

# Secure and Efficient Protocols for the IIoT Adapting PTP and TLS to meet IIoT constraints and requirements.

**Tange, Koen Pieter**

*Publication date:*
2022

*Document Version*
Publisher's PDF, also known as Version of record

Ph.D. Thesis
Doctor of Philosophy

DTU Compute
Department of Applied Mathematics and Computer Science

# Secure and Efficient Protocols for the IIoT

## Adapting PTP and TLS to meet IIoT constraints and requirements

Koen Tange

Kongens Lyngby 2021

**DTU Compute**
**Department of Applied Mathematics and Computer Science**
**Technical University of Denmark**

Richard Petersens Plads, Building 324
2800 Kongens Lyngby, Denmark
Phone +45 4525 3031
compute@compute.dtu.dk
www.compute.dtu.dk

# Summary (English)

This dissertation is a collection of publications that aims to address several security challenges in the Internet of Things (IoT), Industrial Internet of Things (IIoT), and Fog sphere.

More and more often, industrial environments look at Cloud-like solutions for industrial applications. Modern developments such as machine-learning, Cloud services and so on, promise attractive improvements to industrial systems. Examples of this include smart (predictive) maintenance, flexibility in production chain targets, and data analytics features. However, latency, compatibility, data privacy, and reliability issues prevent the direct application of Cloud computing to many industrial scenarios. The Fog computing paradigm is a relatively new computing paradigm for the IIoT that aims to fill this gap between the Cloud and Edge, by providing Cloud-like capabilities closer to the Edge in a decentralized fashion. With this new paradigm, new security challenges arise.

In this thesis, we first briefly describe the concept of Fog computing and its relation to the Cloud, Edge, IoT, and IIoT. Then, a collection of IIoT/Fog security requirements and challenges is presented, based on a large-scale systematic literature analysis. These requirements form the motivation for the other two major contributions collected in this thesis, both of which focus on solving concrete problems that could prevent the use of specific protocols in Fog environments. Firstly, we propose a redundant Precision Time Protocol (PTP) node architecture that increases PTP security, resilience, and availability by protecting against compromised subsystems. Since Fog nodes will need to support PTP for a myriad of applications, this helps boost the security of Fog nodes as well. Secondly, we explore the feasibility of using Transport Layer Security (TLS) in IIoT environments, and introduce ratchet TLS (rTLS), a session resumption extension for the TLS protocol that allows fast session resumption to be safely used in IIoT scenarios. The rTLS extension additionally lowers TLS resource requirements, turning the protocol into an option for lightweight devices connected to e.g. Fog nodes.

# Summary (Danish)

Denne afhandling er en serie af publikationer der sigter mod at adressere flere sikkerhedsmæssige udfordringer IoT og Fog området.

Oftere er industrielle miljøer begyndt at kigge på cloud lignende løsninger for industrielle applikationer. Moderne udvikling som machine learning, cloud services og lignende, lover attraktive forbedringer for industrielle systemer. Eksempler på dette inkluderer smart (forudsigend) maintenane, fleksibilitet i produktionsleds kriterier, og data analytiske funktioner. Hvorimod forsinkelse, kompabilitet, databeskyttelse og pålidelighedsproblemer forhindrer en direkte applikation af cloud computing til mange industrielle scenarier. Fog computing paradigmet er et relativ ny computing paradigme for IoT der sigter mod at fylde dette behov mellem cloud og edge, ved at muliggøre cloud lignende muligheder tættere på grænsen i en dcentraliseret maner. Nye sikkerhedsmæssige udfordringer opstår ved dette nye paradigm

I denne afhandling vil konceptet bag fog computing og dets relation til Cloud, Edge og IoT blive forklaret indledende. Herefter vil en samling af IoT/ fog sikkerhed forudsætninger og udfordringer blive præsenteret baseret på en large scale systematisk litteratur analyse. Disse forudsætninger udgør motivationen for de andre to bidrag, samlet i denne afhandling. Begge bidrag fokuserer på at løse et konkret problem, der kan forhindre brugen af specifikke protokoller i fog miljøet.
Første foreslag er en redundant PTP node arkitektur der forøger PTP sikkerhed, modstandsdygtighed og tilgængelighed ved at beskytte imod kompromimering af under systemer. Da Fog noder er nødvendige for en samling af utallige applikationer, bidrager dette med med at forbedre sikkerheden. Sekundært vil gennemførligheden for brug af TLS IoT miljøer undersøges, og en session resumption fo ngelse til TLS protokollen, der tillader fast session resumption til sikker brug i IoT scenarier introduceres. rTLS forlængelsen formindsker samtidig ressource kravet, og gør derved protokollen som en light weigth løsning for enheder forbundet til f eks. Fog nodes.

# Preface

This PhD thesis was prepared at the department of Applied Mathematics and Computer Science at the Technical University of Denmark. The PhD was supervised by Professor Nicola Dragoni and Associate Professor Xenofon Fafoutis, and conducted in the period October 2018 to December 2021.

Kongens Lyngby, December 31, 2021

Koen Tange

# Acknowledgements

This work would not have been possible without the guidance and support of my supervisors, Nicola Dragoni and Xenofon Fafoutis, who were always ready to provide valuable input and suggestions to nudge my PhD in the right direction. Further, my thanks go out to my office mates Michele and Niklas, who provided guidance, laughter, collaborations, and were excellent sparring partners for the many discussions on research ideas and views. I'd also like to thank the FORA team and students for enabling this journey, organizing workshops and courses, and being flexible when COVID reared its head, messing up research plans for my PhD. And, of course, I'd like to thank everyone at the ESE section, and the administrative staff in particular for always being there to help me! Moving beyond DTU, I would like to thank Stefano Pepe for helping me get around in San Francisco, keeping an eye out for collaboration opportunities, connecting me with Itron and getting the ball rolling on rTLS. Further, my thanks go out to Travis Shanahan and David Howard for their support and participation in the rTLS project.

Finally, I'd like to thank my family and friends for their unyielding support. Special thanks to Hoff for translating the summary to Danish, and to Roosa and Khaled for providing valuable input on sentence structure and vocabulary usage, sharp feedback keeping my tendency to write overly long sentences in check, as well as helping me maintain a consistent and balanced use of academic language throughout the thesis without resorting to enigmatic, esoteric, and possibly archaic terminology when its use would be supererogatory, and keeping facile descriptions at a minimum.

Thanks!

# List of Publications

This thesis summarizes 6 published papers, as well as 3 MSc. thesis projects which are relevant to the research contributions made in this thesis. In this section, we list the included works first, after which we list other publications made during the PhD.

## Included Publications

The following publications are included in this thesis. In the rest of this thesis, they are referred to by the identifiers given in this list.

