additionally measured the latency for different frame sizes, to see how that parameter might affect the results. The jump at high sizes happens when the input frame size exceeds the maximum payload size, resulting in fragmentation, which increases the overall round-trip time.

Scenarios one and two show again mostly expected behavior. Additional encryption incurs immense overheads. As expected, the throughput gets halved from scenario one to two. On the contrary, the latency behavior turned very erratic, showing that the stacking of two standard protocols (VXLAN and Wireguard) can produce additional interference in this unusual multi Gigabit setting. The behavior stems from the fact that on top of the Linux networking stack, in this scenario, two nested protocols are running, which are not properly aligned. Frame size clearly has no influence on the results. A single frame gets queued and dequeued multiple times during transit This unpredictable behavior probably stems from cache misalignments, yet we cannot be certain.

On the other hand, our approaches show superior performance. Both compare vastly better than the state-of-the-art approach in scenario two and even better by some margin compared to the insecure scenario one. And while both approaches show same latency behavior, the identifier-based approach achieves ~10% more throughput compared to the encryption-based approach. This again stems partly from the different header lengths of the approaches. Yet, this time, a slightly better performance of the identifier-based approach can be detected. However, this small performance margin, that is additionally only noticeable using 10 Gigabit NICs, is not enough to justify the increased complexity of the identifier-based approach. Except for special use cases, where performance is of paramount importance, the encryption-based approach should be preferred.

5.6 Conclusion

This chapter investigated two approaches how MACsec frames can be tunneled in a secure and efficient way. One approach was optimized for performance as much as possible, while the second could be implemented much simpler. Results showed that, while both approaches were significantly better compared to the state-of-the-art, the encryption-based approach should be preferred, as it showed roughly the same performance while being a lot less complex, compared to the identifier-based approach.

The work of this chapter still offers pointers towards future research. A non DPDK version running inside the Linux kernel might yield acceptable performance while at the same time increasing the applicability of our approach and allowing for better comparison. Another improvement might be to handle MKA traffic differently. We piped it over the management channel as it was the easiest way, yet it could also be transferred over the tunneling channel. Further integration could also lead to detection of attackers inside the local network. This scenario is excluded so far. Furthermore, certain hardware acceleration features could be researched to increase performance even more. This includes leveraging CPU parallelism, where DPDK provides a lot of potential. We used only one core so far. Finally, even more sophisticated acceleration technologies could be investigated, like for example FPGA-powered SmartNICs.

This chapter also concludes our research efforts in making MACsec applicable as encryption protocol to be used on encryption gateways as is the scenario of this thesis. We enabled MACsec to run on legacy networking infrastructures by adding fragmentation and optimized it by implementing a highly efficient cipher that did not rely on hardware acceleration features rarely found on embedded systems. Finally, we provide a secure and efficient mechanism for MACsec protected frames to be routed across network borders. It is now possible to apply MACsec-powered encryption gateways in a wide variety of application scenarios, without having to mind certain legacy technologies, performance concerns or particular network topologies.

6 Switchbox - A Device to Guarantee Availability in Spite of Gateway Failure

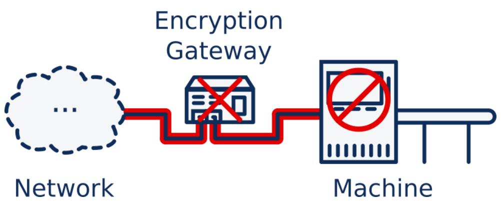
Encryption gateways as a concept have one major drawback, when applying them in the industrial environment. They constitute an additional point of failure. If one of the gateways fails as depicted in Fig. 6.1a, the communication path is cut and the machine it is attached to, is taken offline. In factories, this threat is rated higher than the type of IT security threats that encryption gateways protect against. This assessment hinders the widespread deployment of encryption gateways or, more general, any types of (also non security-related) middleboxes.

This chapter presents a proposal to tackle this problem. The Switchbox, shown in Fig. 6.2, is a physical networking device that can be added to middleboxes, like encryption gateways. Its aim is to increase the availability and reliability of the communication path the gateways protect by removing the gateways from that path in case of their malfunction, as shown in Fig. 6.1b. Although the machine is then unprotected, it can still work and a potential gateway failure had no impact on the productivity of the factory. This changes the initially described security assessment of encryption gateways and as a result, it becomes viable to deploy them in the industrial environment, where high availability has top priority.

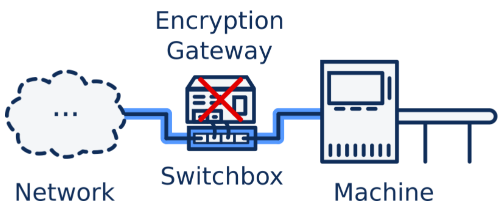
In more abstract terms, this approach effectively solves the conflict between the different protection goal hierarchies (A-I-C vs. C-I-A), that were discussed in Section 2.1.3 and summed up as the second requirement "Protection Goal Hierarchy". This will be discussed in detail below.

This work was a collaboration with a project partner as part of the research project *fastVPN* [49]. While we researched and designed the principle of the Switchbox, the partner designed and prototypically produced the physical device. This work was previously published in [83] and a patent application is pending as well [34].

The remainder of this chapter is structured as follows. Section 6.1 gives further background on the security considerations that form the basis for this work. Section 6.2 will give an overview

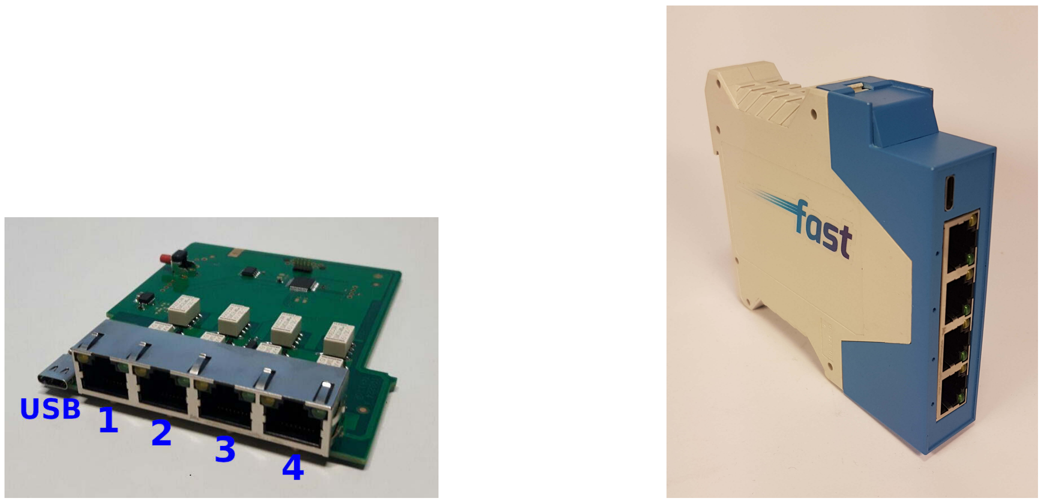


(a) Failed gateway taking machine offline.



(b) Switchbox removing gateway from communication path and reconnecting machine to the network.

Figure 6.1: Encryption gateway failure affecting the attached industrial machine (a) without and (b) with a Switchbox attached.



(a) (b)

Figure 6.2: Switchbox as (a) board only and (b) with casing.

about the related work. Sections 6.3 and 6.4 will first introduce the design of the Switchbox and then evaluate the implementation against a set of necessary further requirements that are derived from the industrial environment. Section 6.5 will then describe and discuss the integration of the Switchbox within the encryption gateway scenario of this thesis. Section 6.6 will give concluding remarks.

6.1 Background

As already described at length in the background chapter, the transition phase industrial automation (IA) currently finds itself in can be described as a clash of different approaches towards security. The main goal of vintage IA is to make a factory work as smoothly and seamlessly as possible. Every disruption, even temporary, must be avoided as much as possible (e. g. no software updates on machines). The communication networks within a factory are designed in that spirit, as in case of a loss of communication, production may be halted (direct financial loss) or equipment may be damaged and personnel may be harmed (safety).

This leads to a general risk-averse attitude towards additions of any kind to the network that might jeopardize the workflows within the factory, like e. g. the encryption gateways of our approach. This results in the most important protection goal of any factory network being availability, meaning the ability of hosts connected to the network to provide their services. It is vastly more important than the other two classic protection goals of integrity (data sent over the network is accurate and consistent or the contrary can be detected) and confidentiality (data sent over the network can only be viewed by authorized peers), that are typically associated with IT security threat models. IT security threats were always largely ignored in the industrial environment, due to the perimeter security assumption, where factory networks were isolated from external threats and no inside attackers existed. Therefore, in IA it is generally considered better to send data unprotected, than to not send data at all and actions towards improving the other protection goals must not impede on the first one (A-I-C).

Yet in the future, more and more paradigms and technologies from the world of information technology (IT) will be integrated into IA and within that world, there is no clear protection goal hierarchy and the three protection goals rather form a triad, where the hierarchy is strongly use case specific. Nevertheless, confidentiality is usually considered the most important protection goal and it is generally understood to be better for a system to not work at all, than to work in an insecure state (C-I-A). Typically, IT systems are not generally trimmed towards high availability and system crashes and necessary reboots tend to be more frequent. These systems are more complex, which allows for misconfiguration and, compared to industrial control systems (ICSs), generally break more often.

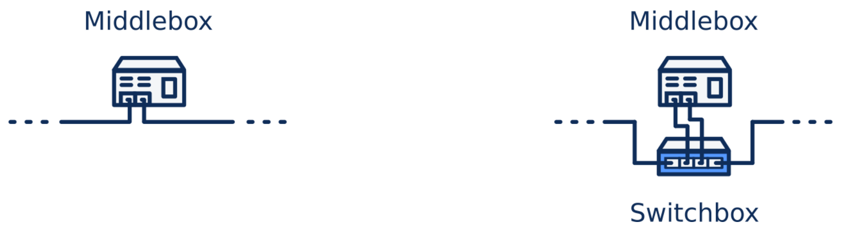
Middleboxes, which originate from the IT world, do not treat availability with high priority. They enable a variety of use cases and bring the adaptability of the IT world, believed necessary to implement the goals of Industry 4.0. Especially in the case of encryption gateways, they are added to the critical communication path of data flows within a factory. These gateways tend to be typical IT systems with the typical IT woes (extensive software stack, attack surface, broken or missing updates etc.), more focused on providing a variety of services than reliability. This then goes directly against the main principles within IA (A-I-C).

As a result, the adoption of middleboxes is hindered, as they are generally considered additional points of failure, deemed too risky for industrial application. There is a dilemma between the goals of adding new features in the form of middleboxes and maintaining safety and security within industrial networks. Yet, to successfully transition to modern highly integrated networks that correspond to the vision of Industry 4.0, it is necessary to provide a solution to this predicament.

6.2 Related Work

The latest trend in networking technology is software-defined networking (SDN) and there is a huge amount of research going on in the direction of introducing SDN into the industrial environment. However, SDN requires a whole infrastructure consisting of intelligent switches and servers acting as SDN controllers. These cannot be considered standard (especially in legacy environments). Therefore, the introduction of SDN would incur heavy financial costs. It would also hugely increase the amount of hard- and software, that needed to be relied upon for service delivery. Finally, SDN equipment is generally not (yet) designed and built for assurance of high availability.

There is some research going on in the direction of network bypass devices [113, 110, 114, 74, 28]. These relay data traffic depending on certain circumstances or availability of certain services. Yet, none of these employ cryptographic means to protect their switching signals. They trust the correct behavior of their controllers or rather the correctness of incoming commands. These solutions are typically also less suited for retrofitting, as they make certain assumptions about their environment (e. g. concerning power supply and type of networking technology). Furthermore, they describe no provisions in case of middlebox failure and they are also not designed towards low latency. This makes them unsuitable for the industrial environment.



(a) (b)

Figure 6.3: A middlebox in a network (a) without and (b) with a Switchbox attached.

6.3 Design

This section presents the design of the Switchbox. The set of requirements as well as the attacker model, on which the design is based, will be discussed first. This is followed by an introduction to the architecture of the device.

6.3.1 Requirements and Attacker Model

To focus the design, we formulated a set of requirements derived from the industrial environment, described previously. The solution must:

1. *Add-on:* be able to be added to existing infrastructures with already deployed middleboxes,
2. *Security:* provide additional availability for the communication path under a certain attacker model (see below), while not adding too much complexity (and hence error proneness or attack surface),
3. *Industrial environment:* be able to be deployed in an industrial setting,
4. *Performance:* impose only minimal performance overheads and detect and react to middlebox failure fast.

The *attacker model* we considered did not include physical attackers within the factory, as those could also just manipulate the relevant cabling or even the physical process directly, voiding any possible security measures. We also did not consider sophisticated attackers, that could overtake or infiltrate the middlebox as they would then be able to do any manipulation they liked anyway without being detected. We considered attackers trying to disrupt the service of the middlebox (e. g. through encrypting ransomware or denial-of-service (DoS) attacks) as well as faulty behavior of the middlebox introduced by software faults and hardware failures.

6.3.2 Device Architecture

To protect the availability of a connection when a middlebox is present, we designed a hardware component, the Switchbox, that can be put in front of the middlebox. It would either let data traffic pass to the middlebox or not, depending on certain circumstances. We assumed the middlebox to have two Ethernet ports, where traffic is patched through (Fig. 6.3a).

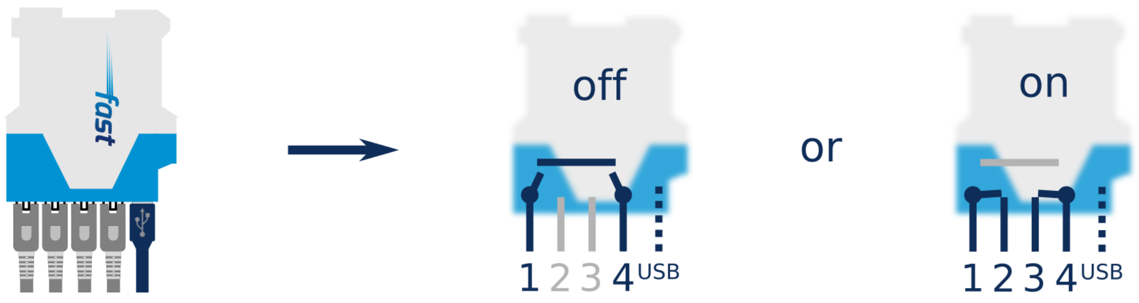


Figure 6.4: Switchbox switch states.

The Switchbox consists of four Ethernet ports, relay switches, a small embedded microcontroller and a USB port (see Fig. 6.2).

The relays physically (galvanically) switch the Ethernet ports onto each other in two different possible configurations, leading to two distinct states. The first state is “off”, where ports 1 and 4 are connected to each other, while ports 2 and 3 are not connected to anything (see Fig. 6.4). This means that traffic coming from port 1 is patched through to port 4 (and vice versa), while traffic from ports 2 and 3 is ignored. In the setting shown in Fig. 6.3b, this would mean, that the traffic does not flow through to the middlebox. The second state (“on”) patches ports 1 and 2 as well as 3 and 4 together, resulting in traffic being patched through to the attached middlebox.

The CPU is an embedded microcontroller (Atmel ATSAMD21G18). It does not run a big general purpose operating system, but only a small specific firmware, that is flashed on once.

The Switchbox is attached to the middlebox via its USB port, which is used for command, control and power supply.

The default state of the Switchbox is off. It only switches on when certain criteria are met:

1. powered on,
2. heartbeat message received within a certain time span,
3. successful periodic challenge-response procedure, where the middlebox authenticates itself to the Switchbox.

Power as well as all control messages are transmitted via USB. Regular heartbeats from the middlebox proof to the Switchbox, that the middlebox is alive and the challenge-response procedure ensures, that heartbeats are not faked. Owing to the computational restrictions of the embedded microcontroller, the heartbeats are sent without cryptographic protection as they could not be authenticated very frequently. Instead, an additional periodic challenge-response procedure was added, so that authenticity could be checked (less frequently, but still) continuously. Therefore, the middlebox must actively cooperate by running a software daemon.

Due to the relays physically switching the ports, the Switchbox is always in one of two defined states. Even during a spontaneous power outage of the middlebox, the relays fall down to state “off” and connectivity is preserved.

6.4 Evaluation

This section evaluates the design introduced against the set of requirements proposed in the previous section.

6.4.1 Requirement 1: Add-on

We opted for a piece of actual hardware, as it could be easily added to existing installations. A conceivable software solution using SDN would not be feasible, as respective hardware and software tools are not yet widely deployed in factories (and will not, for some time). The small software daemon necessary on the middlebox (for heartbeats and challenge-response) is an acceptable cost, as middleboxes typically do not need to be re-certified when software updates happen and should feature a modern modular software environment, where such additions should be easily possible. Accordingly, a USB port should be available on any reasonably modern middlebox.

The design makes only minimal assumptions about the environment. No additional power supply is needed and, as the Switchbox just physically switches the ports and no logical access to the data happens, it can also be used in Power over Ethernet environments and is generally agnostic to the actual networking technology. It could be used with any copper-based medium, like fieldbuses or even digital subscriber line (DSL).

6.4.2 Requirement 2: Security

When a middlebox stops working due to faults, failures or attacks (according to our attacker model), the communication path the middlebox sits on would be broken, effectively nullifying the availability of any attached end point. However, an attached Switchbox would register the loss of heartbeats and would then take the middlebox out of the communication path, effectively re-establishing the path and hence achieving the goal of preserving availability of the communication path in spite of a middlebox breakdown.

Adding features to a system generally introduces more complexity, resulting in an increase of the trusted computing base (TCB) as well as the attack surface of the whole system. The TCB is considered to be the set of hard-, firm- and software components that are critical to the security of a system, while the attack surface is considered to be the set of possible entry points for an attacker to a system.