**Paper A:** K. Tange, M. De Donno, X. Fafoutis, and N. Dragoni. "Towards a Systematic Survey of Industrial IoT Security Requirements: Research Method and Quantitative Analysis." In: *Proceedings of the Workshop on Fog Computing and the IoT*. IoT-Fog '19. ACM, 2019. DOI: 10.1145/3313150.3313228. [28]

**Paper B:** K. Tange, M. De Donno, X. Fafoutis, and N. Dragoni. "A Systematic Survey of Industrial Internet of Things Security: Requirements and Fog Computing Opportunities." In: *IEEE Communications Surveys Tutorials* (2020). DOI: 10.1109/COMST.2020.3011208. [27]

**Paper C:** E. Kyriakakis, K. Tange, N. Reusch, E. O. Zaballa, X. Fafoutis, M. Schoeberl, and N Dragoni. "Fault-tolerant Clock Synchronization using Precise Time Protocol Multi-Domain Aggregation." In: *2021 IEEE 24th International Symposium on Real-Time Distributed Computing (ISORC)*. IEEE, 2021. DOI: 10.1109/ISORC52013.2021.00025. [14]

**Paper D:** M. Barzegaran, N. Desai, J. Qian, K. Tange, B. Zarrin, P. Pop, and J. Kuusela. "Fogification of electric drives: An industrial use case." In: *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 2020. DOI: 10.1109/ETFA46521.2020.9212010. [1]

**Paper E:** K. Tange, D. Howard, T. Shanahan, S. Pepe, X. Fafoutis, and N. Dragoni. "rTLS: Lightweight TLS Session Resumption for Constrained IoT Devices." English. In: *Proceedings of the 22nd International Conference on Information and Communications Security*. Springer, 2020. DOI: 10.1007/978-3-030-61078-4_14. [29]

**Paper F** K. Tange, S. A Mödersheim, A Lalos, X. Fafoutis, and N. Dragoni. "rTLS: Secure and Efficient TLS Session Resumption for the Internet of Things." In: *Sensors* (2021). DOI: 10.3390/s21196524. [30]

Each of these publications is appended at the end of the thesis, and when reading it digitally can be navigated to whenever referenced in the textby clicking on their labels. These works are included with permission from the publishers.

### Supervised thesis projects

Below, we list MSc. thesis projects that describe research tasks supporting the research conducted in this PhD. Note that these are original works by their respective authors, and my role was supervisory. In the rest of this thesis, these works are referred to by the identifiers given in this list.

**MSc. Thesis 1:** A. K. Mathiasen and E. Bejder. "TLS Extension Performance Impact." 2021. URL: https://findit.dtu.dk/en/catalog/2692890744 (visited on December 24, 2021).[17]

**MSc. Thesis 2:** J. Pecl. "A Practical Study on Online Tracking Using TLS Session Resumption." 2021. URL: https://findit.dtu.dk/en/catalog/2691679817 (visited on December 24, 2021).[19]

**MSc. Thesis 3:** C. Xenofontos. "rTLS: Proof-of-Concept and Empirical Evaluation." 2021. URL: https://findit.dtu.dk/en/catalog/2692305257 (visited on December 25, 2021).[34]

## Other Publications

The following works were published during the PhD but are not summarized in this thesis:

- M. De Donno, K. Tange, and N. Dragoni. "Foundations and Evolution of Modern Computing Paradigms: Cloud, IoT, Edge, and Fog." In: *IEEE Access* (2019). DOI: 10.1109/ACCESS.2019.2947652. [5]

# Acronyms

**0-RTT** 0 Round-Trip Time

**AADL** Architecture Analysis & Design Language

**AC** Access Control

**BMCA** Best Master Clock Algorithm

**CA** Certificate Authority

**CPS** Cyber-Physical System

**DH** Difie-Hellman

**DoS** Denial of Service

**DTLS** Datagram TLS

**FCP** Fog Computing Platform

**HMI** Human-Machine Interface

**IANA** Internet Assigned Numbers Authority

**IIoT** Industrial Internet of Things

**IoT** Internet of Things

**IT** Information Technology

**KDF** Key Derivation Function

**OFMC** Open Source Fixed-Point Model Checker

**OT** Operational Technology

**PKI** Public Key Infrastructure

**PLC** Programmable Logic Controller

**PSK** Pre-Shared Key

**PTP** Precision Time Protocol

**RTC** Real-Time Clock

**rTLS** ratchet TLS

**RTT** Round-Trip Time

**SDN** Software-Defined Networking

**SSL** Secure Socket Layer

**TLS** Transport Layer Security

**TSN** Time-Sensitive Networking

# Contents

# CHAPTER 1

# Introduction

Cloud infrastructure and the Internet of Things (IoT) have seen great commercial success with smart devices being adopted rapidly across the globe. The industrial sector has observed this development with interest, as smart industrial appliances could provide significant benefits to factory floors: concepts such as data analytics, over-the-air updates, smart maintenance, and flexible production chains promise higher productivity with lower costs. This movement is often referred to as the Industrial Internet of Things (IIoT) or Industry 4.0.

Unfortunately, in many industrial scenarios, Cloud capabilities cannot be used directly. Often, there are availability, safety, and latency requirements for Operational Technology (OT) used on factory floors that cannot be met with existing Cloud infrastructure. To remedy this, Fog computing has emerged as a novel paradigm that operates anywhere in the Cloud-Edge spectrum and essentially decentralizes Cloud resources into "Fog nodes", capable systems with powerful virtualization capabilities. Fog nodes are not considered a replacement for Cloud infrastructure, but rather an extension that moves those tasks that cannot be executed in Cloud environments close to the Edge. At the same time, Fog nodes can still offload highly resource intensive tasks such as data analytics to the Cloud. Since this development will directly and indirectly connect critical systems to the outside world, new security solutions are needed that can protect Fog nodes and industrial systems without violating performance, safety, and reliability requirements.

Due to its novelty, it has taken some time for the scientific community to agree on a definition of Fog computing. Throughout this dissertation the definition of Fog computing as given by the OpenFog Consortium is used [3]: "a system-level horizontal architecture that distributes resources and services of computing, storage, control, and networking anywhere along the continuum from Cloud to Things, therey accelerating the velocity of decision making". This manifests as a Fog layer between the Cloud and the IoT or IIoT, as can be seen in the example given in Figure 1.1. For a more in-depth treatment of this definition, as well as definitions of Cloud and Edge computing, interested readers are invited to read our earlier work [5], which collects common definitions and describes the evolution of these paradigms.

Because the scope of Fog systems extends from the Cloud all the way down to the Edge and spans many different use cases and platforms (often relying on application- or platform-specific protocols), it is challenging to identify security requirements applicable to the Fog at large. Additionally, Fog nodes will have to interact with a large

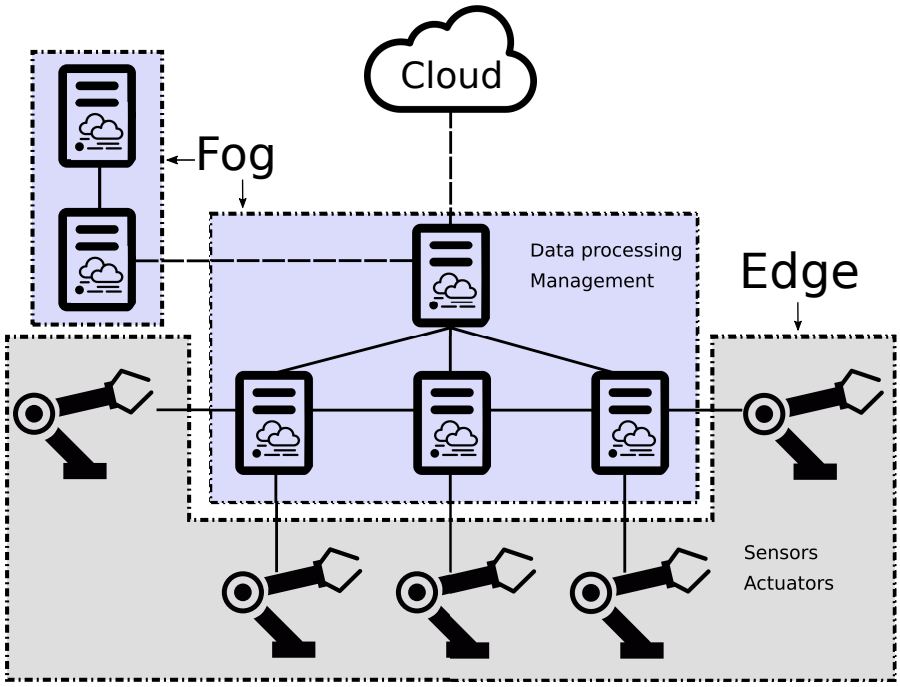



**Figure 1.1:** An example of a "Fogified" factory floor, where sensors and actuators are managed by Fog nodes. The Fog nodes cooperate with the Cloud, with each other, and with other Fog networks in a decentralized fashion.

variety of IIoT appliances with highly heterogeneous resource capabilities, turning the development of widely applicable security solutions into a very complex challenge. After a decade filled with IoT security incidents, one thing is abundantly clear: strong security is an absolute necessity if we are to entrust the Fog with managing critical industrial systems.

## 1.1 Motivation

Fog Computing and the IIoT are intimately connected paradigms, and this is reflected in the security landscape: both face security challenges for a vast heterogeneity in used platforms and protocols. Yet, a coherent view of the security requirements in these fields is lacking. In order to protect the smart industrial devices of the future, it is necessary to first identify and understand the involved security challenges. Additionally, this allows us to explore the potential Fog computing has in addressing these challenges and satisfying the corresponding security requirements.

There already exist plenty of well established application-specific protocols that enjoy wide use in the industry, but are not designed for operation in potentially hostile networks. It is inevitable that in the transition process to Industry 4.0, Fog nodes need to be compatible with these protocols. It thus makes sense to look at hardening these protocols and bringing them up to security standards for a connected industrial environment. This not only benefits Fog computing, but also the IIoT field in general. A similar argument holds for existing security protocols: if they can be adapted to work within constraints posed by IIoT environments, then they can be applied to Fog computing and the IIoT to provide security features as well as interopability with other areas where these protocols are commonly supported (e.g. the Cloud). Moreover, it is logical to first try to adapt these protocols before reinventing the wheel with completely new protocols, allowing us to rely on decades worth of expierence and analysis for these protocols.

Based on these arguments, a number of research goals were specified for the work presented in this thesis.

## 1.2 Research goals

This thesis explores security needs within the context of Fog computing and the IIoT. As part of this exploration, we look at several protocols used in these fields. Although the protocols can be considered as topics by themselves, they are in this thesis combined under the common theme of security for Fog computing and the IIoT; a popular saying in information security circles is "A chain is only as strong as its weakest link", an elegant reference to the necessity for a holistic view when security is concerned. With this in mind, we have chosen to consider the needs of a Fog system from multiple angles, and focused on improving safety, privacy, and security aspects of popular protocols addressing those needs. While by no means a complete security solution, the purpose of this research is mainly to find ways to adapt existing technologies to work well with next-generation computing paradigms.

We can more formally describe our research goals as follows:

**G-I** Catalogue security and privacy requirements of the emerging Fog and IIoT paradigms as they are identified in scientific literature;

**G-II** Identify several protocols with the potential to address common security requirements, and in addition find points of improvement for these protocols;

**G-III** Study potential protocol improvements that can help our chosen protocols to make it easier to create secure Fog and IIoT systems, be it by increasing safety, security, privacy, or performance on low-end platforms.

## 1.3 Organization

We begin this dissertation with a general motivation for the chosen research topics. Additionally, the introduction includes a brief description of Fog computing and the need for specialized security solutions for Fog systems. After that, the research contributions themselves are presented throughout the remainder of this dissertation. Each chapter presents contributions concerning a specific theme, which in turn is connected to one of the research goals. We elaborate upon this connection in the introduction to each chapter. Chapters are further subdivided into sections, with one section per contribution (with the exception of Chapter 2, which merges a preliminary survey with its full version into one section).

Contributions in this thesis are either published articles or MSc. thesis projects. In both cases we maintain the same structure per section: the start of the section describes my contribution to the work, after which a summary of the work itself follows, concluded by a discussion that puts the work in perspective to the chapter, and thereby the thesis as a whole. The summaries are written to be self-contained but cannot cover every detail. Interested readers are invited to also read the corresponding works for a complete view of their contribution. Published articles are included in full at the end of this dissertation, and MSc. theses can be retrieved online through DTU Findit[1]. Additionally, a direct link to the theses is included in in the bibliography.

| Chapter | Goals | Contributions |
|---|---|---|
| 2 | G-I | Systematic Literature analysis, IIoT security requirements, Fog computing opportunities |
| 3 | G-II, G-III | Distributed PTP node Architecture |
| 4 | G-II | Use case study, TLS Extension performance impact study, TLS 1.3 fast session resumption privacy impact study |
| 5 | G-III | TLS 1.3 fast session resumption extension (rTLS), Formal Security analysis of rTLS, empirical evaluation of rTLS |

**Table 1.1:** An overview of the contributions in each chapter and the corresponding research goals.

The contributions are separated into four chapters. Chapter 2 summarizes a systematic literature analysis and presents a collection of identified security requirements for the IIoT in the scientific literature. A number of these security requirements are used as motivators for the research presented in the remaining chapters. Next, a distributed PTP node architecture is presented in Chapter 3. This architecture enables PTP systems to operate in the presence of compromised subsystems. This builds on the requirements identified in the preceding chapter, and marks the first concrete

[1] https://findit.dtu.dk/

protocol improvement presented in this thesis. Afterwards, in Chapter 4, we summarize an investigation into the applicability of the TLS protocol to the IIoT. This starts with a use case study of a fogified electric drive, which highlights the opportunities for a protocol such as TLS. Next, the performance impact of TLS extensions is investigated, as well as the privacy implications of using its fast session resumption protocol. This investigation verifies that TLS session resumption is compatible with several privacy-oriented security requirements identified in Chapter 2. The final major contribution is detailed in Chapter 5 and encompasses a protocol extension that adapts TLS session resumption to be suitable for the IIoT. This includes the definition of the extension itself, a formal security analysis of the extension, and an empirical evaluation of its performance.

For convenience, Table 1.1 summarizes the contributions of each chapter, and provides a mapping to the corresponding research goals.

# CHAPTER 2
# Literature Analysis

To identify those areas in which research contributions can make the most impact, a good overview of the Fog and IIoT security landscape was needed. Unfortunately, no survey covering security requirements for these fields existed at the time. To fill this gap, we conducted a thorough systematic survey, cataloguing requirements identified in the scientific literature, and touching upon many security and privacy aspects. The research conducted for this survey directly addresses research goal G-I, and provides a basis that motivates subsequent works in this dissertation.