The Switchbox can remove middleboxes from the communication path rendering them useless. In case of the middlebox providing security-related functionality, the correct operation of the Switchbox becomes critical. So, the additional protection the Switchbox offers (described above), must, in light of the security considerations of the overall system, be weighted against the additional complexity it also induces. Therefore, in the following, we will discuss the potential increases in attack surface as well as TCB induced by the application of the Switchbox.

Attack Surface

The Switchbox is only connected to its corresponding middlebox via USB (Ethernet traffic is only patched through), making the middlebox the only communication partner and solely responsible for defining the state of the Switchbox. Consequently, to successfully attack the Switchbox, the middlebox must be penetrated in a way such that heartbeats as well as the

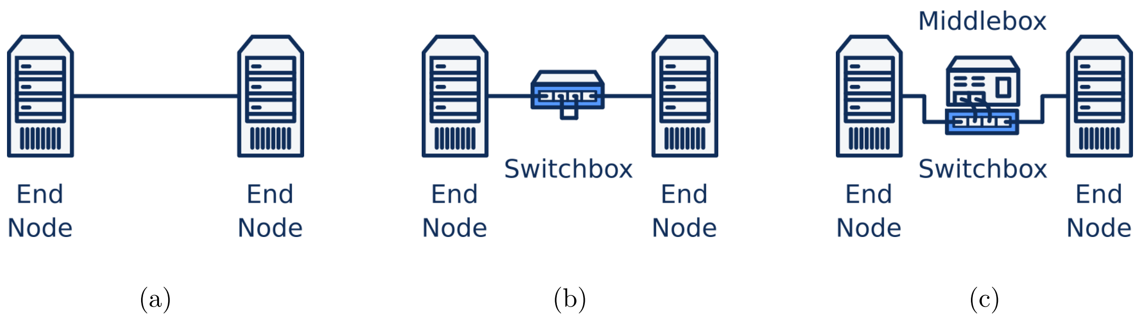


Figure 6.5: Settings for performance evaluations of the Switchbox (a) without a Switchbox, (b) with a cable and (c) with a transparent middle-box attached.

challenge-response procedure are kept intact. Yet, if the middlebox is taken over by such an advanced attacker, the attacker could just as well make the middlebox itself perform the attack (e. g. drop traffic).

Furthermore, the heartbeats that dictate the state of the Switchbox must be protected, so they could not easily be faked, provoking wrong behavior. The challenge-response procedure is done through the same interface (USB) as the heartbeats and can therefore work as a safeguard against faked heartbeats, ensuring the authenticity of the heartbeat source (the middlebox at the other end of the USB connection).

Therefore, the introduction of the Switchbox does not increase the overall attack surface.

Trusted Computing Base

To be able to send heartbeats and do challenge-response, the middlebox requires an additional software component. Our prototypical implementation consisted of ~2000 source lines of code (LoC) (excluding linked Linux standard libraries). This small amount compared to the full software stack of a contemporary middlebox does not significantly increase the overall amount of software that needs to be trusted.

The Switchbox itself must be trusted. Yet it only consists of very reliable industry grade hardware components (see below) and a small firmware. Our prototypical firmware had a size of ~50 kB. Once deployed, the firmware can be locked, meaning the internal flash memory is set to write-protected. This then can only be reversed with special equipment. We deem this increase in TCB as small and, compared to the gained increase in security, as cheap.

6.4.3 Requirement 3: Industrial Setting

All hardware components used within the Switchbox (embedded controller, relays, Ethernet ports, wiring, casing) adhere to the safety requirements for industrial equipment as laid down in IEC 61810 [24]. This standard describes certain tolerances to external factors, the equipment must endure. These factors consist of e. g. extended temperature ranges, power surges, vibrations or physical shocks.

6.4.4 Requirement 4: Performance

To test the performance of the Switchbox, different tests were conducted. In the following, measurements on the latency and bandwidth overheads induced by the Switchbox as well as detection and reaction times of the Switchbox are presented.

Induced Overheads

Latency and throughput of the Switchbox was measured using a JDSU 6000 network equipment testing device [8]. This device runs tests according to the RFC 2544 standard [35]. The test setting used is shown in Fig. 6.5b, where both end nodes are embodied by the JDSU. Both Switchbox states (on and off) were tested separately. To make the results comparable, a baseline measurement was also conducted. This setting is basically the same, just without the Switchbox attached (see Fig. 6.5a).

Each test run consisted of a 10 second duration, where the testing device tried to saturate the device under test with 1518 byte frames. Tab. 6.1 shows the average latency, jitter and bandwidth during those runs. As the jitter is 0.00 μs for each experiment, the average measurement values given reflect the Switchboxes behavior very accurately. The minimal differences in latency stem from the physical path of the connection being slightly longer for each consecutive experiment. A difference in throughput could not be measured at all.

These results show that the Switchbox practically does not induce any performance overheads, as no networking logic is employed to route packets or frames. Only electrical currents are switched.

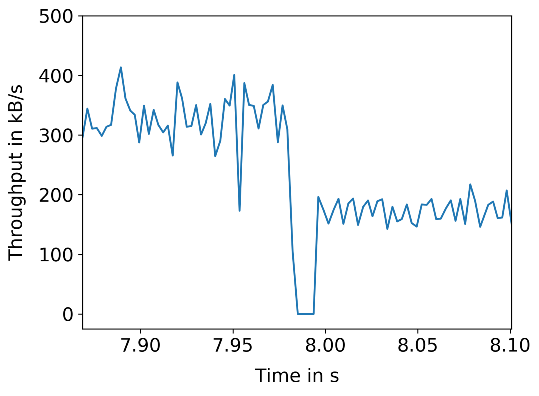
Detection and Reaction Times to Middlebox Failures

Before the Switchbox can react to a middlebox failure by switching, it first has to detect the failure. As the aliveness of the middlebox is constantly checked via heartbeats, we investigated the smallest possible heartbeat interval the Switchbox could manage, to find the shortest possible detection time. Our tests showed, that the embedded controller on the Switchbox allowed for a smallest possible heartbeat interval of 9 ms, leading to a lowest possible expected mean detection time of middlebox failures of 4.5 ms.

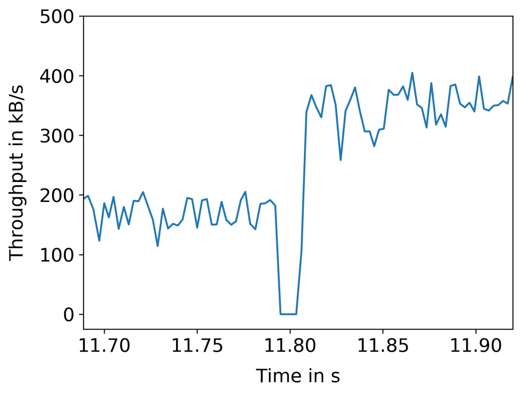
To evaluate the reaction time of the Switchbox, we measured the duration of link loss on the communication path during a switch. We used the test setting shown in Fig. 6.5c, where we generated a data stream on one end node and measured the throughput (over time) on the other, while the Switchbox did switching operations. We expected to see a sharp decline in throughput for the time the switching operation was conducted. 2 PCs with Intel Core i5 processors (2.30 GHz and 3.30 GHz) were used as end nodes in this setting, while the middlebox was simulated by using a PC with an Intel Core2 Quad (2.66GHz) with two bridged Ethernet ports just relaying the traffic.

	Baseline	**Switchbox Off**	**Switchbox On**
Latency (μs)	25.61	25.62	25.63
Jitter (μs)	0.00	0.00	0.00
Throughput (Mbit/s)	999.91	999.91	999.91

Table 6.1: Performance measurements of the Switchbox in both switch states as well as a baseline for comparison.



(a) Switch on.



(b) Switch off.

Figure 6.6: Duration of link loss during switching of the Switchbox.

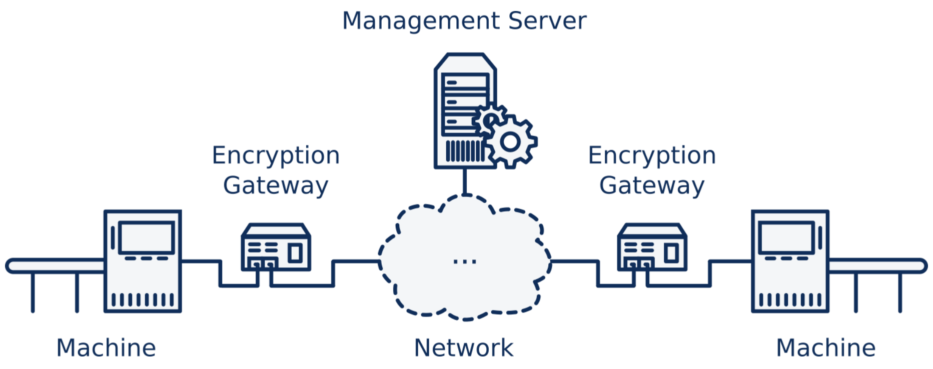
Exemplary results for a switch on as well as a switch off operation can be seen in Fig. 6.6. The figures show the throughput between the end points over time during switching of the Switchbox. Connection is lost on either switch for ~11 ms. After the switch on (and before the switch off) operation, the traffic flows through the middlebox, resulting in reduced throughput, as in this case the middlebox has to handle each frame additionally as well.

These measurements combined, result in a worst-case time of 20 ms to re-establish end-to-end communication in case of a middlebox failure. Such a short link loss should not lead to connection losses between real end points as only single packets could get lost. These packet losses should be detected and repaired by higher layers such as TCP/IP. In fact, when tested with slower hardware (Raspberry Pis), a link loss in many cases could not be detected at all.

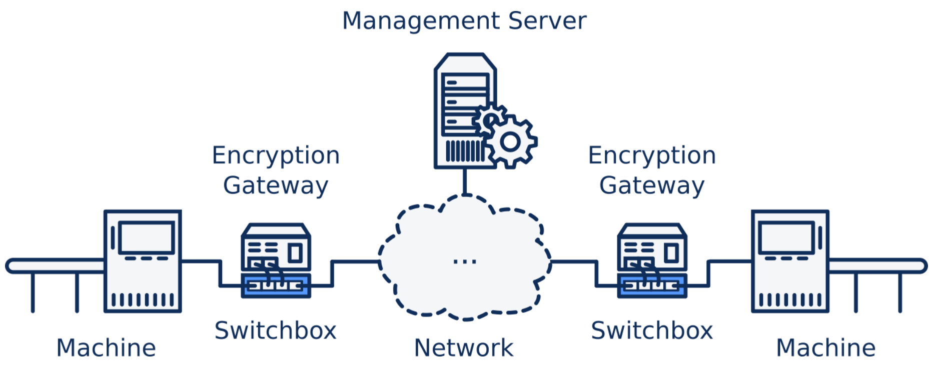
6.5 Integration with Encryption Gateways

This section finally integrates the Switchbox with the encryption gateway use case of this thesis, where the basic idea is to overlay an existing network communication path with an additional security layer by retrofitting encryption gateways to legacy end nodes, like industrial machines or programmable logic controllers (PLCs) (see Fig. 6.7a). Gateways transparently encrypt and decrypt the data traffic sent between the machines so that the data is always protected in transit over the network. For this scenario to work in practice, gateways must cooperate and a failing gateway on one side must be detected, so that the opposing gateway can be reconfigured, i. e. by stopping to encrypt outgoing data and accepting unencrypted incoming data. This requires a central management server to be present, that orchestrates and coordinates the encryption gateways. This server would also constantly check the aliveness of its clients (via e. g. heartbeats) and act accordingly if a gateway goes offline.

Encryption gateways are put between the end nodes that they are supposed to protect and the network. In effect, they break the direct communication link of the nodes to the network and to other end nodes. Traffic data must be actively processed by the gateways for the communication link between end nodes to work. A failure in a gateway would make its end node physically inaccessible and would effectively remove the end node from the network.



(a) No Switchboxes deployed.



(b) Switchboxes deployed.

Figure 6.7: Standard encryption gateway scenarios with and without Switchboxes deployed.

This necessarily leads to a trade-off when assessing the application of encryption gateways. The added confidentiality (of the data traffic) is weighted against the increased risk towards the availability (of the communication path). As in industrial environments availability of communication is typically considered much more important than its confidentiality (A-I-C), this assessment leads in most cases to the exclusion of the additional security measure.

However, this assessment can be changed by augmenting each gateway with a Switchbox, as depicted in Fig. 6.7b. Now, the Switchbox would detect the failure of the gateway and would then bypass it by switching to the off state. This would then reconnect the end node to the network. The management server would then, after detecting the absence of the broken gateway, configure the other gateway to stop encrypting and to accept unencrypted traffic, effectively re-establishing the connection between both end nodes, albeit in an unprotected fashion.

This fundamentally changes the security assessment of the overlaying security scheme, making it possible to rationalize the deployment of encryption gateways in the industrial environment in the first place. In summary, the Switchbox makes it possible to actually implement the policy of availability before confidentiality (A-I-C) instead of the present availability without confidentiality, as now in the good case confidentiality of data can be established while in the bad case availability can be guaranteed.

6.6 Conclusion

Within this chapter, we investigated a general approach on how to incorporate middleboxes into industrial networks via a hardware switching device that removes the middlebox physically from the communication path by only patching traffic data through, if the middlebox is recognized as alive. This allows data to flow even when the middlebox fails, achieving the goal of allowing the middlebox to provide its service while at the same time guaranteeing the availability of the communication path in case of malfunction or failure.

In combination with encryption gateways, the Switchbox concept makes it possible to implement advanced security policies in an industrial environment by reconciling opposing protection goal hierarchies.

We showed that our proposed solution only incurred minimal performance penalties on the communication while at the same time not significantly increasing the TCB as well as the attack surface of the complete system. With the help of our solution, the security assessment of middlebox deployments in industrial settings significantly changes. This makes such deployments more likely in the future.

7 Configuration of Encryption Gateways via Hardware-based Security Tokens

Encryption gateways represent in themselves an additional layer of infrastructure within the factory. Introducing them means to add another system that needs to be managed and maintained by factory staff, increasing the overall complexity of the factory in the process. Additionally, security management in itself is generally a rather complex and error-prone process. Configuration errors happen even to expert staff leading to vulnerable networks.

Non experts, on the other hand, often see any security measure as mere overhead or hindrance to the "actual" work that needs to be done. This then typically leads to setups, where security measures are configured in such a way that they interrupt the productive work the least. This often leads to configuration rules that do not restrict behavior in any way and hence offer no practical increase in security. This is especially true in the industrial environment, where staff is often not trained in these matters, as the topics of IT security or network protection were not deemed relevant in the past. Yet, as factory networks will become more complex, by e. g. employing encryption gateways, these problems will only increase in the future. Consequently, the usability will be a key factor in the successful implementation of any security-related scheme. We already discussed this matter in Chapter 2 and summed it up as requirement 3 "Usability".

Therefore, in this chapter, we present a novel approach on how to configure encryption gateways. We aim for a solution that is very understandable and actionable by staff that is not trained in IT security matters and is still secure. To this end, we use hardware-based security tokens to reduce the configuration of a secure channel to one physical action that does not require further interaction with any software user interface.

This work is, at the time of writing this thesis, being published in [81].

The remainder of this chapter is structured as follows. Section 7.1 gives more insights into the importance of usability for security-related mechanisms in industrial environment and introduces our general approach. Section 7.2 surveys some related work. Section 7.3 first motivates our design by presenting certain goals, we based our design on, and then presents our scheme. A prototypical implementation of our scheme is presented in Section 7.4. Section 7.5 evaluates the design as well as the implementation. Section 7.6 gives concluding remarks.

7.1 Background

Today, many industrial systems and factory networks are vulnerable due to bad configuration. A recent study by Dahlmanns et al. showed that Internet-facing industrial control systems (ICSs) are frequently configured in a way, that makes them or the end points that they connect to susceptible to attacks [41].

In more detail, the researchers scanned the whole Internet (whole IPv4 address range) for OPC UA servers. OPC UA, short for Open Platform Communications Unified Architecture, is a modern standard for industrial communication that among other things also offers state

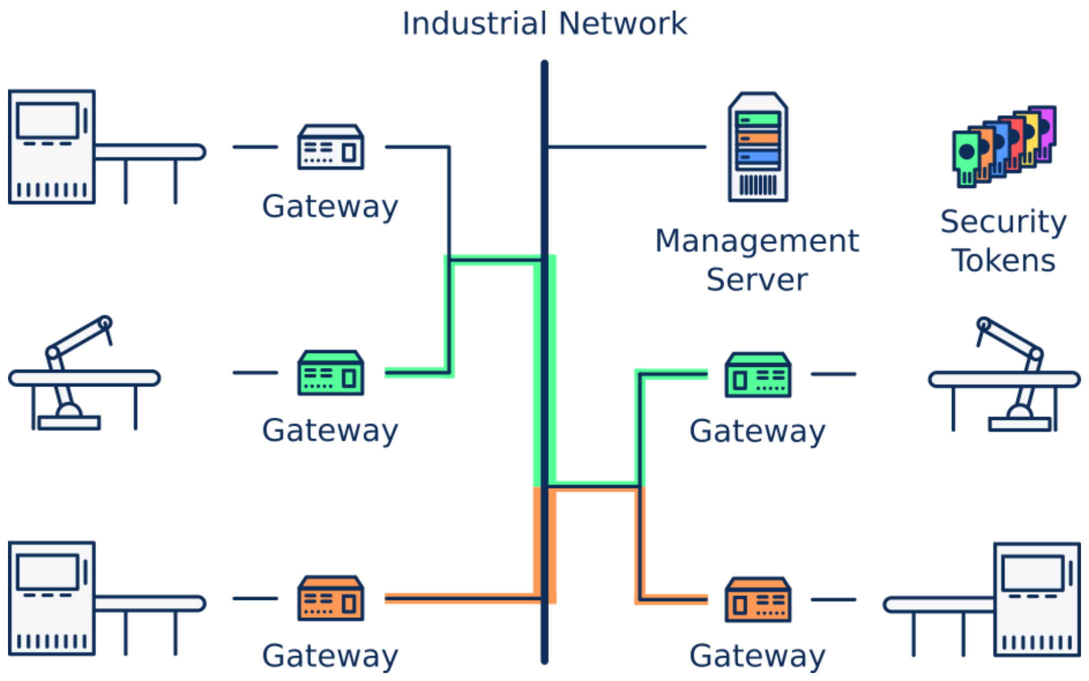


Figure 7.1: Multiple secure channels protecting communication traffic of end points in an industrial network. The gateways are managed by a central management server. Secure channels can be configured via security tokens.

of the art encryption. The authors showed that 26% of the OPC UA servers connected to the Internet were configured without any access restriction. Further 25% of servers used a deprecated and now considered insecure hash function (SHA-1). On many servers that used certificate-based authentication, the study found that identical certificates were used, probably due to the operators of these systems just copying the certificate from another server during provisioning. The manual provided by the vendor of the OPC UA server was obviously not consulted. The authors of the study then went on to alarm the vendor, who in turn notified his customers, the operators of the servers, about this security incident. Yet, this did not result in an update of those systems. Additionally, 44% of all OPC UA servers on the Internet allowed unauthorized users to read and write values from industrial devices and execute system functionality. Only 8% of systems were configured correctly.