## 2.1 [Papers A, B] a Systematic Survey of Industrial IoT Security: Requirements and Fog Computing Opportunities

This survey has been published two parts. The first part, paper A, covers works in the years 2012 to 2018. Its focus lies on numerical data extracted from the investigated works, and on the methodology for the survey itself. The second part, paper B, can be considered the "full" survey and is a significant extension to paper A. The year 2019 was included in the survey scope (translating to an exponential increase in considered works), and each included work has been analyzed thoroughly. This resulted in not just a numerical overview of research trends, but also a qualitative analysis and categorization of Fog and IIoT security requirements as they were identified by the academic community in this period.

Since paper B is a complete extension of paper A, including a more complete analysis of the trends first observed in A, we consider it redundant to separately summarize paper A, and will discuss only paper B in this chapter.

### Personal Contribution

As the first author of this work, I was responsible for the majority of the content and writing of the paper, as well as the research process. I defined the search queries, collected all found publications, and (together with the second author, as this required two independent reviews per article) filtered down the results until we arrived at our

final selection of included papers. I then read each selected work, identifying security requirements. I collected these, grouped them into common themes, and categorized them in the many tables that form the basis of the analysis in this work.

### 2.1.1 Extended Summary

This systematic survey aims to answer the following set of research questions:

**RQ1**: What are the security requirements of the IIoT?

**RQ2**: How are publications related to IIoT security spread throughout the years?

**RQ3**: How is IIoT Security research activity geographically distributed?

**RQ4**: What are the most popular publication venues for IIoT Security research?

In addition, after an analysis of the investigated works, we discuss various opportunities for Fog computing in addressing the identified security requirements.

The survey follows the research methods for conducting a systematic survey in software engineering proposed by Petersen et al. [20], and covers the period 2011-2019. The year 2011 is a safe lower bound, as it predates the introduction of the term "Fog Computing" by Bonomi et al. [2]. The research repositories queried were the ACM Digital Library, IEEE XPlore, and Elsevier/ScienceDirect. An adjusted set of PICOC criteria [13] were used as search strategy, and a multi-step filtering process eventually resulted in 218 works that were selected for inclusion in the survey. The filtering process was done by independently by two authors, to keep researcher bias to a minimum. A vizualitation of the full filtering process can be seen in Figure 2.1, which involved two search queries (one broad, one narrow), duplicate removal, filtering based on title and abstract, filtering based on full-text reading, and addition of relevant papers through snowball sampling.

In order to answer RQ1, the works are analyzed for identified security requirements, which are then grouped by theme and further by specific requirement. For RQ2 to RQ4, relevant quantitative metadata is analyzed and visualized in figures, to show research trends. In the following subsections, we will first summarize the security requirements analysis, then the quantitative analysis, and finally the discussion on fog computing opportunities.

#### 2.1.1.1 Security Requirements

A total of 7 overarching themes are identified in the literature: authentication, access control, maintainability, resilience, data security and data sharing, security monitoring, network security, and models and methodologies. Most of these can be further divided into more specific categories, as is shown in the mindmap in Figure 2.2. For each of these categories, we discuss specific security requirements, and provide tables mapping investigated works to these requirements, based on which work identifies

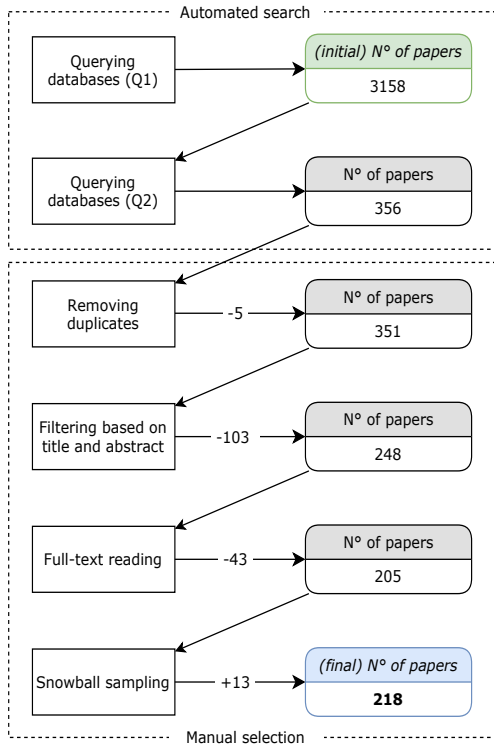



**Figure 2.1:** A schematic representation of the entire study selection process (source: [28]).

which requirement. Additionally, we show the relative popularity of these requirements within that field, to give an indication of which topics receive more attention within the scientific community.

The section closes with a comprehensive table listing each specific requirement, ordered by relative popularity. This table is included here as Table 2.1. In the context of this dissertation, notable requirements include: mutual authentication (A-06), key distribution (A-02), availability (NS-06), continuation of operation with compromised subsystems (R-01), secure data transport (DSS-04), and privacy-preserving authentication (A-07).

#### 2.1.1.2 Quantitative Results

The quantitative analysis shows an exponentially growing trend in security-related Industry 4.0 and IIoT publications since 2011, as can be seen in Figure 2.3. The growth really takes off after 2016, which we postulate might be related to several high-profile security incidents around that period.


**Figure 2.2:** A mindmap of the identified security requirements, grouped hierarchically by overarching categories and specific categories, and with identifying sources (from [27]) included in textboxes (source: [27]).

**Table 2.1:** An overview of each identified security requirement and its relative popularity (source [27]).

| Overall interest | ID | Security Requirement | Category | Overall % |
|---|---|---|---|---|
| Very High | SM-01 | infrastructure monitoring | Security Monitoring | 9.5% |
| | DSS-05 | secure external data storage | Data Security and Data Sharing | 7.1% |
| | A-06 | mutual authentication | Authentication | 4.6% |
| | MM-03 | security by design | Models and Methodologies | 4.6% |
| High | DSS-02 | data confidentiality | Data Security and Data Sharing | 3.9% |
| | A-02 | key distribution | Authentication | 3.5% |
| | SM-02 | threat response | Security Monitoring | 3.5% |
| | NS-07 | wireless transmission security | Network Security | 3.5% |
| | MM-01 | adequate risk/threat assessment | Models and Methodologies | 3.5% |
| | A-08 | minimization of user interaction | Authentication | 2.8% |
| | AC-04 | decentralized Access Control (AC) | Access Control | 2.8% |
| Medium | A-01 | multi-factor authentication | Authentication | 2.5% |
| | NS-04 | network isolation | Network Security | 2.5% |
| | A-07 | privacy-preserving authentication | Authentication | 2.1% |
| | NS-05 | timeliness | Network Security | 2.1% |
| | NS-06 | availability (Denial of Service (DoS), jamming, etc.) | Network Security | 2.1% |
| | A-03 | node addition, revocation, rekeying | Authentication | 1.8% |
| | A-04 | decentralized key management | Authentication | 1.8% |
| | AC-02 | fine-grained AC | Access Control | 1.8% |
| | R-01 | continuation of operation with compromised subsystems | Resilience | 1.8% |
| | R-03 | standards compliance | Resilience | 1.8% |
| | A-10 | attestation | Authentication | 1.4% |
| | AC-01 | handle dynamic changes | Access Control | 1.4% |
| | M-01 | software updateability | Maintainability | 1.4% |
| | M-08 | secure status transfer | Maintainability | 1.4% |
| | DSS-04 | secure data transport | Data Security and Data Sharing | 1.4% |
| | A-09 | non-repudation | Authentication | 1.1% |
| | AC-06 | transparency | Access Control | 1.1% |
| | M-02 | configuration updateability | Maintainability | 1.1% |
| | M-03 | disturbance-free updates | Maintainability | 1.1% |
| | DSS-06 | data flow control | Data Security and Data Sharing | 1.1% |
| | DSS-07 | data protection legislation compliance | Data Security and Data Sharing | 1.1% |
| | SM-04 | security policy enforcement | Security Monitoring | 1.1% |
| | NS-01 | dynamicity of configuration | Network Security | 1.1% |
| Low | AC-03 | centralized AC | Access Control | 0.7% |
| | AC-05 | privacy-preserving AC | Access Control | 0.7% |
| | M-05 | traceability | Maintainability | 0.7% |
| | M-06 | compatibility | Maintainability | 0.7% |
| | R-02 | operation with intermittent connectivity | Resilience | 0.7% |
| | NS-03 | management overhead minimization | Network Security | 0.7% |
| | A-05 | transitive authentication | Authentication | 0.3% |
| | AC-07 | compatibility | Access Control | 0.3% |
| | M-04 | usability of update process | Maintainability | 0.3% |
| | M-07 | transparency | Maintainability | 0.3% |
| | DSS-01 | data loss mitigation | Data Security and Data Sharing | 0.3% |
| | DSS-03 | standardization | Data Security and Data Sharing | 0.3% |
| | SM-03 | handle heterogeneous sources | Security Monitoring | 0.3% |
| | NS-02 | security policy enforcement | Network Security | 0.3% |
| | MM-02 | minimization of overall attack surface | Models and Methodologies | 0.3% |