In summary, this study shows a deep divide between the potentially achievable security levels offered by modern ICSs and the actual levels of security in deployed systems in practice due to the bad configuration of the available security mechanisms. Additionally, the study hints at a lack of understanding about network security topics at the side of the factory operators as they did not update their vulnerable systems even after being informed about the weakness. As this study scanned the whole Internet, these problems must be considered industry-wide.

As already described above, the encryption gateways of this thesis' approach are added to the network and provide an additional layer of security. To be effective in practice, numerous gateways have to be distributed within the factory, which then require some kind of mechanism that allows for regular reconfiguration of the secure channels they provide.

For our scenario, we assume a central server, that manages the deployed gateways, see Fig. 7.1 for reference. Secure channels can be configured dynamically depending on the current work schedule of the factory and multiple channels can be active at the same time. Although not pictured, a single secure channel can be configured between more than two gateways. A gateway, where no secure channel is configured, does not touch the traffic and just transparently patches it through.

The standard way to configure such as setup is a management software with some graphical user interface (GUI) running on the server, where gateways are listed and an operator can configure the secure channels. In a best case scenario, the gateways are represented with some form of graphical icons, but more typically these interfaces are rather text-heavy and based around drop-down menus. In any case, this type of interface is already pretty abstract, virtual and divorced from the actual physical reality of the factory.

Normal functional configuration of factory equipment is already complicated and a security measure, like encryption gateways, adds yet another layer of complexity. Additionally, the secure channels are going to be frequently reconfigured as these are dependent on the concrete schedule the factory is executing at a certain time. Being able to configure them with ease is therefore paramount for the acceptance and adoption of encryption gateways in industrial networks as a whole.

The easiest and most comprehensible way to configure these secure channels would be by using a physical token, which is plugged into each respective gateway. This would mimic established security protocols from the real world, namely locking a door with a key, where the key is the secret in need of protection. This would immediately make sense for anyone, including people not classically trained in IT security matters. We believe, the approach of simplifying the usability as much as possible is an important contribution to increase the efficacy of the whole encryption gateway approach.

This scheme is not supposed to supplant classic GUI-based configuration schemes, but to augment and improve them. We still assume some kind of graphical configuration interface on the central management server.

7.2 Related Work

Hardware security modules (HSMs) are a special class of devices. They are fully independent computing devices equipped with their own microprocessor and memory. They can store secrets safely and use them for cryptographic functions to provide various security functionalities, like issuing digital keys, encryption of data, digital signatures, secure storage or authentication.

A specific subgroup are security tokens. These are typically used to authenticate its holder to gain access to some resource. Therefore, they are also called electronic keys. They range from wireless key cards for opening doors to tokens provided by banks offering additional protection for online banking services.

A common use case is to bind a software product to a so-called security dongle. These are hardware tokens that have a certificate stored and which have the ability to attest the existence of that certificate without it having to leave the token. These dongles must be plugged into the device wishing to execute the bound software application. Entire business solutions are readily available, that comprise of dongles, software libraries and cloud authentication services. The CodeMeter product line from the company Wibu-Systems might serve as good example [117].

Yet, these types of approaches have in common that they only support a static mapping of token to resource. Tokens are just used as physical embodiments of a digital certificate and their presence always unlocks the same resource. The tokens of the scheme presented in this work on the other hand, must be able to be dynamically mapped to different resources (secure channels) and interact with multiple devices (encryption gateways) interchangeably.

Another common use case for security tokens is two-factor authentication (2FA). There, users are only granted access to a remote service, when they provide a password as well as a second type of identification. In this case, a token that can authenticate itself against the remote server. The application of 2FA in online authentication processes has been increasing in the last years, as more and more online service providers start offering this type of remote authentication. Yet, to the knowledge of the authors, there is no work that employs those security tokens in a similar usability concept as proposed here, not in the industrial environment or anywhere else.

The closest work we could find, was a commercial industrial router, where a classic physical key has to be plugged in and turned to a certain position to allow a remote party to access the network [92]. The physical action of turning the key is used to signify the unlocking of the network behind the router. This goes in the same direction as our approach in that it reduces a complex security configuration to a simple physical action. Yet, since this is just a normal physical key, it can easily be copied and, should it get stolen, the router has to be replaced. Furthermore, this approach does not cover our use case, as we want to propose a solution for encryption gateways and not routers.

7.3 Design

Section 7.3.1 will present design goals derived from the observations above. These goals on the one hand try to formulate in detail why and where exactly security measures must be applied to produce trust in the overall system. On the other hand they try to formulate what exactly is meant with the general goal of usability. These design goals form the cornerstones for our design, which is presented in Section 7.3.2.

7.3.1 Design Goals

To achieve the goal of configuring secure channels on the encryption gateways using only physical tokens, certain preconditions must be met, that revolve around establishing trust relationships between the distributed entities of the system (server, gateways, tokens). Therefore the following design goals include the establishment of trust, based on which further design goals will be defined.

Trust between Server and Gateways

The encryption gateways are scattered within the factory and are connected via the factory network to the server. We consider this network insecure, otherwise there would be no need for encryption gateways in the first place. Therefore, the gateways must build a trust relationship with the server, so that they can authenticate themselves when connecting to the server remotely. This trust can then form the basis for a secure management channel, where the server receives status updates from and can send configurations to the gateways.

Trustworthy Tokens

The main idea of our scheme is that just a token should suffice to establish (or tear down) a secure channel between two gateways. Therefore, these tokens must not be forgeable and it should not be possible to just copy or replicate them. To this end, they should have some kind of unique identifier so that they can always be also physically accounted for. Furthermore, they should be able to authenticate themselves remotely against the management server so as to be able to build a trust relationship.

Trustworthy Configuration

When trust between the server and the gateways as well as between the server and the tokens has been established, the scheme must make sure, that only trustworthy tokens connected to trustworthy gateways are allowed to change the configurations of secure channels.

Ease of Use

The everyday use, meaning the setup and tear down of secure channels should be as easy as possible. One physical action should be enough and no interaction via some software interface should be necessary.

Life-cycle Management

Any scheme of that kind can only hope to be implemented in practice, if it is possible to manage the whole life-cycle of all the components. This means that it should be possible to also remove gateways and tokens from the system without compromising the security of other still connected parts. Especially, the scheme should be able to deal with a token getting lost, broken or potentially stolen.

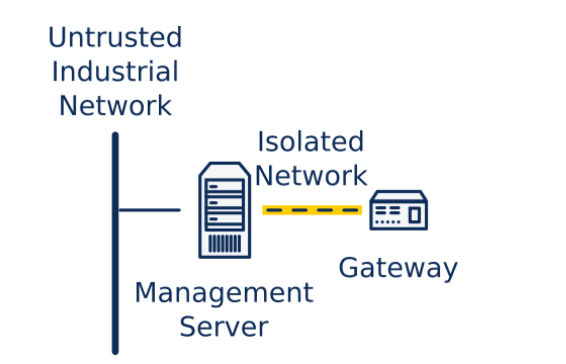
Attacker Model

The general aim of our scheme is to increase the usability of encryption gateway-based systems. Yet, security still has the highest priority and the security of the encryption gateways must not be compromised in any way. Therefore, no new attack vectors shall be opened and increases in attack surface shall be as minimal as possible.

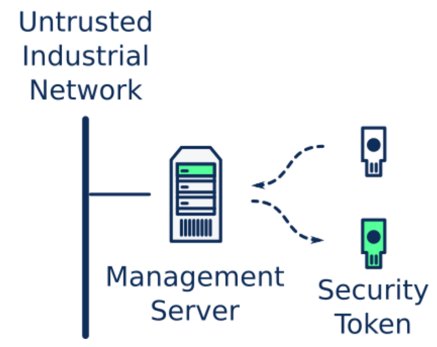
For our design, we assume an attacker that has physical access to the network. He can steal tokens and maliciously use them inside the factory. Yet, he cannot copy them or forge fake tokens. We do not assume the attacker to be able to physically remove the encryption gateways or manipulate the network infrastructure. If we granted the attacker this power, any security scheme would be moot, as he could simply remove all gateways or mount a man-in-the-middle attack between the encryption gateway and the individual end point the gateway is trying to protect. The protection of the physical infrastructure is a concern of operational security and outside the scope of such a scheme as described here. For example, the removal of a life encryption gateway could be detected by a missing heartbeat signal. A subsequent automatic alarm issued by the system could then alert personnel to handle the situation.

7.3.2 Scheme

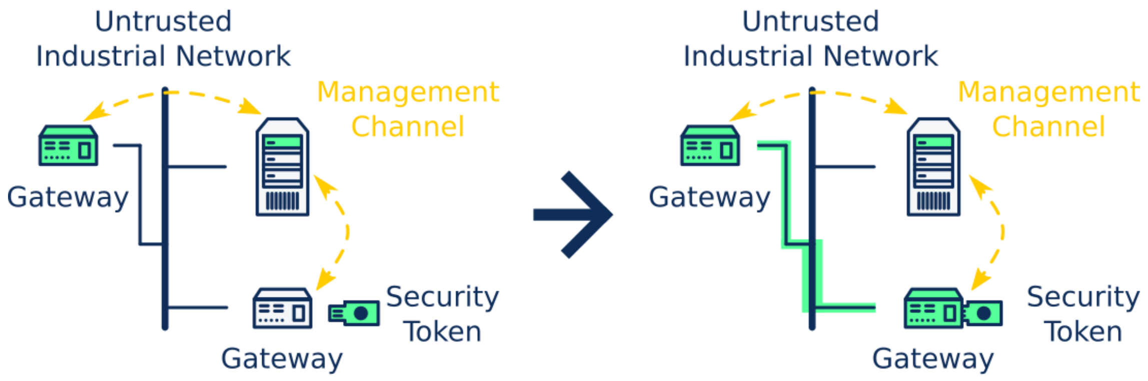
Our scheme consists of multiple steps that establish trust between the individual components involved in the system. This binds them together and as a result easy configuration of secure channels between gateways becomes possible. All steps are presented in the following, while the first three are also depicted in Fig. 7.2 for additional clarity:



(a) Step 1: Gateway provisioned by the server.



(b) Step 2: Token provisioned and bound to green secure channel.



(c) Step 3: Secure Channel Configuration on deployed gateway via security token.

Figure 7.2: Steps 1 to 3 of our configuration scheme for encryption gateways using hardware-based security tokens.

Step 1: Establishing a Trust Relationship between Server and Gateway

Before a gateway can be deployed in the network, there must be an initial phase, where network addresses as well as cryptographic keys (e. g. public keys, pre-shared keys or certificates) are exchanged between the gateway and the server, so that later mutual authentication and the establishment of a secure management channel over the network becomes possible. This exchange necessarily happens without any security as there is no secure channel established yet. This is typically done in a provisioning phase, where the gateway is directly connected to the server in an isolated environment, where it can be guaranteed that no attacker is present. This can be realized by connecting the gateway to a dedicated physical port on the server (e. g. USB or Ethernet) as shown in Fig. 7.2a, or by putting both in a separate isolated network. In more complex settings, an offline backup server can be used to onboard gateways. The exchanged credentials are then mirrored back to the live server via a single direct channel. Using some kind of external boot medium for the first boot of the gateway would also be possible. The gateway could read the credentials from there. Yet, the return channel would be cumbersome, as the credentials of the gateway could only be stored on the token and must then be transferred back to the server.

Irrespective of how it is concretely done, the gateway can then be deployed in the network and can establish a secure management channel with the server over the network.

Step 2: Establishing a Trust Relationship between Server and Token

The token must also go through a provisioning phase, where it is directly connected to the server so that secrets can be securely exchanged, making later remote authentication possible. Additionally, the token's identifier must be registered by the server and the server must bind this token identifier to some secure channel identifier as shown in Fig. 7.2b. This also makes it possible to bind multiple tokens to the same secure channel.

Step 3: Setup of a Secure Channel on a Gateway

Once trust relationships are established and the gateways are deployed, the actual secure channels (green in Fig. 7.2c) protecting the communication traffic of the end points can be configured. First, a token must be plugged into the gateway in question. After a physical user interaction, like pressing a button on the token, the token emits its token identifier and some secret necessary for authentication against the server. Both are then sent via the secure management channel to the server. The server can then validate all information. It can prove that the token is trustworthy and known and that it is connected to a trustworthy gateway. It then registers the gateway as a new participant in the secure channel bound to the token identifier. After that, it sends back all necessary information for the gateway it needs to configure the secure channel. This includes network addresses of other participants and security keys or certificates. It also updates other participants of the secure channel about this new participant. All gateways proceed to update their configurations accordingly. Finally, the newly configured gateway indicates its new state by some means to give feedback to the operator. This might be implemented by for example signal LEDs or an attached display. The gateways are now able to exchange encrypted traffic from their respective endpoints, as is depicted in Fig. 7.1.

Step 4: Tear Down of a Secure Channel on a Gateway

To tear down a secure channel, the gateway must be removed from the list of participants of that channel on the server. This can be achieved in two ways. First, the gateway can be removed by using the corresponding token again. It is plugged in and everything happens exactly as described in the previous step, only that now the gateway is deregistered from the secure channel. All remaining participants update their configurations accordingly. The gateway also removes the secure channel from its configuration and goes back to its default behavior, which for example, might be to transparently patch traffic through. The second way to remove a gateway is to just issue a command to deregister it directly on the server. Everything else then happens as described just now.

Step 5: Decommissioning of a Gateway

Should a gateway need to be removed or replaced because it broke or was successfully attacked, the server can just order other participants to stop interacting with this gateway by deregistering it from all its secure channels. Other gateways will then simply ignore this one.

Step	Token (T)	Management Server (MS)	Gateways (GW)
1.	—	GW = (pubKey$_{GW}$, IP Address$_{GW}$, MS Address$_{GW}$), privKey$_{MS}$	pubKey$_{MS}$, IP Address$_{MS}$, privKey$_{GW}$
2.	tokenID, OTP Secret	T = (tokenID, OTP Secret), Secure Channel = (secID, MS key, {T}, *)	—
3.	*(tokenID, OTP)*	Secure Channel = (secID, MS key, {T}, {GW})	secID, {MS Address$_{GW}$}, MS key$_{secID}$

Table 7.1: Exchanged information between entities within our hardware-based configuration scheme per step. Data in cursive is ephemeral and only used once.

Step 6: Decommissioning of a Token

In the same manner as in the previous step, a token can be decommissioned by removing the binding of its token identifier to the secure channel identifier. If the token is lost or is feared to have been stolen, it is also necessary to tear down the associated secure channel on all participants, as described in Step 4 in Section7.3.2. A new channel replacing the old can then be configured using a new token.

7.4 Implementation

This section discusses our prototypical implementation, and follows the same structure that was used above. Tab. 7.1 shows which entity in our scheme holds which information and in which step that data is exchanged.

We implemented the server and the gateways using Raspberry Pi 4 minicomputers running Ubuntu Linux 20.04.

The trust relationship between server and gateways as described in Section 7.3.2 in Step 1 was realized using Wireguard [43]. It is a modern, secure and easy to use Internet Protocol (IP) layer encryption protocol that is superior to older comparable solutions. Using Wireguard provides us with a state-of-the-art protocol that is agnostic towards the transport layer and the concrete implementation of the management protocol.

The secure channels being configured in Step 3 were implemented using MACsec [45]. This encryption scheme allows to protect whole Ethernet frames, which makes it agnostic or transparent to the upper layer protocols. This makes it a good choice for the heterogeneous field of industrial communication [79]. Additionally, it fits our approach, as it can be configured on a peer-to-peer basis.

Step 1: Establishing a Trust Relationship between Server and Gateway

We implemented the isolated channel by connecting the gateway directly to a dedicated network port of the server.

(a) A YubiKey FIPS security token, as was used in our prototypical implementation.

(b) A YubiHSM 2 hardware security module, as was used for secure storage of secrets on the server.

Figure 7.3: HSMs used in our hardware-based security scheme.

To setup the management channel using Wireguard, both communication partners each need to generate a public/private key pair. The public keys (pubKey) are exchanged, while the private keys (privKey) are stored on the device. Additionally, both partners exchange IP addresses, enabling them to establish the management connection later, when the gateway is deployed in the network.

The management server must also store MACsec addresses (MS address) for each gateway. These are different from the IP addresses of the management channel and are necessary for the MACsec software clients running on the gateways to be able to connect to each other. The addresses are disseminated in a later step to gateways joining a secure channel.

We implemented our own rudimentary management protocol using gRPC [5]. All configuration steps detailed here were automated using scripts written in Python or bash if not otherwise stated.

Step 2: Establishing a Trust Relationship between Server and Token

We implemented the hardware security tokens using the YubiKey FIPS[1] [18], depicted in Fig. 7.3a. These tokens can produce one-time passwords (OTPs), which are basically secret numbers that can be used once to authenticate a transaction against a remote station (the management server in our case). The YubiKey is plugged into a USB port and issues one OTP when the button at the center is pressed.

For the remote authentication to work and to establish the trust relationship described in Step 2 in Section 7.3.2, corresponding OTP secret keys (OTP secret) from which individual OTPs are derived, must be provisioned on both the token and the remote station. We used a software tool provided by the vendor for this purpose [19].

Furthermore, we use a YubiHSM 2 module [17], depicted in Fig. 7.3b, attached to the management server as secure physical storage for the OTP secret keys. Received OTPs can be checked against stored secret keys via an API. As a result, the security-critical secret keys are only stored on the employed hardware security modules and never on the server or gateways.

Additionally, each YubiKey comes with a unique serial number, which we use as the token identifier (tokenID). Furthermore, the token is bound to a certain secure channel (e. g. to green

[1]Federal Information Processing Standards (FIPS)

as depicted in Fig. 7.2b), which will be the one configured on the gateways later. Each secure channel is denoted by a secure channel identifier (secID).

Additionally, each time a new secure channel is configured, a new MACsec encryption key for that channel is also created (MS key). For our prototypical implementation, we chose to configure MACsec using simple symmetric pre-shared keys.

Step 3: Setup of a Secure Channel on a Gateway

Establishing a secure channel on the gateway works by inserting a provisioned YubiKey into a USB port of the gateway and pressing the button of the YubiKey. This prompts the YubiKey to issue one OTP, which is then sent together with the token identifier to the server via the management channel.

The server replies with all necessary configuration information for the secure channel bound to the YubiKey's token identifier. This includes the secure channel identifier (e. g. "green"), MACsec addresses of all participants in that secure channel as well as the shared encryption key. The server then updates all other participants of this secure channel with the necessary information about the appearance of another communication partner.

Finally, the gateway configures a corresponding MACsec interface on itself and sets up the necessary virtual network bridges.

Step 4: Tear Down of a Secure Channel on a Gateway

Removing gateways from a secure channel works the same way as described in the previous step, only that the server removes the gateway from the secure channel (and not adds it) and updates the other participants to that effect. The gateway then modifies its network configuration accordingly.

The scheme also allows for a direct removal of a gateway via the local configuration interface of the management server.

Step 5: Decommissioning of a Gateway

The decommissioning of a gateway was implemented directly on the server by triggering the removal of said gateway from all configured secure channels.

Step 6: Decommissioning of a Token

The decommissioning of a token was implemented directly on the server by removing the binding of the token identifier to its secure channel.

7.5 Evaluation

In the following, the design of our scheme, presented in Section 7.3.2, as well as the implementation, detailed in Section 7.4, are compared to the design goals presented in Section 7.3.1.

The first design goal was to have a trustworthy relationship between the management server and the gateways, so that the gateways could be centrally managed. We used Wireguard to set up and protect the management channel and as long as the server only reacts to management and configuration calls from the gateways from its local Wireguard interface, potential attackers cannot even address the service managing the gateways.

The second design goal was to be able to trust the hardware tokens, that are used for the configuration of the secure channels. The employed YubiKeys offer unique identifiers and can be used to establish an OTP mechanism with the management server, effectively providing protection against being forged, copied or impersonated.

The third design goal was a trustworthy configuration. Since the server only reacts to calls from the management interface that are authenticated by a proper OTP, only trustworthy modifications to the configuration are done.

The fourth design goal was to provide a solution, where the everyday use was as simple as possible. While the bootstrap of the scheme is still fairly complex, the configuration of the actual secure channels is very easy and can be accomplished by just physically plugging a token into the gateway and pressing the button once. The complexity of the bootstrapping process compares to other approaches, as establishing trust in distributed systems just demands a certain complexity that cannot be optimized any further. Secrets have to be exchanged and network configurations have to be set. Yet, this has to be done only once for each deployment of a gateway or token. This should happen fairly rarely and can be delegated to a domain expert. The configuration of secure channels on the other hand can be accomplished by even in IT security matters untrained staff.

The fifth design goal stated that the solution should allow for life-cycle management. Our design accounted for this, by including steps, where gateways and tokens could be decommissioned, even without their cooperation. Broken or compromised components cannot compromise the system as their information can be removed from the databases directly. Renewal of components is possible by decommissioning them and adding new ones. In case of a token, a new token can even be bound to the same secure channel, the previous token was bound to, so that it is not necessary to interchange the secure channel, when a token is replaced.

The last design goal stated, that while usability was the aim of this scheme, the security of the encryption gateway-based system to which our token-based scheme is only attached to, still has paramount priority.

From the software perspective, the security of our scheme hinges on the security of the management server, but that is already the case, even without our hardware-based security scheme on top. We just added some more functionality and responsibility, but as the server already manages encryption gateways, it is already considered to be a high value target and should therefore be engineered and deployed with the utmost consideration for operational security.

Modern management servers as well as the encryption gateways are in practise equipped with general purpose operating systems that support the application layer with rich functionality. The overall amount of software employed in this scenario is huge and our scheme only adds very small amounts of complexity in comparison. A fully realized software stack, that is necessary to implement such a use case, consists of 10s to 100s of millions of lines of code (LoC), while the scripts necessary to implement our scheme, and which are additionally executed by the management server and the gateways, only comprise of ~600 LoC. From the software perspective, the increase of attack surface by our scheme is negligible.

Yet, the scheme also introduces additional hardware tokens that can be apprehended by an attacker inside the factory. And while he can use them to change or delete gateways from secure channels to disturb the functioning of the factory, he cannot divert protected traffic outside the factory, as tokens alone are not sufficient to establish a secure connection. He would need a properly configured gateway as well and we assume he cannot easily steal one. Also, while the attacker can maliciously configure secure channels using a stolen token, this configuration would be registered via the regular mechanisms of our scheme. A factory operator then has the chance to rectify this action by canceling the issued command and by removing the stolen token from the system by means of its token identifier, as it is not necessary for the operator to be in possession of the physical item to do so. The token identifier is either found in the system as the issuer of the malicious command or in some (paper) file that was filled in when the token was physically issued to a factory worker, which reported its loss. From this prospective, our scheme also increases the attack surface. Yet, it is again minimal, as malicious actions can be monitored in real-time and be reverted instantly. An attacker can hence only incur temporary harm. How long this temporary phase is, depends on operational considerations within the factory and is outside of our scope.

7.6 Conclusion

In the light of increasing complexities in future smart factories, the levels of security that can actually be realized in a factory will strongly depend on the proper deployment as well as maintenance of protection mechanisms. The deployment hinges on two factors. First, factory operators, who are no domain experts for IT security, must understand the protection mechanism and the mechanism itself must be as frictionless as possible so as to not impede the productive work of the factory.

In this chapter, we made the case for a novel mechanism that allows to easily and understandably configure encryption gateways, which will be important building blocks for the security architecture of future factories. We implemented a configuration scheme that employed hardware security tokens. These tokens cannot be forged and reduce the actions necessary to implement a security policy to a simple physical action, comparable to locking a door with a key. Our results showed, that a reduction of the complexity of configuration must not necessarily be accompanied with a loss of security.

8 Towards More Fine-grained Network Separation in Factory Networks

The previous chapters of this thesis assumed a most simplistic application of the encryption gateways, where two gateways protected two single end nodes via one secure tunnel. And while it is important from a scientific standpoint to use this most basic of setups to research and design the core functionalities, in practice, real setups will be more complex.

Fig. 8.1 shows a setup where three encryption gateways have two secure tunnels configured between them and where gateways do not protect a single but a set of devices within a subnetwork structure. Gateway 1 has two tunnels configured in parallel that connect to the two other subnetworks.

This scenario is more complex and therefore realistic, as network separation can be challenging in practice, especially in legacy networks. For example, from a pure security-centric standpoint, it would make sense to protect every device in the network with its own gateway. Yet, this is in most cases not a viable option. Gateways will instead be implemented at the borders between subnetworks, that group multiple devices.

The network structure within a factory is typically defined by purely functional considerations and also cannot easily be modified for many reasons (e. g. legacy, fixed cabling installations or other physical or even organizational limitations). Furthermore, it often makes no sense to map the security domains, that are defined by the placement of the encryption gateways in the network, one-to-one onto those functional domains. For example, it might be the case, that a machine is part of a certain subnetwork only because it is physically situated in that part of the factory. In such a case, it becomes difficult to isolate subnetworks using gateways, while at the same time still allowing for all necessary functionalities to work unhindered, e. g. access to some central shared resource. Finally, also financial concerns may lead to gateways being used to protect multiple end points at the same time. For those reasons, in reality, gateways will have to support multiple security tunnels in parallel.

As a result of these considerations, it cannot be assumed that all devices behind one gateway should be allowed to talk to all other devices behind the other gateway. This assumption becomes even more important, when considering that in our use case multiple secure tunnels (meaning access to multiple subnetworks) can be configured on one gateway. This results in a gateway having to separate communication traffic from the different security domains (green and orange in Fig. 8.1) by making specific and fine-grained routing decisions based on further characteristics of the traffic data.

Therefore, in this chapter, we analyze the communication traffic of various real industrial communication protocols in this more complex scenario. We investigate potential attacks that may arise when multiple secure tunnels are configured on a single gateway and which strategies to apply to counter these attacks, while not reducing the functional range of the protected devices.

The remainder of the paper is structured as follows. Section 8.1 describes in detail our scenario and gives necessary background knowledge in the networking technologies we employed in

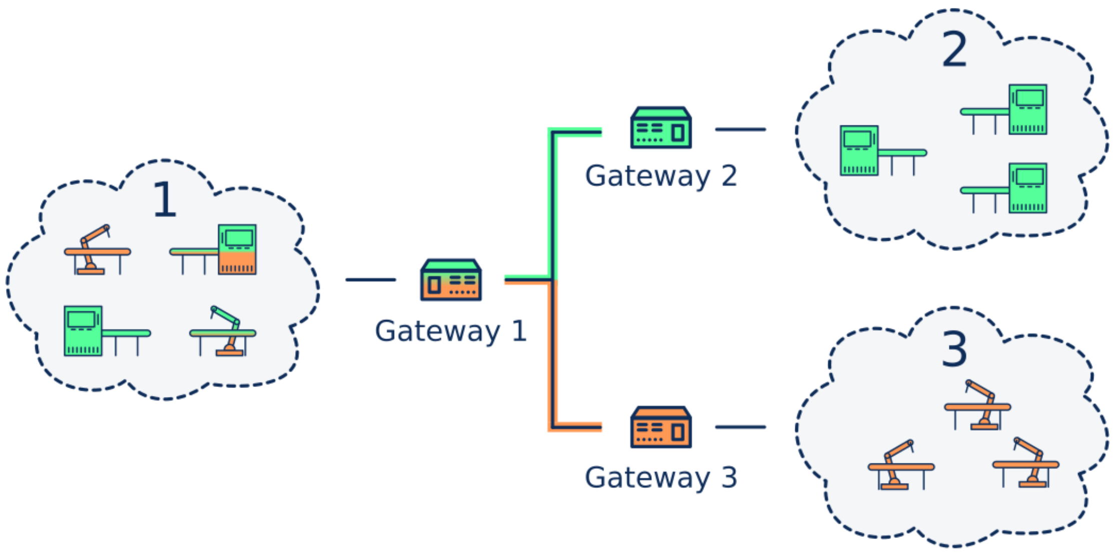


Figure 8.1: Three encryption gateways providing two secure tunnels between subnetworks. Some machines in Subnetwork 1 communicate to machines in both other subnetworks.

this work. Related work is discussed in Section 8.2. Section 8.3 describes how we implemented our test scenario. In Sections 8.4 to 8.7 we analyze and discuss various industrial protocols and give strategies on how to improve their security. Section 8.8 summarizes our findings and tries to distill some general learnings. This work is concluded in Section 8.9.

8.1 Background

Today's modern Internet Protocol (IP)-based industrial control protocols carry some technical debts from their past incarnations as fieldbuses, where they worked on top of their own custom physical and data link layers. These include rather weak security measures, as fieldbus networks used to be isolated and served specific use cases that mainly revolved around parametrizing machines and gathering sensor data from them. The switch to the general purpose IP layer networking, where suddenly all devices could talk among each other, happened mainly out of cost considerations. That change in environment was not reflected in the protocol's security architecture. As industrial networks still tended to be isolated, this was not deemed necessary.

With the advent of trends like smart manufacturing and Industry 4.0, the lack of security considerations cannot be ignored any longer. Industrial control protocols now run on top of IP networks, which span not only the factory floor but which are part of the whole company network including the office floor. Additionally, in the future more and more cloud services will open these networks up even more [125]. The assumption, that an industrial network is isolated and therefore secure, cannot be held any longer.

Specialized security overlays and tunnels will become the norm in future industrial networks where new and old equipment is run side by side and which will have many parallel connections to various cloud services [80]. Whether these will be provided by the encryption gateways that form the basis of this thesis or through the networking environment itself (via e. g. software-defined networking (SDN) etc.), will depend on the specific use case. And although this chapters scenario is based on gateways, the findings are applicable to other overlay scenarios as well.

More specifically, we will look at the routing decisions Gateway 1 in the scenario introduced above (Fig. 8.1) has to make for data packets coming from the subnetwork it protects (Subnetwork 1). The difficulty to decide stems from the fact, that it does not know in which remote subnetwork the packet's destination is located in and hence which secure tunnel to use. The gateways in this scenario are not assumed to have global knowledge about the network structure. They do not know where nodes sit, what network addresses they have or to which secure tunnel they belong to. They merely sit in between and can only use information derived from the packet headers. In a worst-case scenario, the gateway sends an incoming packet through both tunnels, effectively removing the network separation property from the tunnel setup.

Of course, in reality, this does not happen very often. In a normal networking environment, there are certain management protocols at play, that prevent useless transmissions. Yet, the goal of these protocols is to increase network efficiency. They are in no way security protocols. They assume benign network participants and are easily spoofed. A security mechanism like overlay tunnels cannot rely on them.

MAC address learning is one of those mechanisms. Networking equipment like switches learn over time to which (physical) port a certain communication device is connected to and only relay frames accordingly. This is done based on received data frames by creating a mapping of the source address of that frame with the port number it was received on. The device would then only broadcast frames (that is flood frames on all ports), when there is no mapping or if the destination is a broadcast address. In a benign setting this would achieve our goal of separating Subnetworks 2 and 3 most of the time if we assume Gateway 1 to behave in such a way. Yet, as mappings are usually deleted after some time, initial frames from new connections would get frequently broadcast. Additionally, a network device cannot in any way check the authenticity or integrity of the source address of a given frame. A malicious node can very easily spoof the source address of its frames and therefore redirect traffic of benign nodes. Also, the attacker could create random frames and overload the devices, effectively disabling MAC learning. Additionally, even with MAC learning, a participant in Subnetwork 3 can still connect to a participant in Subnetwork 2, if it knows the address. Gateway 1 would readily relay the packets. Hence, we cannot base our network separation approach on this insecure mechanism.

We translate this problem into the scenario pictured in Fig. 8.2. Two legitimate communication partners communicate via some industrial protocol behind Gateways 1 and 2, while a potential attacker sits behind Gateway 3. This attacker might be an additional malicious device that was placed there, or, as depicted, an existing device that was compromised.

The questions that we want to answer in the following are, what parts of the communication between the legitimate partners can an opportunistic attacker see although he is not supposed to, due to unintended spillover of information to the subnetwork, he is connected to. We also want to investigate, whether an active attacker can manipulate end nodes directly or indirectly in both different subnetworks and what measures can be taken against that behavior. We assume that the gateways do not know about specific characteristics of nodes in any subnetwork. This means they do not now their network addresses, which services are running on them or even their presence. They only see communication data in transit.

We will investigate multiple industrial control protocols used in practice. To be able to handle

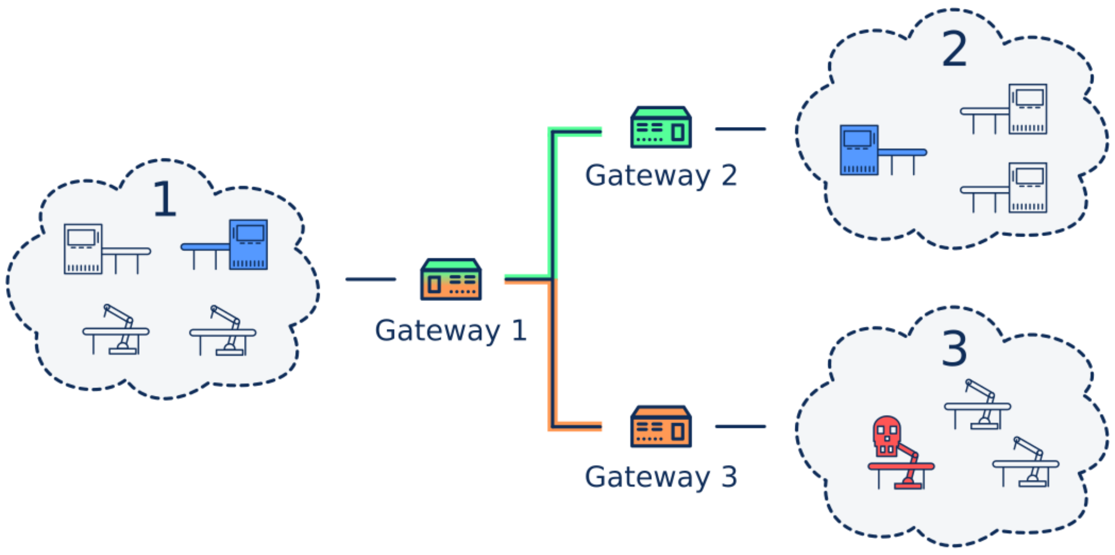


Figure 8.2: Two legitimate communication partners in different subnetworks in blue, attacker in red.

traffic data on a protocol-specific level, we will employ an SDN feature, called OpenFlow [93]. Network devices, like switches, that support OpenFlow can be configured with so-called flow rules. These make it possible to define fine-grained policies on how data frames or packets should be handled based on characteristics that can be derived from the various standard headers each datagram contains.