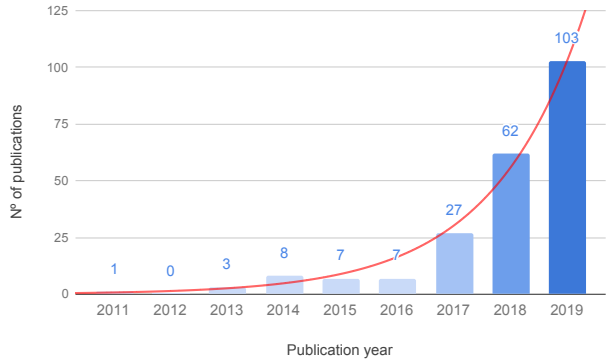



**Figure 2.3:** The number of Industry 4.0 and IIoT security publications over the period 2011-2019 (source: [27]).

Further, we analyze the geographical distribution of publications, which shows a disproportionate representation of German-speaking countries (22% of total publications). Finally, we look at the publication venues and observe that the vast majority of publications come from conferences and journals, with the "IEEE Transactions on Industrial Informatics" being the most popular venue, followed by "IEEE Access'.

#### 2.1.1.3 Opportunities for Fog Computing

For each of the overarching categories, we look at the identified security requirements, and provide a discussion on the potential solutions that Fog computing might bring. For example, it is proposed that Fog nodes could act as local Certificate Authorities for distributed systems, which would address concerns about relying on third-party services for authentication. This is characteristic for most of the opportunities discussed in the section; it is clear that Fog nodes operating as buffers in this middle ground between the network edge and cloud can greatly help with addressing modern day IIoT security issues. The section closes with a short discussion on the challenges and limitations of Fog computing. For example, it would shift significant responsibility from service providers to Fog node maintainers, posing a risk when e.g. critical security updates must be performed.

### 2.1.2 Closing Remarks

In this survey we identified and categorized a multitude of security requirements that have previously been discussed in the scientific literature. Additionally, we looked at quantitative data surrounding these works to provide an image of the growth and geographic popularity of the topic. Finally, we discussed the potential for Fog computing in addressing these security requirements.

The security requirements collected in this work provided us with a motivation to look at securing the PTP protocol, which is discussed in Chapter 3, as well as improving the usability for the TLS protocol for the IoT and IIoT, discussed in Chapters 4 and 5.

CHAPTER 3

# Improving Precision Time Protocol Reliability

Distributed real-time systems that rely on precise synchronization between subsystems, often require a mechanism to establish a common time-frame and correct for clock drift between individual subsystems. The IEEE 1588 [9] standard defines the Precision Time Protocol (PTP) to address this issue. This protocol has been in use for decades in various industrial fields, and is considered the golden standard for time synchronization. Unfortunately, due to its age, it has not been designed with security in mind. This opens the protocol to a multitude of threats in modern industrial deployments which are increasingly often connected to the internet. Recognizing this, the most recent iteration of IEEE 1588 (IEEE 1588-2019 [10])standard includes several security measures which when enabled successfully protect against a variety of attacks. However, the standard only includes minimal discussion on how system reliability is affected when one of the subsystems is compromised and tries to disrupt clock synchronization in the overall system.

While researching protocols candidates for goals G-II and G-III, it became clear that the IEEE 1588-2019 standard fails to satisfy security requirement R-01[1] and by implication NS-06[2], both of which are identified in Paper B (see Chapter 2). At the same time, the PTP protocol is used in modern IIoT and Fog era communication protocols such as Time-Sensitive Networking (TSN) and 5G mobile infrastructure. These observations eventually led to the work described in Paper C. Thus, the PTP protocol was chosen as a first protocol for goal G-II, and the resulting proposed improvements are in pursuit of G-III.

[1] Continuation of operation with compromised subsystems
[2] Availability

## 3.1 [Paper C] Fault-tolerant Clock Synchronization using Precise Time Protocol Multi-Domain Aggregation

With the publication of the IEEE 1588-2019 [10] standard, the PTP protocol has received a number of additions strengthening the security of this protocol. These include authentication and encryption of its communication, and architectural considerations. One proposed security measure included in the standard involves adding redundancy to the network by modifying the network communication links, so that there exist more than one path between any two nodes. Unfortunately, no implementation details are given, nor what kind of perfomance impact this might have on the quality of clock synchronization in the system. Using network simulation tools, This work explores how a redundant PTP system can be implemented, as well as the impact on clock synchronization quality.

### Personal Contribution

This work is the result of a collaboration between DTU Fotonik and DTU Compute. It represents an integral part of both my research, as well as that of Eleftherios Kyriakakis, a colleague PhD student at the ESE section of DTU Compute. We both invested a significant amount of time into it, and ended up with a roughly equal split of the work collected in this publication. We decided to share first authorship, as neither of us could claim responsibility for a majority of the work. I contributed to the ideation of the proposed work, and was reponsible for approximately half of the implementation. Additionally, I defined the algorithm pseudocode and wrote large parts of Section IV. I contributed to the evaluation topologies, and wrote part of the discussion.

### 3.1.1 Extended Summary

The IEEE 1588 Precision Time Protocol [9] is a distributed hierarchical clock synchronization protocol that works over Ethernet. It follows the classical Master-Slave paradigm, and autonomously selects a Master that dictates the time, based on which participating node advertises the most accurate Master Clock. This algorithm is called the Best Master Clock Algorithm (BMCA). Once a Master is elected, it will periodically transmit offsets to the rest of the network over a logical tree topology, which are then used by Slave nodes to correct their clocks if necessary.