These characteristics include Ethernet types, source and destination addresses (Ethernet or IP), source or destination ports (of OSI[1] layer 4 transport protocols like the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP)) and many others. For example, we can define a flow rule that applies specifically to a certain protocol like HTTP and which limits the flow of these packets to a certain physical outgoing port. In this example, the device would filter for packets with TCP port 80 and selectively handle these packets differently from the default. Therefore, we can use flow rules to describe security policies, which are then enforced by network devices (our encryption gateways).

8.2 Related Work

Many studies in the recent past found many elements of industrial networks to be insecure. Pfrang et al. found many vulnerabilities in the software of industrial automation (IA) components [103], while Dahlmanns et al. found that these components tend to be configured insecurely [41].

Other studies focused on the industrial protocols themselves. A study by Drias et al. looked specifically at Modbus [104] and DNP3[2] [62]. The authors showed security-related flaws in the designs of these protocols and built a taxonomy of possible attacks from their findings [44].

[1]According to the Open Systems Interconnection model (OSI layering model)
[2]DNP3 - Distributed Network Protocol 3

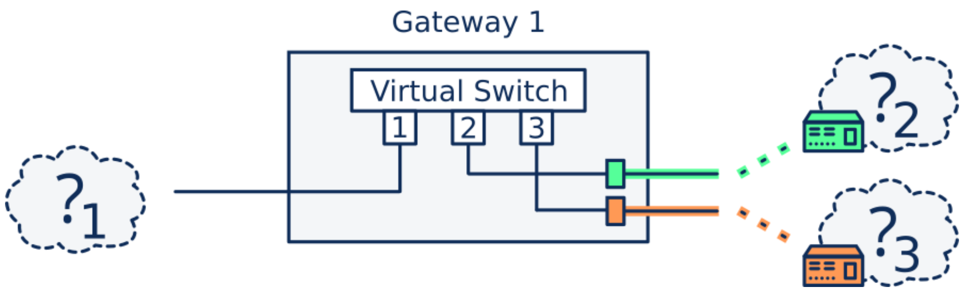


Figure 8.3: Virtual switch setup of Gateway 1 from scenario depicted in Fig. 8.1.

They specify an attacker model, which agrees with the one we also propose. Their proposal for improvement is to include security measures into said protocols. Yet, a redesign of standardized and widely rolled out protocols is very cumbersome. Standardization typically takes years and legacy installations, especially in the industrial context, will take decades to be replaced. Our approach, on the other hand, employs technologies that sit at the network level and can hence immediately improve the situation.

Two further studies by Luis et al. as well as Xu et al. analyzed various industrial protocols and also found many vulnerabilities [91, 121]. Both studies propose many different countermeasures ranging from operational improvements (regular software updates, risk assessment and testbed experiments), redesigns of the protocols (include encryption and authentication) to network level measures like firewalls, virtual private network (VPN)s, network separation and intrusion detection systems. Yet, they do not specify on how to implement those mitigations. In detail, they do not specify on how to implement network separation in more complex networks.

A study by Béla et al. on the other hand looked specifically at network segmentation strategies [53]. The authors conducted an experimental analysis of security strategies for industrial control networks, where they tested different best-practice network segmentation strategies against a sample of the real Stuxnet malware. These alone failed to contain Stuxnet and they showed that these measures must be augmented with additional more fine-grained defense-in-depth strategies compared to the state of the art, as for example described by the United States National Institute of Standards and Technology (NIST) Guide to Industrial Control Systems Security [112]. This study suggested approaches in the direction of our work, namely the application of SDN and OpenFlow technologies, to generate more fine-grained networking security policies. Yet, the authors do not present or propose specific implementation strategies.

To the knowledge of the authors, there is no study that tries to use flow management to secure and mitigate attacks on concrete industrial control protocols on the network level.

8.3 Testbed Implementation

To implement our scenario, an entity is necessary for switching the secure tunnels and for actively managing and filtering the traffic data on the gateways (particularly Gateway 1 in the scenario introduced previously). We implemented the secure tunnels using MACsec [45]. It works by encrypting Ethernet frames on OSI layer 2. It exposes the end points of its tunnels as virtual devices, so we chose to use a virtual switch to bind the virtual MACsec devices to the physical device facing the subnetwork. We used the software Open vSwitch [102], as it also provides a modern SDN primitive for managing and filtering the data flows on a protocol-specific level. This feature is called OpenFlow and was already introduced above.

Fig. 8.3 depicts the configuration of the virtual switch on Gateway 1 within our scenario. It connects the necessary ports together: the (physical) Ethernet port with the protected subnetwork as well as two virtual ports provided by the secure tunnels. Virtual switches work just like physical ones. They relay packets or rather frames on OSI layer 2. In this simple configuration, all nodes in Subnetwork 1 can talk to all nodes from 2 and 3. Yet, this also results in devices in Subnetworks 2 and 3 to be able to communicate to each other. The virtual switch does not differentiate between its attached ports and just as well forwards the traffic. Yet, this should not happen as there is no secure tunnel configured between 2 and 3.

To implement our scenario from Fig. 8.1, simple flow rules can be defined on the virtual switch on Gateway 1, which state that packets coming from switch port 2 or 3 are only relayed to port 1. This will prevent direct communication between nodes from Subnetworks 2 and 3. In the following, when discussing the individual industrial protocols in the next sections, we will always assume these simple rules to be in place.

In contrast, general rules on what to do in the opposite case, when packets are coming from Subnetwork 1, cannot be derived. Instead, additional use case and protocol-specific information is necessary. Therefore, the following sections will conduct an analysis of different industrial communication protocols. We will investigate how they behave in our scenario and which steps have to be taken to ensure network separation is achieved as much as possible considering our scenario and attacker model.

8.4 OpenPLC

We implemented our scenario using OpenPLC. It is a fully functional open source programmable logic controller (PLC) consisting of hard- and software [10]. It follows a client server model, where an OpenPLC server manages deployed OpenPLC clients, which serve as the actual PLCs. For our analysis, we installed each the server and client software on Raspberry Pi 3 minicomputers. The OpenPLC software uses two different types of communication. It uses HTTP for configuration and uploading of applications from the server to the client and it uses the fieldbus protocols Modbus and DNP3 for communication and monitoring. All communication works on top of TCP/IP.

The test setup matched the one described in Fig. 8.2, where the OpenPLC nodes were put behind Gateways 1 and 2. The attacker was put behind Gateway 3. Two scenarios are possible, depending on where to put client and server. We will investigate both.

In the first scenario, the OpenPLC server was put behind Gateway 2 and the OpenPLC client was put behind Gateway 1. OpenPLC generally works in the way that the server actively manages the clients, meaning typically the server starts the conversation. At the beginning of every TCP connection, a three-way handshake happens to synchronize both parties. In our scenario, the reply from the client (second packet from the three-way handshake) gets broadcast on Gateway 1 and hence reaches the attacker behind Gateway 3. Although, this handshake packet does not carry any payload, the attacker can already deduce at which address (IP and Ethernet) and on which ports both devices can be reached. From that he can further deduce which services are running. This is enough information to mount a targeted attack on the PLC behind Gateway 1.

In the other scenario, where the server is situated behind Gateway 1 and the PLC is behind Gateway 2, the attacker can freely communicate with the server, by just using the regular secure tunnel connecting both their subnetworks. If he manages to successfully manipulate or attack the server, he can control the client behind Gateway 2 indirectly. The past has shown that IA equipment generally must be considered vulnerable to attacks and under this light, attack surfaces should be as minimal as possible. And indeed, as a testament to its security-mindedness, OpenPLC itself does not even use HTTPS but only the unencrypted HTTP.

An effective countermeasure to this is to define flow rules on Gateway 1, that eliminate leakage of TCP traffic from and to the attacker behind Gateway 3. As both the server and the client communicate on predefined TCP ports, the rules can be specified so that only the OpenPLC-related services are blocked, while other services would still work. The following rules do just that, assuming the services run on TCP port 8080:

#	**In Port**	**EtherType**	**TCP Dst Port**	**Action**
1	3	IPv4	8080	<drop>
2	1	IPv4	8080	out port 2

Rule 1 drops any packet coming from Subnetwork 3 (the attacker) destined to reach the service port (TCP Dst port) of either client or server, while the second rule prevents possible information leakage that might occur during the handshake phase.

Additionally, OpenPLC makes it possible to customize the ports. Should these rules impede on other services running in the network, it is always possible to change the ports to a unique number and modify the rules accordingly. The result would be tailored flow rules, that minimize the impact on the network.

8.5 CODESYS

The CODESYS development system is an environment for programming controller applications and for controlling PLCs [63]. It is an entirely software-based system and is applicable for programmable hardware controllers from various vendors. It can use different industrial fieldbus protocols and consists of server and client components.

For our setup, we ran the SoftPLC software as the client on a Raspberry Pi platform and CODESYS Studio as the server on a Windows machine. All communication between the entities was by default UDP-based. The interesting scenario in this setup is when the server components sits behind Gateway 2 and the PLC sits behind Gateway 1, while the attacker is still behind Gateway 3. So we will investigate this setting further.

UDP is a connectionless protocol, where no handshakes happen in the beginning and where no messages are sent to acknowledge received packets. This leads to a certain particularity in our scenario. If Gateway 1 never learns the MAC address of the server behind Gateway 2, because the server is offline for whatever reason (or an attacker actively poisons the involved switches), all datagrams from the client would be broadcast to both Subnetworks 2 and 3 and hence also to the attacker. In the first case, the attacker would not even need to play an active role.

A special feature of CODESYS clients are periodical broadcast messages that emit a heartbeat intended for the server. Yet, these are by definition also forwarded to the attacker. In both cases, he gains insights into which devices are behind Gateway 1, which services they are running and how to address them.

Additionally, there is a mechanism of so-called network variables. These are simple values that are sent over the network via broadcast. They are used by clients to achieve highest possible interoperability with devices from other vendors. These messages would reach the attacker as well. Since they are not signed or authenticated in any way, the attacker can easily manipulate these messages to attack clients, after having learned about their existence in a previous phase. Flow rules to prevent this and to restrict the traffic to the intended subnetworks look like this:

#	In Port	EtherType	UDP Dst Ports	Action
1	3	IPv4	1700 - 1703, 1740	<drop>
2	1	IPv4	1700 - 1703, 1740	out port 2

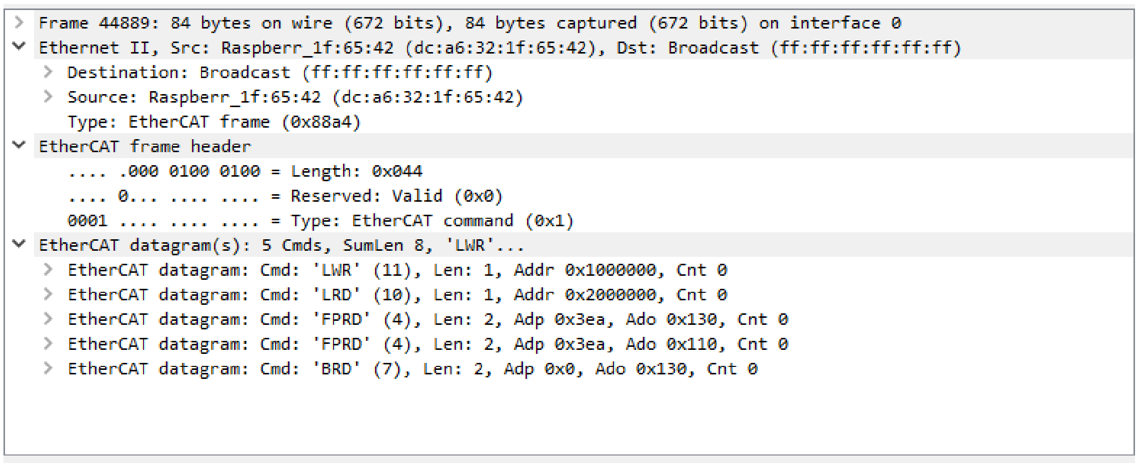
The network variables are by default configured on the UDP destination ports 1700 to 1703, while the broadcast heartbeat messages are sent on port 1740. The first rule drops any traffic that the attacker might direct at the CODESYS components. The second rule makes sure, that no CODESYS traffic reaches the attacker's subnetwork.

8.6 EtherCAT

EtherCAT is an industrial communication protocol fulfilling real-time requirements [64]. It can also be used for non real-time communication using plain TCP or UDP. It works by arranging a master node and multiple clients (slaves) in a ring topology. Only the master node can initiate communication by sending datagrams into the ring. Clients only receive, modify (by adding data to the intended fields in the datagram) and forward until the frame is ultimately returned to the master node.

Although EtherCAT today works on top of the Ethernet protocol, it does not make use of the Ethernet addressing scheme. Instead all frames are always addressed to the Ethernet broadcast address. If all devices are placed in a closed ring without external links, this is not a security problem. This makes sense for the real-time use case, as in such a scenario all connected devices would be tightly coupled anyway. They would all serve the same application and therefore also belong to the same security domain. Yet, in other scenarios, where no strict ring topology is employed, because real-time is not a requirement and only command and control communication to devices is needed, this can lead to information leaks and even make active attacks possible.

We implemented our test scenario, using two Beckhoff EK1100 EtherCAT couplers connected to Raspberry Pis. We first investigated the setup, where the master node sat behind Gateway 1 and where the client node sat behind Gateway 2. In this setting, since everything the master node sends is a broadcast, the attacker behind Gateway 3 gains full knowledge. This includes the commands sent to the clients, as well as EtherCAT addresses and memory locations of all these clients. Fig. 8.4 shows such a message.



```
> Frame 44889: 84 bytes on wire (672 bits), 84 bytes captured (672 bits) on interface 0
v Ethernet II, Src: Raspberr_1f:65:42 (dc:a6:32:1f:65:42), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
  > Destination: Broadcast (ff:ff:ff:ff:ff:ff)
  > Source: Raspberr_1f:65:42 (dc:a6:32:1f:65:42)
    Type: EtherCAT frame (0x88a4)
v EtherCAT frame header
    .... .000 0100 0100 = Length: 0x044
    .... 0... .... .... = Reserved: Valid (0x0)
    0001 .... .... .... = Type: EtherCAT command (0x1)
v EtherCAT datagram(s): 5 Cmds, SumLen 8, 'LWR'...
  > EtherCAT datagram: Cmd: 'LWR' (11), Len: 1, Addr 0x1000000, Cnt 0
  > EtherCAT datagram: Cmd: 'LRD' (10), Len: 1, Addr 0x2000000, Cnt 0
  > EtherCAT datagram: Cmd: 'FPRD' (4), Len: 2, Adp 0x3ea, Ado 0x130, Cnt 0
  > EtherCAT datagram: Cmd: 'FPRD' (4), Len: 2, Adp 0x3ea, Ado 0x110, Cnt 0
  > EtherCAT datagram: Cmd: 'BRD' (7), Len: 2, Adp 0x0, Ado 0x130, Cnt 0
```

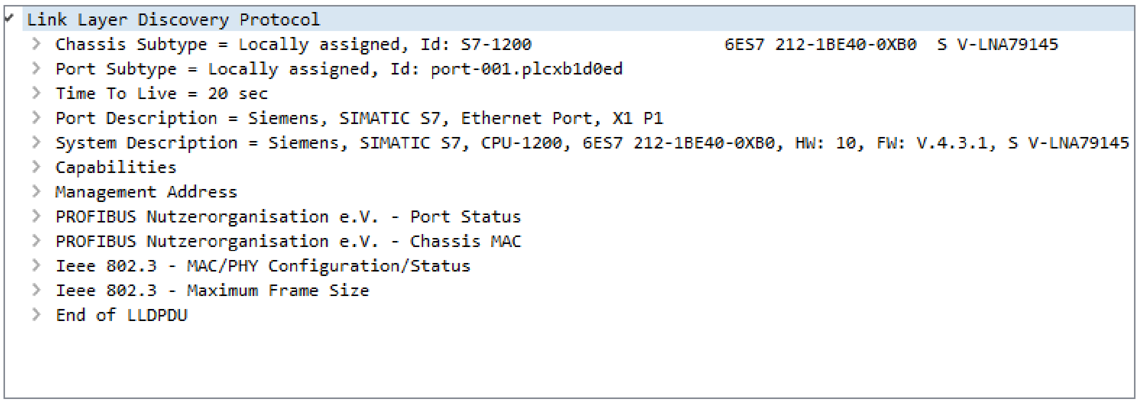
Figure 8.4: EtherCAT frame sent from a master node.

In the other setting, where the masters node sits behind Gateway 2 and the client behind Gateway 1, the attacker can learn even more. Here, the attacker does not only learn the commands sent to the clients, but also their answers as these are added to the same datagram. If the client behind Gateway 1 is the last client in this ring, then the attacker would have full knowledge of the communication. Additionally, due to the lack of authentication in this protocol, the attacker can even impersonate the master and send malicious commands to the client behind Gateway 1.

This can be prevented by not using EtherCAT in OSI layer 2 mode. The nodes can be configured to use only layer 4 transport protocols, where all messages then would be sent as unicasts. This would increase the management overhead, but also the security considerably. Yet, as the previous sections have shown, even using TCP or UDP does not give perfect security in our scenario. Since EtherCAT has an exclusive EtherType (header field indicating the encapsulated protocol) and open flow rules can filter frames based on them, it is not even necessary to switch to layer 4. The following flow rules suffice to ensure network separation in our scenario:

#	In Port	EtherType	Ports	Action
1	3	EtherCAT		<drop>
2	1	EtherCAT		out port 2

The first rule drops any EtherCAT traffic the attacker might try to inject, while the second prevents any information leakage towards the subnetwork of the attacker. These rules result in a complete isolation of EtherCAT traffic to Subnetworks 1 and 2. Not even a legitimate EtherCAT device in Subnetwork 3 can communicate with one of the other subnetworks, while other types of traffic are unaffected.



```
Link Layer Discovery Protocol
  > Chassis Subtype = Locally assigned, Id: S7-1200                6ES7 212-1BE40-0XB0  S V-LNA79145
  > Port Subtype = Locally assigned, Id: port-001.plcxb1d0ed
  > Time To Live = 20 sec
  > Port Description = Siemens, SIMATIC S7, Ethernet Port, X1 P1
  > System Description = Siemens, SIMATIC S7, CPU-1200, 6ES7 212-1BE40-0XB0, HW: 10, FW: V.4.3.1, S V-LNA79145
  > Capabilities
  > Management Address
  > PROFIBUS Nutzerorganisation e.V. - Port Status
  > PROFIBUS Nutzerorganisation e.V. - Chassis MAC
  > Ieee 802.3 - MAC/PHY Configuration/Status
  > Ieee 802.3 - Maximum Frame Size
  > End of LLDPDU
```

Figure 8.5: Periodic LLDP message sent from a Simatic device.

8.7 Siemens Simatic

Simatic denotes a whole product range of PLCs and other equipment from the vendor Siemens. We used two Simatic devices, the monitoring and control device HMI KTP400 and the PLC S7-1200 to implement our test scenario [13, 14]. These employ various protocols for different purposes, which we will discuss in the following in turn.

The Link Layer Discovery Protocol (LLDP) is an OSI layer 2 protocol used for discovering other Simatic devices in the same network. Both of the Simatic devices we employed send periodic messages informing other devices about themselves. Fig. 8.5 shows such a message. These messages contain information about ports and services running on them, capabilities of the devices, network addresses and firmware versions among other things. Both devices use LLDP in the same way, so for the purpose of analyzing this protocol, it is not important, which of the devices is connected to Gateway 1 and 2. The attacker in our scenario is still behind Gateway 3. LLDP uses a certain multicast group as destination MAC address. Since these are never used as source addresses, no MAC learning can happen on the virtual switch on Gateway 1. This means that even in the benign case, the messages are constantly broadcast to everyone, including the attacker in Subnetwork 3. These LLDP messages are very verbose and offer valuable information, which can be used for mounting a targeted attack.

As a countermeasure, LLDP could be disabled, but this would result in a loss of functionality as well. It is better to create flow rules based on the special multicast address of these messages so that they are only forwarded to the subnetworks, where cooperating devices are actually expected. Respective flow rules look like this:

#	In Port	EtherType	Ethernet Dst	Action
1	3	LLDP	01:80:c2:00:00:0e	<drop>
2	1	LLDP	01:80:c2:00:00:0e	out port 2

The first rule prevents the attacker from injecting spoofed LLDP messages, while the second prevents information leakage to the attacker in Subnetwork 3.

The ISO-on-TCP industrial Ethernet protocol (Profinet) is used by Simatic devices to communicate to each other [64]. It was created by porting the old ISO fieldbus protocol (PROFIBUS) to the IP layer, by encapsulating it into the TCP transport protocol. From a network perspective, ISO-on-TCP is a normal TCP-based protocol, hence the same observations apply as with the OpenPLC setup in Section 8.4. Flow rules to mitigate the possible attacks look similar compared to the ones for the OpenPLC scenario.

Open User Communication (OUC) is another protocol used by Simatic devices. It is used for communication with devices from other vendors. It can be configured to work on top of TCP or UDP. Simatic PLCs can be configured to send periodic values, just like with other protocols described previously. If necessary for compatibility reasons, these periodic messages can also be configured to be sent as broadcasts (e. g. to communicate with CODESYS devices). This protocol then behaves security-wise exactly the same way as other protocols using broadcasts described in the previous sections. The attacker could mount the same attacks and flow rules to mitigate these attacks can be created accordingly. Flow rules can even be more fine-grained, as OUC allows to configure custom ports.

The Simple Network Management Protocol (SNMP) is a standard protocol for gathering information about devices and managing them. It works on top of UDP and is also used by the Simatic devices we tested. Part of this protocol are so-called trap messages. These are automatically sent, when a client device changes state. SNMP comes in three versions and in versions 1 and 2, there are no measures in place to secure these trap messages in any way. Instead, a so-called community string is sent in plaintext that just specifies a domain. In our scenario, the attacker could easily get knowledge of this string (as described previously with other in this respect similar protocols) and then use it to pose himself as an SNMP server to manipulate the clients. These old versions of SNMP are still available for reasons of compatibility and should be disabled in favor of version 3, if possible. The latest version employs encryption and authentication between the SNMP server and the clients. If this is not applicable, flow rules can be constructed using the standard SNMP ports in the same manner as described above to restrict SNMP traffic to the right subnetworks.

8.8 Summary and Lessons Learned

Industrial protocols carry certain technical debts from their fieldbus pasts. Even without considering any special use case like ours, their security level is not good. For example, there are typically no encryption or authentication schemes in place that could protect data in transit. Generally, security measures are often neglected in favor of easier management and undisturbed functionality of the network. Yet, it is possible to at least increase the security, even within the means of existing industrial protocols. This must actively be pursued by the factory operators, by configuring the available security measures and by not relying on legacy modes of operation.

A general trend in industrial protocols seems to be to reduce their complexity as much as possible. Either because management overheads shall be kept to a minimum or because interoperability with other vendors shall be achieved. This then results for example in using broadcast messages. Some protocols rely heavily on them (e. g. CODESYS network variables or EtherCAT), so simply switching them off is not possible as this would limit the functionality of the protocols. Yet, as we have shown, broadcasts can have unintended consequences. And although we can deal with these issues in our scenario, vendors should strive for more sophisticated solutions, like for example standardized mediation layers between protocols.

#	In Port	<Protocol-specific Fields>	Action
1	2		out port 1
2	3		out port 1
3	3	<protocol-specific values>	<drop>
4	1	<protocol-specific values>	out port 2

Table 8.1: Universal abstract Flow rules to mitigate attacks.

We also saw that many industrial control protocols today are based on standard OSI layer 4 transport protocols. With OpenPLC, we investigated some protocols using TCP, while in the section on CODESYS, we looked at a protocol using UDP. We analyzed security-related drawbacks on both and proposed measures to improve them. The findings can readily be generalized for other industrial protocols also working with TCP and UDP. In fact, when analyzing the Siemens Simatic devices, we found that although different industrial protocols were encapsulated, the same kind of security considerations led to the same kind of mitigations.

In summary, our findings can be reduced to four universal but unspecific Flow rules, which can be found in Tab. 8.1. The first two rules are the default basic rules, which determine that no data can flow from Subnetwork 2 to 3 and vice versa. Rule 3 and 4 are protocol-specific and block any traffic flowing from and to a potential attacker. Since rule 2 and 3 cover data flowing in from the same port, it becomes important which rule has priority. Here, rule 3 has higher priority. If a datagram matches rule 3, it is dropped, if not, it is relayed. How (detailed) a certain protocol is specified, must be decided on each case separately. It may also depend on special characteristics of the respective factory network and the devices therein. Thanks to OpenFlow it does not matter, whether the protocol in question is an OSI layer 2, 3 or 4 protocol. It does not matter, whether it uses unicast, multicast or broadcast. All types can be described via flow rules.

In general, the approach we propose can always be implemented, even disregarding the specific protocol that needs to be handled. Yet, this security mechanism must be implemented on a use case-specific manner and on a different managing domain (on the network, and not on the devices directly). Ordinarily, it is bad practice to try to patch something up with auxiliary means. Yet, in this case, where the protocols cannot be easily replaced or updated, because of long standardization processes and the general longevity of legacy installations, this might be the best chance to effect some positive change in the short term.

8.9 Conclusion

In this chapter, we conducted a security analysis on how different industrial protocols behaved in an encryption gateway scenario, that is more realistic compared to the previous chapters. We found some vulnerabilities that could lead to attacks in our scenario as well as in other settings. We proposed mitigations for these vulnerabilities by employing SDN and OpenFlow rules to actively manage data flows on the gateways. Although those rules are specific to our use case, they may serve as examples and the general ideas behind them can be extended to other network separation scenarios as well. In general, our analysis showed that more fine-grained and service-specific security policies are possible and necessary to bring encryption gateways into practice.

This work also opens up many pointers for future research. More protocols could be investigated and further research should be conducted on how to implement the findings of this work in practice. Future research should focus on automation of the application of filter rules. This could be done by employing SDN functionality so that switches learn over time on themselves or by augmenting tunneling solutions with information from factory planning tools, so that factory operators could not only define secure tunnels, but also define the communication protocols allowed to pass. The system could then automatically infer appropriate rules.

9 Conclusion

Future industrial networks will consist of a mixture of old and new components, due to the very long life-cycles of industrial machines on the one hand and the need to change in the face of trends like Industry 4.0 or the industrial Internet of things (IIoT) on the other. These networks will be very heterogeneous and will serve legacy as well as new use cases in parallel. This will result in an increased demand for network security and precisely within this domain, this thesis tried to answer one specific question: how to make it possible for legacy industrial machines to run securely in those future heterogeneous industrial networks.

The need for such a solution arises from the fact, that legacy machines are very outdated and hence vulnerable systems, when assessing them from an IT security standpoint. For various reasons, they cannot be easily replaced or upgraded and with the opening up of industrial networks to the Internet, they become prime attack targets. The only way to provide security for them, is by protecting their network traffic.

The concept of encryption gateways forms the basis of our solution. These are special network devices, that are put between the legacy machine and the network. The gateways encrypt data traffic from the machine before it is put on the network and decrypt traffic coming from the network accordingly. This results in a separation of the machine from the network by virtue of only decrypting and passing through traffic from other authenticated gateways. In effect, they protect communication data in transit and shield the legacy machines from potential attackers within the rest of the network, while at the same time retaining their functionality. Additionally, through the specific placement of gateways inside the network, fine-grained security policies become possible. This approach can reduce the attack surface of the industrial network as a whole considerably.

As a concept, this idea is straight forward and not new. Yet, the devil is in the details and no solution specifically tailored to the needs of the industrial environment and its legacy components existed prior to this work.

Therefore, we present in this thesis concrete building blocks in the direction of a generally applicable encryption gateway solution that allows to securely integrate legacy industrial machinery and respects industrial requirements. This not only entails works in the direction of network security, but also includes works in the direction of guaranteeing the availability of the communication links that are protected by the gateways, works to simplify the usability of the gateways as well as the management of industrial data flows by the gateways.

In detail, we made the following contributions:

- The central component of any encryption gateway setup is the data tunneling mechanism. Previously, no solution was available that was specifically tailored to the industrial use case.

  We designed and implemented a mechanism that can transparently protect any kind of industrial data traffic and which respects certain particularities found in legacy infrastructures. We built on the existing protocol MACsec and extended it in multiple aspects to make it suitable for our use case. Additionally, we managed to generally increase the performance of the MACsec protocol by investigating the use of high performance ciphers. On resource-restricted platforms especially, we achieved throughput increases of up to 220%.

- Encryption gateways constitute a single point of failure. When they break, they take the machine attached to them offline as well. In the industrial sphere, this risk of failure is rated higher than risks caused by the type of IT security threats, encryption gateways protect against. As a consequence, the assessment is made against their application and it is precisely this assessment, that is the biggest hindrance to the widespread application of encryption gateways.

  To solve this problem, we designed and implemented a small additional networking device, the Switchbox, that can be attached to an encryption gateway and monitor its state. In case of a gateway failure, it patches network traffic around the gateway by physically removing the gateway from the critical path and in effect reestablishing the communication link of the machine to the network. As a result, although the machine is now unprotected, it can resume its work and the gateway failure did not have an impact on the productivity of the factory.

  Our approach changes the security assessment for the concept of encryption gateways considerably and makes their application viable in practice for the first time.

- Another point of contention concerning the application of encryption gateways is their usability. For them to be implemented and used in practice, they must integrate into existing workflows as frictionless as possible. Key in that respect is the way their configuration is handled. This must be as easy and as practical as possible.

  Hence, we designed and implemented a novel scheme using hardware security tokens, that reduces the configuration of the secure tunnels between gateways to one physical interaction. This technique is practical and understandable for staff not trained in the IT security domain, while at the same time not reducing the gateway's level of security in any way. Our configuration scheme helps to increase the practicability of the encryption gateway approach immensely.

- Deployment scenarios of encryption gateways in practice will be more complex, compared to the typically simplistic settings in academia. In reality, a multitude of gateways will be deployed in a factory in parallel and practical considerations will demand one gateway to provide multiple secure channels at the same time.

  In an effort to better understand realistic settings, we analyzed multiple contemporary industrial communication protocols in a more complex scenario involving multiple gateways and secure channels. We identified several potential vulnerabilities that may arise when applying encryption gateways naively and proposed countermeasures using active and fine-grained data flow management, further increasing the practical applicability of the encryption gateway approach.

This thesis took important steps towards the practical application of general purpose encryption gateways in industrial networks. It not only focussed on core network security aspects, but broadened its scope by investigating and proposing solutions for further practical problems that in the past hindered the widespread dissemination of the encryption gateway concept in real life.

Yet, open questions remain. We improved the performance of the encryption protocol that forms the basis of our tunneling mechanism, but we only did so in a best effort manner. Future research could investigate approaches to enable real-time capabilities. Active research in the direction of deterministic Ethernet networks is done by the Time-Sensitive Networking task group of the IEEE 802.1 working group. The group defines various Ethernet extensions for bridging networks. These techniques might serve as valuable starting points for this type of research.

Furthermore, our approach could be extended to integrate more security functionality than mere encryption of traffic. Further concepts that actively manage traffic, like firewalling or deep packet inspection could be explored.

Another direction of future research might be the extension of the concept to further use cases like remote access. Remote access in the industrial setting works differently compared to the more ordinary concept from the corporate information technology (IT) world. Here, for reasons of safety and security, access of remote parties to the factory must be heavily restricted. This means in practice, that connections are not allowed automatically, but must be established via a rendezvous server and a staff member that actively authorizes that connection. Then, any action the remote party takes must be monitored and registered. Research questions could revolve around, how gateways could be used as rendezvous servers, how configuration of secure channels could be organized between the factory operators and the remote entity and how the network traffic could be efficiently monitored and stored.

The Switchbox concept could also be explored further. Different use cases, like hot swapping to a backup server in case of a failure, could be implemented using our device. Other security-centric use cases could also be pursued, where the Switchbox acts as a lock to a wired network and only gives access if it was unlocked using a digital key, comparable to the ones used in our usability study.

Finally, further steps could be taken to generally push our solution more into practice. Novel usability concepts could be pursued, that use wireless devices like smartphones or concepts from the realm of augmented reality to configure and manage whole fleets of deployed gateways.

Acronyms

2FA two-factor authentication 92

AEAD authenticated encryption with associated data 48, 53, 54

AEGIS A Fast Authenticated Encryption Algorithm 53, 54, 57, 58

AES Advanced Encryption Standard 28, 29, 33–36, 40, 45, 48, 54, 57, 58, 70, 72, 73, 121

AN association number 61, 70

ARP Address Resolution Protocol 74

BSI German Federal Office for Information Security (Bundesamt für Sicherheit in der Informationstechnik) 14, 16

CAESAR competition Competition for Authenticated Encryption: Security, Applicability, and Robustness 53

CBC cipher block chaining 29, 34, 35

CFB1 cipher feedback mode 1-bit 33, 36, 42

CFB8 cipher feedback mode 8-bit 33, 36

CMAC block cipher-based message authentication code 34

CTR counter mode 35

DCS distributed control system 11–13

DES Data Encryption Standard 33

DNP3 Distributed Network Protocol 3 104, 106

DoS denial-of-service 31, 59, 65, 67, 69, 70, 73, 81

DPDK Data Plane Development Kit 63, 71–73, 75, 76

DSL digital subscriber line 83

ERP enterprise resource planning 14

FIPS Federal Information Processing Standards 97, 117, 125, *see Glossary:* Federal Information Processing Standards

FPGA field-programmable gate array 62, 76

GCM Galois counter mode 35, 61, 121

GMAC Galois message authentication code 34

GRE generic routing encapsulation 59

GUI graphical user interface 91

HMAC hash-based message authentication code 35

HMI human-machine interface 13, 14, 110

HSM hardware security module 91, 123, 125

IA industrial automation 8–12, 14–16, 19–21, 25, 79, 80, 104, 107, 118, 121, 122, *see Glossary:* industrial automation

IC integrated circuit 13

ICS industrial control system 8, 9, 11–20, 80, 89, 90, 118, 122, *see Glossary:* industrial control system

ICV integrity check value 46, 49, 50, 54, 61, 62, 69

IIoT industrial Internet of things 8, 15, 114, 118, *see Glossary:* industrial Internet of things

IoT Internet of things 23

IP Internet Protocol 17, 27, 28, 33, 46, 47, 63, 64, 66, 96, 97, 102, 104, 106, 111, 118, 121–123, *see Glossary:* Internet Protocol

IT information technology 8–18, 20, 25, 48, 80, 116, 118, 122, *see Glossary:* information technology

L2TP Layer 2 Tunneling Protocol 27, 28, 31, 40, 51–53, 59, 63, 66, 124

LAN local area network 27, 30–32, 45, 59, 64, 65

LLDP Link Layer Discovery Protocol 110

LoC lines of code 84, 99

MAC message authentication code 27–29, 32–35, 62

MD5 Message-Digest Algorithm 5 33, 34

MDC-2 Modification Detection Code 2 33

MES manufacturing execution system 14

MF more fragments 50

MKA MACsec key agreement 61, 62, 68, 71–73, 76

MTU maximum transmission unit 47–51, 53, 121

MTU master terminal unit 13

NIC network interface controller 63, 71, 75, 76

NIST United States National Institute of Standards and Technology 14, 16, 17, 105, 121