With the IEEE 1588-2019 iteration of the standard, security measures are introduced to provide authenticity guarantees as well as confidentiality of PTP messages. However, it presents little information on how to protect against delay attacks (where PTP messages are deliberately delayed by an adversary, thereby possibly skewing

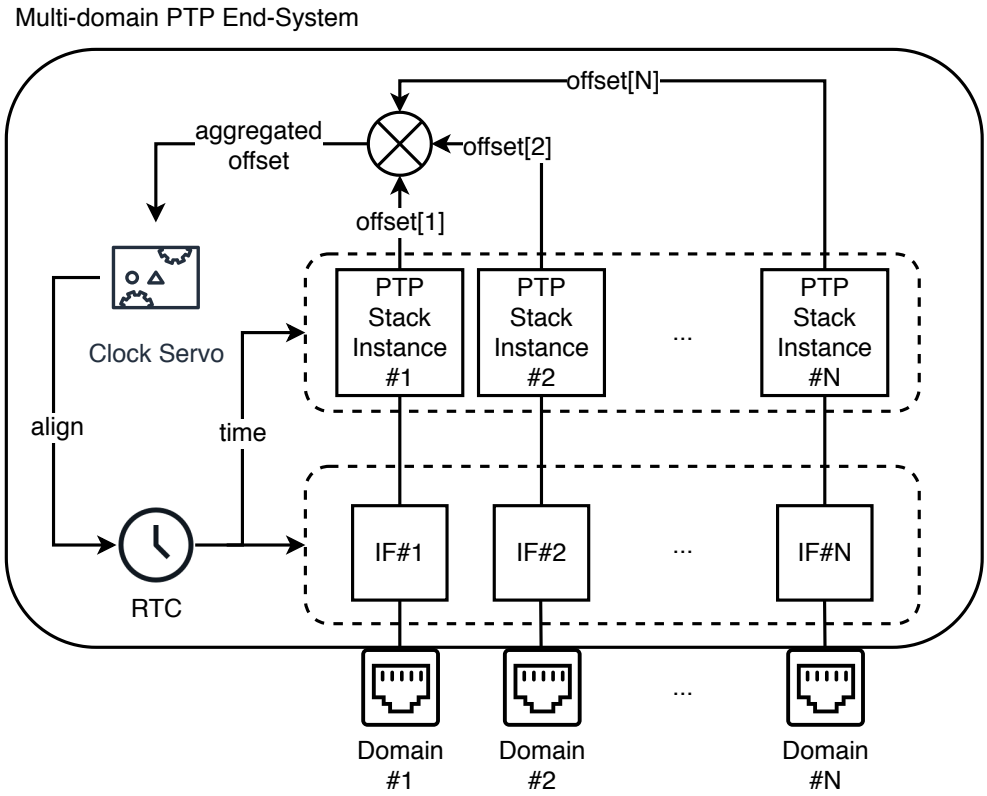



**Figure 3.1:** The proposed architecture of a redundant PTP node. For each individual PTP stack, a separate network interface is used, to stimulate physical redundancy. The $\oplus$ symbol denotes the convergence algorithm (source: [14]).

clock times of Slaves). Another threat that is not taken into consideration, is that of a faulty Master node, e.g. because it is compromised by a malicious actor. Without going into detail, the standard hints at either introducing redundant Master nodes, or deploying redundant network topologies running in parallel PTP domains. This work explores the latter, proposing a node architecture that can work with parallel PTP domains, as well as comparing two convergence algorithms responsible for aggregating the offsets received from each domain.

The first contribution, a redundant PTP node design, is visualized in Figure 3.1. To tolerate $f$ Byzantine (i.e. arbitrary) faults, the node will need to run $n = 3f + 1$ PTP stacks in parallel. In our design, we completely separate the network interfaces as well. Running all PTP stacks over a single interface is possible but introduces a single point of failure. A key component of the node architecture is the convergence algorithm. This algorithm is transparent to the rest of the system and aggregates the

most recently received offsets for each domain into an aggregated offset which is fed to the Real-Time Clock (RTC).

The work introduces a windowed decision algorithm, which separates the last received time offsets on each domain into observation windows. Whenever the node receives a new correction frame on any domain, it will construct an observation window defined by the frame ingress timestamp and a carefully tuned time difference threshold. The observation window is used to find eligible frames on each domain that will be considered for the aggregation algorithm, and to create an average ingress timestamp that (together with the aggregated offset) is fed into the correction algorithm for the RTC. Two aggregation algorithms are considered: a simple averaging algorithm (AVG) that averages incoming offsets, and a fault-tolerant aggregation algorithm (FTA) that is able to tolerate $k$ faulty offsets in its input, without compromising significantly on output accuracy.

The redundant node architecture, windowed decision algorithm, and both aggregation algorithms are implemented as additions to libPTP [32], a PTP simulation library for the OmNet++ [31] network simulator, and evaluated for two network topologies, depicted in Figure 3.2. For a baseline comparison, the topologies include a regular PTP Slave node as well.

For the first topology, only a a Master node link failure is simulated, where after a specified time period the individual links between the Master node and its transparent clocks fail at 30-second intervals. This represents a DoS scenario and both the AVG and FTA algorithms turned out to properly mitigate this type of failure.

The second topology is more involved: it is comprised of two separate topologies, as it was not deemed sensible to evaluate regular PTP Slave node over the same topology; it would only be able to connect to a single Master, and once that link fails there is no backup. Instead, the regular Slave node has its own topology which (using the BMCA) will dynamically elect new Master nodes after the first node fails. Two scenarios are evaluated over these topologies. The first scenario is identical to the timed link-failure scenario described above, and provides similar results. The second scenario considers a malicious master clock that adds itself to the network and attempts to skew Slave clock times by advertising as the most accurate clock in the system, while broadcasting skewed offsets. The results of this scenario are included in Figure 3.3. Here, a clear distinction between the AVG and FTA algorithms can be seen. The AVG algorithm is susceptible to this malicious node and its aggregated offset gets skewed when the attack starts. The FTA however, stays completely stable and provides an accurate offset to its clock. At the same time, the regular, BMCA-elected node shows a large dip in its clock drift, which appears to be corrected quickly when a new Master is elected. However, it bears worth noting that this is because the malicious Master is elected, and since that is its only time source, it cannot detect that its clock is skewed after the dip.