OPC UA Open Platform Communications Unified Architecture 19, 89, 90

OTP one-time password 97–99

OUC Open User Communication 111

PCBC propagating cipher block chaining 70

PLC programmable logic controller 11, 13, 86, 106, 107, 110, 111, 124

PN packet number 50, 61, 62, 67–70, 72

RC2 Rivest Cipher 2 33

RTT round-trip time 75

RTU remote terminal unit 13

SA security association 61, 62, 66, 67, 70

SC secure channel 60, 61, 66, 68

SCADA supervisory control and data acquisition 12, 13, 23, 24

SCI secure channel identifier 60–62, 65–67, 70, 73

SDN software-defined networking 80, 83, 103–106, 112, 113

SHA-1 Secure Hash Algorithm 1 29, 33, 34, 90

SHA-2 Secure Hash Algorithm 2 33, 34

SL short length 61, 62, 67

SmartNIC smart network interface controller 62, 76

SNMP Simple Network Management Protocol 111

TCB trusted computing base 83, 84, 88

TCI tag control information 61, 62, 67

TCP Transmission Control Protocol 27, 71, 86, 104, 106–109, 111, 112

UDP	User Datagram Protocol 27, 63, 74, 104, 107–109, 111, 112
UMAC	universal hashing-based message authentication code 35
VLAN	virtual local area network 48
VPN	virtual private network 24, 27, 28, 32–35, 45, 46, 48, 59, 63, 65, 74, 105, 124, 125
VXLAN	Virtual Extensible LAN 59, 63, 66, 71, 74, 76
XCBC	extended cipher block chaining 34

Glossary

A-I-C is short for the protection goal hierarchy, where availability is deemed the most important, followed by integrity protection and confidentiality. 18, 78, 80, 88

AES-GCM Advanced Encryption Standard (AES) in Galois counter mode (GCM). 35, 48, 53, 54, 57, 61

automation pyramid is a way of modeling the different information systems and networks within a factory. 12, 14, 16, 17, 121–123

bridging enables nodes in different networks to communicate to each other. Here, it specifically means to repackage layer 2 frames into packets for transmit over Internet Protocol (IP) networks. 23, 27, 28, 35, 63, 64, 116

C-I-A is short for the protection goal hierarchy, where confidentiality is deemed most important, followed by integrity protections and availability. 18, 78, 80

encryption gateway is a type of network middleboxs that is situated between different security domains. It establishes encrypted network tunnels to protect data traffic being transmitted over insecure networks. 22, 25, 27, 30, 45, 46, 58–60, 62, 64, 65, 69, 73, 74, 76–80, 86, 88–92, 100–102, 104, 112, 114, 115, 122

Ethernet is the most common networking protocol on layer 2 and is standardized as IEEE 802.3. 13, 17, 21, 22, 27–32, 36, 40, 46–49, 51, 52, 54, 60, 61, 63, 66–68, 71, 73, 75, 81–85, 94, 104–106, 108, 121, 122

factory floor is a term that comprises the first three levels of the automation pyramid. It is synonymous with the term shop floor. 14, 102, 123

Federal Information Processing Standards are standards developed and publicly announced by the NIST. 97, 117

fieldbus is an umbrella term for networking protocols designed for and used in industrial automation (IA). 11, 13, 21, 23, 83, 102, 106, 107, 111

fragmentation is the process of breaking up data transmission units into smaller parts for transport over networks that have a smaller maximum transmission unit (MTU). 47–54, 58, 76

frame is the name for the data transmission unit on layer 2. Here, this always means Ethernet frame. 27–32, 40, 44–55, 57, 59–77, 85, 86, 103–106, 108, 109, 121–123

industrial automation comprises all technological efforts to reduce human intervention in and increase the efficiency of production processes. 8, 11, 79, 104, 118, 121

industrial control system is an umbrella term for systems that control and steer all kinds of industrial processes. It comprises control devices, networking and auxiliary services. 8, 11, 80, 89, 118, 122

industrial Ethernet is and umbrella term for industrial communication protocols that are all based on standard Ethernet and which have various use case-specific extensions. 13, 46, 110

industrial Internet of things is another term to describe Industry 4.0 with more focus on distributed networking. 8, 15, 114, 118

Industry 4.0 as a term subsumes trends in the direction of augmenting industrial control systems (ICSs) with IT technologies to implement new use cases. 8, 15, 16, 80, 102, 114, 122, 123

information technology as used here, refers to technologies and systems traditionally attributed to the Internet and the corporate business world, as opposed to systems from the IA sphere. 8, 11, 48, 80, 116, 118

Internet Protocol is the most common networking protocol on layer 3. 17, 27, 46, 63, 96, 102, 118, 121

IT security comprises all efforts to protect IT systems from disclosure or corruption of data as well as disruption or manipulation of the services they provide. 9, 11, 16, 17, 19–21, 78, 79, 89, 91, 99, 100, 114, 115

layer 2 is the data link layer according to the OSI layering model. On this layer, nodes establish direct connections between each other, forming networks. Data is transferred in frames. The standard protocol on this level is Ethernet. 27–30, 45, 46, 48, 51, 59, 60, 63, 66, 105, 106, 109, 110, 121, 122

layer 3 is the network layer according to the OSI layering model. This layer connects networks. Data is transmitted in packets. The standard protocol on this level is IP. 27–30, 32, 46–48, 51–53, 63, 64, 66, 122, 123

MAC address is short for the media access control address. It is a unique identifier assigned to network devices communicating on layer 2. 31, 61, 66, 103, 107, 110

MACsec is short for Media Access Control layer security and is an encryption protocol, that encrypts whole frames on layer 2. 28, 30–32, 35, 36, 40, 42, 43, 45–54, 58–72, 74–77, 105, 114

middlebox is a networking device that manipulates network traffic in transit for various purposes. Encryption gateways are a type of middlebox. 22, 78, 80–86, 88, 121

office floor is a term that comprises the levels four and five of the automation pyramid. 14, 102

OSI layering model is short for the Open System Interconnection model and conceptualizes and layers various communication functions in computer networks. Important for this thesis are layer 2 and layer 3. 46, 47, 104, 122

packet is the name for the data transmission unit on layer 3. Here, this alsways means IP packet. 14, 24, 27, 28, 33, 59, 62, 63, 65–69, 72–75, 85, 86, 103, 104, 106, 107, 121–123

security token is a specific type of hardware security module (HSM) that is small and portable and has a small set of functionality. 89, 91, 92, 97, 100, 115

shop floor is a term that comprises the first three levels of the automation pyramid. It is synonymous with the term factory floor. 14, 121

smart factory is a future factory where the principles and trends of Industry 4.0 are implemented. 17, 21

smart manufacturing is another term to describe Industry 4.0. 8–10, 102

tunneling means the encapsulation of data of one protocol into another (e. g. frames into packets) to transmit from one network to another over a third. 25, 26, 48, 59, 61, 63–68, 71, 75, 76

weighted latency is our name for the concept of combining latency measurements for different frame sizes into one number to make results comparable. 29, 32, 55

Bibliography

[1] Data Plane Development Kit - Framework for high-speed packet processing.
https://www.dpdk.org/
Accessed: 3rd Oct. 2021.

[2] Data Plane Development Kit - Framework for high-speed packet processing, supported Hardware.
https://core.dpdk.org/supported/
Accessed: 3rd Oct. 2021.

[3] dstat - Tool for generating system resource statistics.
http://dag.wiee.rs/home-made/dstat/
Accessed: 12th Feb. 2019.

[4] Freelan - Open source software for creating secure point-to-point VPN connections between remote stations.
https://freelan.org/
Accessed: 12th Feb. 2019.

[5] gRPC - High performance, open source universal framework to implement remote procedure calls in distributed systems.
https://grpc.io/
Accessed: 3rd Mar. 2021.

[6] ip-l2tp - Linux standard tool to setup unmanaged static L2TP tunnels.
https://man7.org/linux/man-pages/man8/ip-l2tp.8.html
Accessed: 18th Dec. 2021.

[7] iperf3 - Standard tool for measuring network performance.
https://software.es.net/iperf/
Accessed: 12th Feb. 2019.

[8] JDSU T-BERD 6000 - Highly integrated field optical test solution for single application.
http://jdsu.fiberoptic.com/T-BERD_6000.htm
Accessed: 24th Jan. 2022.

[9] mpstat - Tool for generating processor-related statistics.
http://sebastien.godard.pagesperso-orange.fr/
Accessed: 12th Feb. 2019.

[10] Open PLC - Standardized open source implementation of a PLC (Programable Logic Controler).
https://www.openplcproject.com/
Accessed: 11th Mar. 2021.

[11] OpenSSH - Suite of secure networking tools based on the Secure Shell (SSH) protocol.
https://www.openssh.com/
Accessed: 12th Feb. 2019.

[12] OpenVPN - VPN software allowing to establish secure point-to-point connections between remote stations.
https://www.openvpn.net/
Accessed: 12th Feb. 2019.

[13] SIMATIC HMI KTP400 Basic panels by Siemens.
https://support.industry.siemens.com/cs/pd/379924?pdti=pi&dl=en&lc=en-DE
Accessed: 25th Jan. 2022.

[14] SIMATIC S7-1200 controllers by Siemens.
https://new.siemens.com/global/en/products/automation/systems/industrial/plc/s7-1200.html
Accessed: 25th Jan. 2022.

[15] strongSwan - Open source implementation of the IPsec protocol.
https://www.strongswan.org/
Accessed: 12th Feb. 2019.

[16] tinc - Open source protocol and VPN implementation.
https://tinc-vpn.org/
Accessed: 12th Feb. 2019.

[17] YubiHSM 2 - USB HSM to augment servers with cryptographic functionality.
https://www.yubico.com/de/product/yubihsm-2/
Accessed: 3rd Mar. 2021.

[18] YubiKey FIPS - FIPS 140-2 validated security token.
https://www.yubico.com/de/product/yubikey-fips/
Accessed: 3rd Mar. 2021.

[19] YubiKey Personalization Tool - Tool used for management and configuration of the YubiHSM 2 and YubiKey devices.
https://www.yubico.com/support/download/yubikey-personalization-tools/
Accessed: 3rd Mar. 2021.

[20] IEEE Standard for Local and metropolitan area networks - Bridges and Bridged Networks. *IEEE Std 802.1Q-2014 (Revision of IEEE Std 802.1Q-2011)*, pages 1–1832, Dec 2014.

[21] IEEE Standard for Ethernet Amendment 5: Specification and Management Parameters for Interspersing Express Traffic. *IEEE Std 802.3br-2016 (Amendment to IEEE Std 802.3-2015 as amended by IEEE Std 802.3bw-2015, IEEE Std 802.3by-2016, IEEE Std 802.3bq-2016, and IEEE Std 802.3bp-2016)*, pages 1–58, 2016.

[22] IEEE Standard for Local and metropolitan area networks - Media Access Control (MAC) Security. *IEEE Std 802.1AE-2018 (Revision of IEEE Std 802.1AE-2006)*, pages 1–239, 2018.

[23] CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness, Feb. 2019.
https://competitions.cr.yp.to/caesar.html
Accessed: 14th Dec. 2021.

[24] Electromechanical Elementary Relays. Standard IEC 61810, International Electrotechnical Commission, Geneva, CH, 2019.
https://webstore.iec.ch/publication/66148
Accessed: 18th Dec. 2021.

[25] IEEE Standard for Local and Metropolitan Area Networks–Port-Based Network Access Control. *IEEE Std 802.1X-2020 (Revision of IEEE Std 802.1X-2010 Incorporating IEEE Std 802.1Xbx-2014 and IEEE Std 802.1Xck-2018)*, pages 1–289, 2020.

[26] K. Ahmed, N. S. Nafi, J. O. Blech, M. A. Gregory, and H. Schmidt. Software defined industry automation networks. In *2017 27th International Telecommunication Networks and Applications Conference (ITNAC)*, pages 1–3, Nov 2017.

[27] A. Alshalan, S. Pisharody, and D. Huang. A Survey of Mobile VPN Technologies. *IEEE Communications Surveys Tutorials*, 18(2):1177–1196, Secondquarter 2016.

[28] A. D. Amicangioli, R. Y. Chow, and D. J. Yates. Message Redirector with Cut-through Switch. Patent EP 1 066 709 B1, Waltham, US, Sep. 2003.

[29] Time-Triggered Ethernet. Standard AS6802, SAE (Society of Automotive Engineers) International, Warrendale, USA.
https://www.sae.org/standards/content/as6802/
Accessed: 15th Dec. 2021.

[30] J.-P. Aumasson and D. J. Bernstein. SipHash: A Fast Short-Input PRF. In S. Galbraith and M. Nandi, editors, *Progress in Cryptology - INDOCRYPT 2012*, pages 489–508, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[31] D. Bhamare, M. Zolanvari, A. Erbad, R. Jain, K. M. Khan, and N. Meskin. Cybersecurity for Industrial Control Systems: A Survey. *arXiv*, abs/2002.04124, 2020.
https://arxiv.org/abs/2002.04124
Accessed: 17th Dec. 2021.

[32] D. Bienhaus, L. Jäger, R. Rieke, and C. Krauß. Gateway for Industrial Cyber-Physical Systems with Hardware-Based Trust Anchors. In I. Kotenko, C. Badica, V. Desnitsky, D. El Baz, and M. Ivanovic, editors, *Intelligent Distributed Computing XIII*, pages 521–528, Cham, Germany, 2020. Springer International Publishing.

[33] A. Bluschke, W. Büschel, P. Rietzsch, A. Senier, P. Sieber, V. Ulrich, R. Wiggers, J. Wolter, M. Hohmuth, F. Jehring, R. Kaminski, K. Klamka, S. Köpsell, A. Lackorzynski, T. Lackorzynski, and M. Matthews. fastvpn - Secure and Flexible Networking for Industry 4.0. In *Broadband Coverage in Germany; 12. ITG-Symposium*, pages 28–35, Berlin, Germany, Apr. 2018. VDE Verlag GmbH.

[34] A. Bluschke, F. Jehring, T. Lackorzynski, M. Matthews, and P. Rietzsch. Netzwerkvorrichtung zur Sicherung der Verfügbarkeit eines Netzwerkgerätes. Patent DE102018117896A1, Dresden, Germany, Jan. 2020.

[35] S. Bradner and J. McQuaid. Benchmarking Methodology for Network Interconnect Devices. RFC (Request for Comments) 2544, Mar. 1999.
https://www.rfc-editor.org/rfc/rfc2544.txt
Accessed: 28th Aug. 2019.

[36] D. Cerović, V. Del Piccolo, A. Amamou, K. Haddadou, and G. Pujolle. Fast Packet Processing: A Survey. *IEEE Communications Surveys and Tutorials*, 20(4):3645–3676, 2018.

[37] M. Cheminod, L. Durante, and A. Valenzano. Review of Security Issues in Industrial Networks. *IEEE Transactions on Industrial Informatics*, 9(1):277–293, Feb 2013.

[38] W. A. Conklin. State Based Network Isolation for Critical Infrastructure Systems Security. In *2015 48th Hawaii International Conference on System Sciences*, pages 2280–2287, 2015.

[39] I. Coonjah, P. C. Catherine, and K. M. S. Soyjaudah. Performance evaluation and analysis of layer 3 tunneling between OpenSSH and OpenVPN in a wide area network environment. In *2015 International Conference on Computing, Communication and Security (ICCCS)*, pages 1–4, Dec. 2015.

[40] B. Czybik, S. Hausmann, S. Heiss, and J. Jasperneite. Performance evaluation of MAC algorithms for real-time Ethernet communication systems. In *2013 11th IEEE International Conference on Industrial Informatics (INDIN)*, pages 676–681, Jul. 2013.

[41] M. Dahlmanns, J. Lohmöller, I. B. Fink, J. Pennekamp, K. Wehrle, and M. Henze. Easing the Conscience with OPC UA: An Internet-Wide Study on Insecure Deployments. In *Proceedings of the ACM Internet Measurement Conference*, IMC '20, pages 101–110, 2020.

[42] K. Doeornemann and A. v. Gernler. Cybergateways for Securing Critical Infrastructures. In *International ETG-Congress 2013; Symposium 1: Security in Critical Infrastructures Today*, pages 1–6, Nov. 2013.

[43] J. A. Donenfeld. WireGuard: Next Generation Kernel Network Tunnel. Technical report, 2018.
https://www.wireguard.com/papers/wireguard.pdf.
Accessed: 18th Dec. 2021.

[44] Z. Drias, A. Serhrouchni, and O. Vogel. Taxonomy of attacks on industrial control protocols. In *2015 International Conference on Protocol Engineering (ICPE) and International Conference on New Technologies of Distributed Systems (NTDS)*, pages 1–6, 2015.

[45] S. Dubroca. MACsec: Encryption for the wired LAN. In *netdev 1.1*. Red Hat, Feb. 2016.
https://legacy.netdevconf.info/1.1/proceedings/papers/MACsec-Encryption-for-the-wired-LAN.pdf.
Accessed: 18th Dec. 2021.

[46] D. Dzung, M. Naedele, T. Von Hoff, and M. Crevatin. Security for Industrial Communication Systems. *Proceedings of the IEEE*, 93(6):1152–1177, 2005.

[47] A. A. Erumban. Lifetimes of Machinery and Equipment: Evidence From Dutch Manufacturing. *Journal of the International Association for Research in Income and Wealth*, 54(2):237–268, 2008.

[48] Ethernet POWERLINK Standardization Group (EPSG) - Official body responsible for standardization.
https://www.ethernet-powerlink.org/
Accessed: 16th Dec. 2021.

[49] fast Zwanzig20. fast vpn - Netzwerkinfrastruktur für Industrie 4.0 - project website, 2016.
https://de.fast-zwanzig20.de/industrie/fast-vpn/
Accessed: 18th Dec. 2021.

[50] Federal Office for Information Security (BSI), Germany. *ICS Security Compendium, Version 1.23*, 2013.
https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/ICS/ICS-Security_kompendium_pdf.pdf?__blob=publicationFile&v=2)
Accessed: 15th Jun. 2021.