The work concludes with a discussion on the overhead of this solution and its feasability. When the PTP stacks are completely software-based, the overhead is minimal, with preliminary results showing only 1% CPU overhead. The main overhead will be in setup- and materials costs, as the network topology requires physical

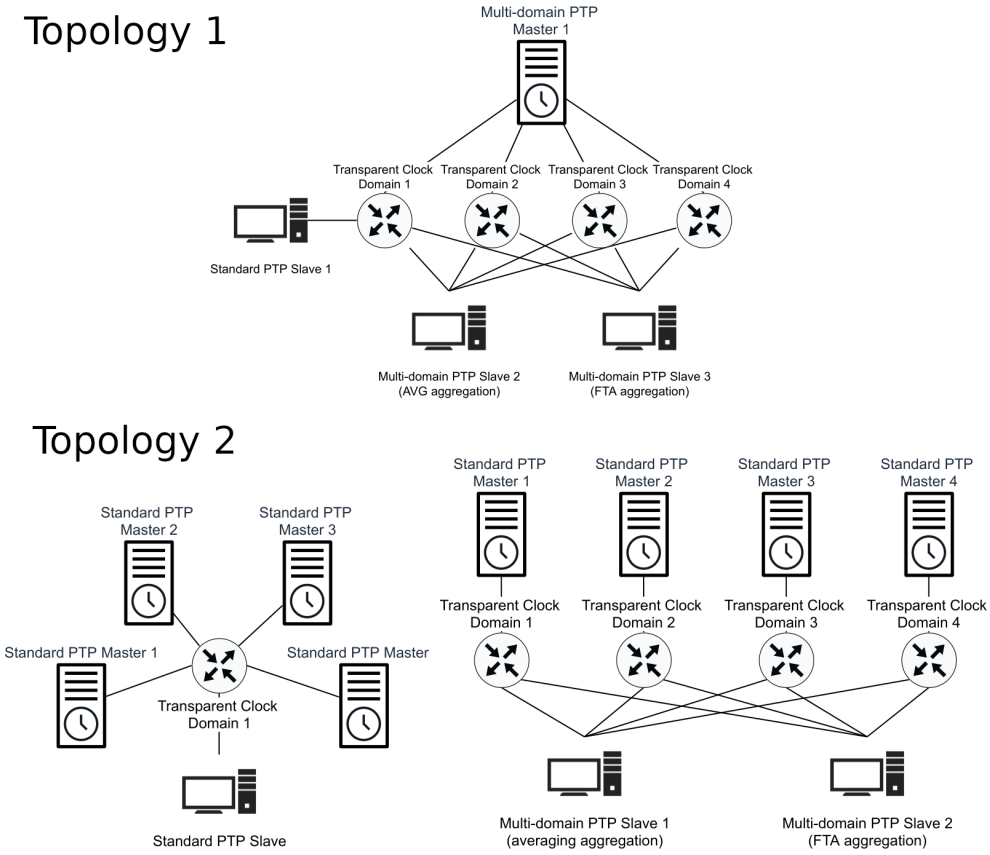



**Figure 3.2:** The network topologies used in the evaluation of this study. Note that Topology 2 consists of two separate topologies, since it is deemed more valuable to evaluate the FTA nad AVG algorithms against a regular BMCA algorithm that will elect backup Master nodes when a failure is detected (source: [14]).

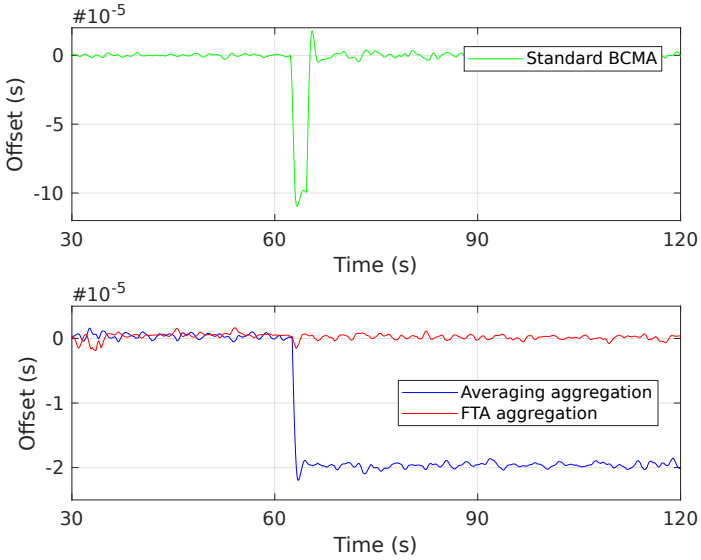



**Figure 3.3:** Measured PTP clock offsets when a malicious Master node takes over at $t = 60$. The first graph shows a normal PTP node, which, apart from the dip, cannot observe that its clock has been skewed. The second graph shows the AVG and FTA algorithms. The AVG can detect the skew, but only the FTA can also prevent it. (source: [14]).

redundancy and if the PTP stacks are hardware-replicated, a significant amount of links and switches as well. Finally, future research opportunities are discussed. For example, there is potential to decrease the overhead by using a hybrid solution where only a subset of nodes runs a fully redundant PTP system.

### 3.1.2 Closing Remarks

In this work we looked at increasing the resilience and security of the the PTP protocol. A redundant node architecture was proposed, which together with the presented offset aggregation algorithms can make a PTP system resilient against a defined number of compromised subsystems, as well as delay attacks. The solution is complementary to security measures already introduced in the IEEE 1588-2019 standard [10], and together they can create robust PTP systems for modern connected industrial environments.

By filling in a gap in the PTP standard, we effectively provided an extension to the protocol that is able to satisfy security requirements R-01 and NS-06.

CHAPTER 4

# TLS: a Solution for Fog and IoT Devices?

The Transport Layer Security (TLS) protocol and its predecessor Secure Socket Layer (SSL) have already seen service as the de facto standards for Web security for decades. Over the years, multiple attempts have been made to adapt these protocols for embedded and IoT scenarios. Despite those attempts, the protocol is often considered too resource-intensive for lightweight devices. At the same time, the TLS protocol and its key infrastructure are widely deployed and understood. Thus, for interoperability reasons, it could be beneficial for lightweight devices to have the ability to communicate over TLS, despite its power costs. A solution for the IIoT and Fog involving TLS could also satisfy multiple popular security requirements identified in paper B. With the main design goal of TLS being the Web, resource usage optimization does not score highly on its list of priorities. At the same time, the modular design means that novel TLS extensions could help to reduce its resource footprint. This makes the TLS protocol a good candidate for a protocol that might be adapted to work for the IIoT.

In this chapter, we first look at an example where TLS can benefit the IIoT and Fog fields by exploring a potential use case for in a Fog environment. Since the TLS protocol offers many extensions that augment its functionality(e.g., by allowing session resumption, different cryptographic suites, or compression), we also look at the performance impact of these extensions. Finally, we analyze privacy implications of modern TLS session resumption extensions.

All the works in this chapter build up to a solution presented in Chapter 5, which aims to reduce the bandwidth (and by implication power and CPU) usage for TLS 1.3 session resumption. As such, the content of this chapter can be considered in pursuit of goal G-II.

## 4.1 [Paper D] Fogification of electric drives: An Industrial Use Case

In order for factories to be properly equipped for the Industry 4.0 era, new technologies are needed that can bridge the gap between traditional Information Technology (IT) (e.g. Cloud services, AI) and OT (e.g. Cyber-Physical System (CPS)). The Fog computing paradigm promises to do this by enabling Cloud-like technologies while preserving quality-of-service and dependability guarantees necessary for industrial environments. In this paper, we explore a use case for Fog Computing Platform (FCP) nodes, by looking at a traditional electric drive and proposing a fogified version that turns this drive into a FCP node. From a security standpoint, this poses an interesting challenge as a fogified drive will be connected to the internet, and we explore the possibilities for securing communication with the FCP node using the TLS protocol.

### Personal Contribution

This paper was a collaboration between Danmarks Tekniske Universitet, Mälardalen University, and Danfoss Power Electronics, and individual authors worked on separate parts of this work. My contribution to this paper was solely focused on its security aspects: I analyzed the scenario from a security standpoint and formulated a security framework for the fog node, which I described in the corresponding section of the paper. Additionally, I was involved in proofreading and final revisions before submission. Since I had no involvement in any other evaluation subsection other than the discussion on security mechanisms, this summary is primarily focused on the security-related contributions.