[51] S. Gallenmüller, P. Emmerich, F. Wohlfart, D. Raumer, and G. Carle. Comparison of frameworks for high-performance packet IO. In *2015 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*, pages 29–38, 2015.

[52] B. Galloway and G. P. Hancke. Introduction to Industrial Control Networks. *IEEE Communications Surveys Tutorials*, 15(2):860–880, Feb. 2013.

[53] B. Genge, F. Graur, and P. Haller. Experimental assessment of network design approaches for protecting industrial control systems. *International Journal of Critical Infrastructure Protection*, 11:24–38, 2015.

[54] A. Greenberg. The Untold Story of NotPetya, the Most Devastating Cyberattack in History. Online, Sep. 2018.
https://www.wired.com/story/notpetya-cyberattack-ukraine-russia-code-crashed-the-world/.
Accessed: 30th Jan. 2019.

[55] S. Hanks, T. Li, D. Farinacci, and P. Traina. Generic Routing Encapsulation (GRE). RFC (Request for Comments) 1701, Oct. 1994.
https://www.rfc-editor.org/rfc/rfc1701.txt
Accessed: 3rd Oct. 2021.

[56] M. Harada. Security management of factory automation. In *SICE Annual Conference 2007*, pages 2914–2917, 2007.

[57] Y. Heo, B. Kim, D. Kang, and J. Na. A design of unidirectional security gateway for enforcement reliability and security of transmission data in industrial control systems. In *2016 18th International Conference on Advanced Communication Technology (ICACT)*, pages 310–313, Jan. 2016.

[58] HMS Networks AB. Study on Industrial network market shares 2020 according to HMS Networks. Technical report, 2020.
https://www.hms-networks.com/news-and-insights/news-from-hms/2020/05/29/industrial-network-market-shares-2020-according-to-hms-networks
Accessed: 4th Nov. 2021.

[59] L. Hu, H. Li, Z. Wei, S. Dong, and Z. Zhang. Summary of Research on IT Network and Industrial Control Network Security Assessment. In *2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, pages 1203–1210, Mar. 2019.

[60] Y. Huang, Z. Zhang, and P. Zhu. The Design of an Industrial Remote Control Network Gateway Based on P2P VPN. In *2012 4th International Conference on Intelligent Human-Machine Systems and Cybernetics*, volume 2, pages 140–143, Aug. 2012.

[61] A. Humayed, J. Lin, F. Li, and B. Luo. Cyber-Physical Systems Security - A Survey. *IEEE Internet of Things Journal*, 4(6):1802–1831, 2017.

[62] Telecontrol equipment and systems - Part 6: Telecontrol protocols compatible with ISO standards and ITU-T recommendations. Standard IEC 60870-6, International Electrotechnical Commission, Geneva, CH.
https://webstore.iec.ch/publication/3756
Accessed: 25th Jan. 2022.

[63] Programmable controllers - Part 3: Programming languages. Standard IEC 61131-3, International Electrotechnical Commission, Geneva, CH.
https://webstore.iec.ch/publication/4552
Accessed: 25th Jan. 2022.

[64] Industrial communication networks - Fieldbus specification. Standard IEC 61158, International Electrotechnical Commission, Geneva, CH, Apr. 2019.
https://webstore.iec.ch/publication/59890
Accessed: 15th Dec. 2021.

[65] Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems. Standard IEC 61508, International Electrotechnical Commission, Geneva, CH.
https://webstore.iec.ch/publication/5515
Accessed: 17th Dec. 2021.

[66] Road vehicles - Controller area network (CAN). Standard ISO 11898, International Organization for Standardization, Geneva, CH, Dec. 2015.
https://www.iso.org/standard/63648.html
Accessed: 15th Dec. 2021.

[67] iTrinegy LLC. iPerf can be wrong! Online, 2020.
https://itrinegy.com/iperf-can-be-wrong/
Accessed: 15th Dec. 2021.

[68] B. Jeon and J. Na. A study of cyber security policy in industrial control system using data diodes. In *2016 18th International Conference on Advanced Communication Technology (ICACT)*, pages 314–317, Jan. 2016.

[69] M. Katsaiti and N. Sklavos. Implementation Efficiency and Alternations, on CAESAR Finalists: AEGIS Approach. In *2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech)*, pages 661–665, Aug. 2018.

[70] S. Kent and K. Seo. Security Architecture for the Internet Protocol. RFC (Request for Comments) 4301, December 2005.

[71] R. Khan, K. McLaughlin, J. Hastings, D. Laverty, and S. Sezer. Inter-Technology Bridging Gateway: A Low Cost Legacy Adaptation Approach to Secure Industrial Systems. In *2018 IEEE Power and Energy Society General Meeting (PESGM)*, pages 1–5, 2018.

[72] A. Khandelwal, I. Agrawal, M. I, S. S. Ganesh, and R. Karothia. Design and implementation of an industrial gateway: Bridging sensor networks into IoT. In *2019 3rd International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, pages 1–4, 2019.

[73] S. Khanvilkar and A. Khokhar. Virtual private networks: an overview with performance evaluation. *IEEE Communications Magazine*, 42(10):146–154, Oct. 2004.

[74] F. Kiremidjian. Serial Redundant Bypass Control Mechanism For Maintaining Network Bandwidth Management Service. Patent US 6,282,169 B1, Freemont, US, Aug. 2001.

[75] I. Kotuliak, P. Rybár, and P. Trúchly. Performance comparison of IPsec and TLS based VPN technologies. In *2011 9th International Conference on Emerging eLearning Technologies and Applications (ICETA)*, pages 217–221, Oct. 2011.

[76] I. Krejí and P. Mazouch. Measuring the Age of Machinery and Equipment in Czech Republic. In *33rd International Conference Mathematical Methods In Economics (MME 2015)*, 09 2015.

[77] S. Kumar, J. Haj-Yahya, M. Khairallah, M. A. Elmohr, and A. Chattopadhyay. A Comprehensive Performance Analysis of Hardware Implementations of CAESAR Candidates. Cryptology ePrint Archive, Report 2017/1261, 2017. https://ia.cr/2017/1261.

[78] T. Lackorzynski, G. Garten, J. S. Huster, S. Köpsell, and H. Härtig. Enabling and Optimizing MACsec for Industrial Environments (Extended Abstract). In *2020 16th IEEE International Conference on Factory Communication Systems (WFCS)*, pages 1–4, 2020.

[79] T. Lackorzynski, G. Garten, J. S. Huster, S. Köpsell, and H. Härtig. Enabling and Optimizing MACsec for Industrial Environments. *IEEE Transactions on Industrial Informatics*, 17(11):7599–7606, 2021.

[80] T. Lackorzynski, S. Köpsell, and T. Strufe. A Comparative Study on Virtual Private Networks for Future Industrial Communication Systems. In *2019 15th IEEE International Workshop on Factory Communication Systems (WFCS)*, pages 1–8, May 2019.

[81] T. Lackorzynski, M. Ostermann, S. Köpsell, and H. Härtig. A Case for Practical Configuration Management Using Hardware-based Security Tokens. In *Seventh Annual Industrial Control System Security (ICSS) Workshop*, pages 1–8, 2021.

[82] T. Lackorzynski, S. Rehms, T. Li, S. Köpsell, and H. Härtig. Secure and Efficient Tunneling of MACsec for Modern Industrial Use Cases. In *Seventh Annual Industrial Control System Security (ICSS) Workshop*, pages 1–10, 2021.

[83] T. Lackorzynski, P. Rietzsch, and S. Köpsell. Switchbox - Low-latency Fail-safe Assurance of Availability in Industrial Environments. In *2020 IEEE International Conference on Industrial Technology (ICIT)*, pages 328–333, 2020.

[84] A. Lakbabi, G. Orhanou, and S. E. Hajji. VPN IPSEC amp; SSL technology Security and management point of view. In *2012 Next Generation Networks and Services (NGNS)*, pages 202–208, Dec. 2012.

[85] J. Lau and M. Townsley. Layer Two Tunneling Protocol - Version 3 (L2TPv3). RFC (Request for Comments) 3931, Mar. 2005.
https://www.rfc-editor.org/rfc/rfc3931.txt
Accessed: 7th Jan. 2019.

[86] D. Z. Lou, J. Holler, S. Germanos, M. Hilgner, and W. Qiu. Industrial Networking Enabling IIoT Communication. Technical report, Industrial Internet Consortium, Aug. 2018.
https://www.iiconsortium.org/pdf/Industrial_Networking_Enabling_IIoT_Communication_2018_08_29.pdf
Accessed: 18th Dec. 2021.

[87] M. Mahalingam, D. Dutt, K. Duda, P. Agarwal, L. Kreeger, T. Sridhar, M. Bursell, and C. Wright. Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks. RFC (Request for Comments) 7348, Aug. 2014.
https://www.rfc-editor.org/rfc/rfc7348.txt
Accessed: 3rd Oct. 2021.

[88] A. Mahboob and J. Zubairi. Intrusion avoidance for SCADA security in industrial plants. In *2010 International Symposium on Collaborative Technologies and Systems*, pages 447–452, May 2010.

[89] H. Mao, L. Zhu, and H. Qin. A Comparative Research on SSL VPN and IPSec VPN. In *2012 8th International Conference on Wireless Communications, Networking and Mobile Computing*, pages 1–4, Sep. 2012.

[90] F. Martin-Tricot, C. Eichler, and P. Berthomé. An Enrolment Gateway for Data Security in Heterogeneous Industrial Internet of Things. In *29th IEEE International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises - WETICE 2020*, Bayonne (Virtual Conference), France, 2020.

[91] L. Martín-Liras, M. A. Prada, J. J. Fuertes, A. Morán, S. Alonso, and M. Domínguez. Comparative analysis of the security of configuration protocols for industrial control devices. *International Journal of Critical Infrastructure Protection*, 19:4–15, 2017.

[92] MB Connect Line GmbH. mbNET.rokey - Secure industrial router using a physical key for configuration, 2019.

https://mbconnectline.com/de/mbnet-rokey-2/
Accessed: 4th May 2021.

[93] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. OpenFlow: Enabling Innovation in Campus Networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74, 2008.

[94] T. Meudt, M. Pohl, and J. Metternich. Die Automatisierungspyramide - Ein Literaturüberblick. Technical report, TU Darmstadt, 2017.
https://tuprints.ulb.tu-darmstadt.de/6298/
Accesssed: 15th Jun. 2021.

[95] O. Mosnáček. Optimizing authenticated encryption algorithms. Master's thesis, Fakulta Informatiky, Masarykova Univerzita, Brno, 2017.

[96] M. Naedele. Addressing IT Security for Critical Control Systems. In *2007 40th Annual Hawaii International Conference on System Sciences (HICSS'07)*, pages 115–115, 2007.

[97] S. Narayan, K. Brooking, and S. de Vere. Network Performance Analysis of VPN Protocols: An Empirical Comparison on Different Operating Systems. In *2009 International Conference on Networks Security, Wireless Communications and Trusted Computing*, pages 645–648, Apr. 2009.

[98] Y. Nir and A. Langley. ChaCha20 and Poly1305 for IETF Protocols. RFC (Request for Comments) 7539, May 2015.
https://www.rfc-editor.org/rfc/rfc7539.txt
Accessed: 8th Jan. 2019.

[99] OPC Foundation. OPC Unified Architecture (OPC UA) - Standard communication protocol used in industrial automation.
https://opcfoundation.org/about/opc-technologies/opc-ua/
Accessed: 28th Oct. 2021.

[100] R. K. Pal. Implementation and Evaluation of Authenticated Encryption Algorithms on Java Card Platform. Master's thesis, Fakulta Informatiky, Masarykova Univerzita, Brno, 2017.

[101] B. Patel, B. Aboba, W. Dixon, G. Zorn, and S. Booth. Securing L2TP using IPsec. RFC (Request for Comments) 3193, Nov. 2001.
https://www.rfc-editor.org/rfc/rfc3193.txt
Accessed: 7th Jan. 2019.

[102] B. Pfaff, J. Pettit, T. Koponen, E. Jackson, A. Zhou, J. Rajahalme, J. Gross, A. Wang, J. Stringer, P. Shelar, et al. The design and implementation of open vswitch. In *12th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 15)*, pages 117–130, 2015.

[103] S. Pfrang, A. Borcherding, D. Meier, and J. Beyerer. Automated security testing for web applications on industrial automation and control systems. *at - Automatisierungstechnik*, 67(5):383–401, 2019.

[104] Modicon Modbus Protocol Reference Guide. Reference Guide PI-MBUS-300 Rev. J, Modicon, Inc., North Andover, USA, Jun. 1996.
https://modbus.org/docs/PI_MBUS_300.pdf
Accessed: 15th Dec. 2021.

[105] D. Quarta, M. Pogliani, M. Polino, F. Maggi, A. M. Zanchettin, and S. Zanero. An Experimental Security Analysis of an Industrial Robot Controller. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 268–286, May 2017.

[106] S. Rose, O. Borchert, S. Mitchell, and S. Connelly. Zero Trust Architecture. *NIST Special Publication 800-207*, 2020.
https://csrc.nist.gov/publications/detail/sp/800-207/final
Accessed: 17th Mar. 2021.

[107] A. Roth. Industrie 4.0 - Technologische Komponenten. In A. Roth, editor, *Einführung und Umsetzung von Industrie 4.0*, pages 47–72. Springer Gabler Verlag, Berlin Heidelberg, 2016.

[108] L. Schleupner. *Perfekt sichere Kommunikation in der Automatisierungstechnik*. PhD thesis, Fernuniversität Hagen, Hagen, Germany, Apr. 2012.

[109] C. Schwaiger and T. Sauter. A Secure Architecture for Fieldbus/Internet Gateways. In *ETFA 2001 - 8th International Conference on Emerging Technologies and Factory Automation*, pages 279–285, 2001.

[110] A. Sørhaug and A. L. Kupchik. Full-Duplex Medium Tap Apparatus and System. Patent US 6,424,627 B1, Merrimack, US, July 2002.

[111] M. Stevens, E. Bursztein, P. Karpman, A. Albertini, and Y. Markov. The First Collision for Full SHA-1. In J. Katz and H. Shacham, editors, *Advances in Cryptology – CRYPTO 2017*, pages 570–596, Cham, Germany, 2017. Springer International Publishing.

[112] K. Stouffer, S. Lightman, V. Pillitteri, M. Abrams, and A. Hahn. Guide to Industrial Control Systems (ICS) Security. Technical report, NIST, May 2015.
https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-82r2.pdf
Accessed: 15th Jun. 2021.

[113] Z. Sufleta. Bypass Switch for Redundant In-line Network Switch Appliance. Patent US 2017/0214566 A1, Santa Clara, US, July 2017.

[114] Z. Sufleta and H. Nguyen. Multi-path Arrangement of Redundant Inline-bypass Switches. Patent US 2018/0123831 A1, Santa Clara, US, May 2018.

[115] A. V. Uskov. Information Security of IPsec-based Mobile VPN: Authentication and Encryption Algorithms Performance. In *2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications*, pages 1042–1048, Jun. 2012.

[116] S. Vitturi, C. Zunino, and T. Sauter. Industrial Communication Systems and Their Future Challenges: Next-Generation Ethernet, IIoT, and 5G. *Proceedings of the IEEE*, 107(6):944–961, 2019.

[117] Wibu-Systems GmbH. CodeMeter - Business solution including hardware tokens for protection of intellectual property, 2004.
https://www.wibu.com/products/codemeter.html
Accessed: 3rd Nov. 2021.

[118] A. K. Wright, J. A. Kinast, and J. McCarty. Low-Latency Cryptographic Protection for SCADA Communications. In M. Jakobsson, M. Yung, and J. Zhou, editors, *Applied Cryptography and Network Security*, pages 263–277, Berlin, Heidelberg, 2004. Springer.

[119] H. Wu and T. Huang. The Authenticated Cipher MORUS (v2). Technical report, Sep. 2016.
https://competitions.cr.yp.to/round3/morusv2.pdf
Accessed: 16th Dec. 2021.

[120] H. Wu and B. Preneel. AEGIS: A Fast Authenticated Encryption Algorithm. In T. Lange, K. Lauter, and P. Lisoněk, editors, *Selected Areas in Cryptography – SAC 2013*, pages 185–201, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[121] Y. Xu, Y. Yang, T. Li, J. Ju, and Q. Wang. Review on cyber vulnerabilities of communication protocols in industrial control systems. In *2017 IEEE Conference on Energy Internet and Energy System Integration (EI2)*, pages 1–6, 2017.

[122] T. Yun, J. Luo, B. Peng, and D. Yao. Dynamic Defense Methods for Endogenously Secure Industrial Control Networks. In *2018 Chinese Automation Congress (CAC)*, pages 635–639, Nov. 2018.

[123] M. H. M. Zaharuddin, R. A. Rahman, and M. Kassim. Technical comparison analysis of encryption algorithm on site-to-site IPSec VPN. In *2010 International Conference on Computer Applications and Industrial Electronics*, pages 641–645, Dec. 2010.

[124] Z. Zhipeng, S. Chandel, S. Jingyao, Y. Shilin, Y. Yunnan, and Z. Jingji. VPN: a Boon or Trap? : A Comparative Study of MPLS, IPSec, and SSL Virtual Private Networks. In *2018 Second International Conference on Computing Methodologies and Communication (ICCMC)*, pages 510–515, Feb. 2018.

[125] ZVEI e. V. - Zentralverband Elektrotechnik- und Elektronikindustrie e. V. *The Reference Architectural Model Industrie 4.0 (RAMI 4.0)*, April 2015.
https://www.zvei.org/fileadmin/user_upload/Themen/Industrie_4.0/Das_Referenzarchitekturmodell_RAMI_4.0_und_die_Industrie_4.0-Komponente/pdf/ZVEI-Industrie-40-RAMI-40-English.pdf
Accessed: 17th Dec. 2016.