### 4.1.1 Extended Summary

This use case study starts with an introduction to the current state of industrial systems: OT and IT are still mostly separated, but these domains need to converge for Industry 4.0 to become a reality. The Fog paradigm is introduced as a system-level paradigm that can address this problem. As a use case, electric drives are studied. These types of drives are found in many domains, such as e.g. automotive, food and beverage, and airconditiong.

Electric drives operate on the “control level” of the automation pyramid (see Figure 4.1), while producing large amounts of data that can be processed in higher (IT) layers. This data is often considered very sensitive and factory operators are reluctant to share it with third-parties. Therefore, on-site solutions are preferred. A FCP node could thus fit perfectly into this problem domain.

The specific type of drive studied is the VLT electric drive by Danfoss electronics [4]. This drive runs its own real-time operating system and connects over a Fieldbus using the Profinet/RT [22] standard. The assumption is that such a drive is

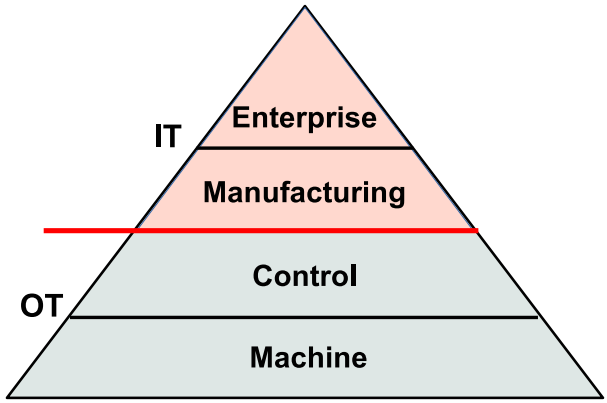



**Figure 4.1:** The automation pyramid, where the bottom two levels are considered OT, and the top two are considered IT (source: [1]).

connected to a Programmable Logic Controller (PLC) and a Human-Machine Interface (HMI) to configure the drive. This architecture is referred to as the baseline. After a description of the drive, a fogified architecture is introduced, which consists of a hardware platform that can run the tasks necessary to operate the drive, as well as mixed-criticallity tasks. This is achieved by using a hypervisor (PikeOS [12]) that can enforce temporal partitioning between applications. Additionally, the fogified architecture features a TSN-enabled switch for communication over the network. Both architectures are modeled in Architecture Analysis & Design Language (AADL) [7]. While the fogified architecture introduces some unpredictability compared to the baseline (due to the transition from spatial to temporal isolation), it does gain the flexibility of a programmable platform for monitoring purposes, and e.g. machine learning applications can be used already in the drive itself.

A number of system-level requirements are identified and presented in Table 4.1. Out of these, requirement 7 is of special interest in the context of this thesis: the drive requires secure access to the Cloud. A number of technology bricks are introduced to satisfy these requirements for the fogified drive. Specifically, these consist of a FCP configuration, machine learning framework, a fault detection isolation and recovery method, and a set of security mechanisms.

In order to evaluate the fogified architecture, a conveyor belt use case is considered. In this use case, the drive uses the same communication medium for hard and soft real-time communication, as well as non-critical communcation. To guarantee satisfiability of timing requirements, TSN is used with a constraint-programming based communication schedule.

Then security requirements and solutions for the FCP node are discussed. The threats together with their mitigation strategies are summarized in Table 4.2. Because the drive will need to communicate sensitive data over the Internet, a confidential and authenticated communication channel is necessary. For this, TLS is suggested as

**Table 4.1:** System-level requirements for Fog-based drives (source: [1]).

| # | **Requirement** | **Realization in the baseline architecture** | **Realization in the fogified architecture** |
|---|---|---|---|
| 1 | Drives shall be designed according to industrial standards | IEC61800-based design | IEC61800-based design |
| 2 | Drives shall have a time-constrained interface | 1 ms time-constrained Profinet interface | Jitter-free TSN interface |
| 3 | Drives shall be able to monitor and process data for predictive maintenance purposes | Machine learning framework with appropriate safety integrity levels | Machine learning framework with appropriate safety integrity levels |
| 4 | Drives shall run mixed-criticality applications according to industrial standards | Spatial separation according to IEC61508 | Temporal separation according to IEC61508 |
| 5 | Drives shall control the electric motor accurately | Motor control with a response time of 30ms and good quality-of-control | Motor control with a response time of 20ms and good quality-of-control |
| 6 | Drives shall be configurable according to industrial standards | Configurable according to IEC61508 | Configurable according to IEC61131 |
| 7 | Drives shall have secure access to the Cloud | Cloud connection provided by external devices | Cloud connection provided by the TSN interface, with security mechanisms |

**Table 4.2:** Threats and their mitigations (source: [1]).

| **Threat** | **Mitigation** |
|---|---|
| Man-in-the-middle, impersonation | Confidential, authenticated comm. channels |
| Attack impact | Service isolation (e.g., partitions) |
| Remote attacks | Firewalls, endpoint whitelisting |
| DoS | Redundant network topologies |
| TSN security | Isolation of the TSN protocol, per-stream filtering |
| Physical attacks | Hardware token for configuration changes |
| Detection | Security monitoring services |

an excellent candidate. Additionally, a firewall together with endpoint whitelisting is suggested to mitigate the attack surface of the fog node as much as possible. Then, the suggestion is made to place applications that connect to remote services (such as the Cloud) into separate partitions managed by the Hypervisor. This sandboxing mitigates the impact of threats that consider those remote services as entry vectors. Additionally, a security monitoring service should run in a separate high-priority partition, so that it can collect data for forensic purposes and detect anomalous behaviour on the system. The TSN protocol provides minimal security features due to its time-sensitive nature, and its software stack should also be isolated from other

partitions where possible. Further, the network traffic itself should be isolated as much as possible, for example through use of Software-Defined Networking (SDN). Finally, some form of authentication for configuration changes is necessary. This could come in the form of e.g. an NFC authentication dongle that must physically be close to a Fog node when committing configuration changes.

Next, predictive maintenance is addressed, where a distributed machine learning framework is proposed to train a global model that can then mark mechanical parts for predictive maintance, based on their expected failure times. Simulated results show an accuracy of up to 97.5%.

After predictive maintenance, a method for fault detection, identification, and recovery is presented. The proposed method aims to provide safety assurance for the safety requirements defined in the IEC 61508 [8] standard. While the primary safety requirement for any drive is to stop the motor in an emergency situation, extra safety requirements are defined for fogified drives. These requirements include a safe timer, deadlock prevention, the requirement of defined behavior upon detection of a failure, adequate redundancy for safety measures, and a requirement for exra validation of run-time changes (e.g. through firmware updates) to critical parts of the fogified drive. Finally, a number of operational states and corresponding safety actions are proposed.

### 4.1.2 Closing Remarks

In this work, we proposed a fogified electric drive architecture, and evaluated its impact with a conveyor belt use case. For the security framework, it turned out that TLS is a promising candidate for securing external communication channels of Fog nodes. Some observations can be made based upon the security-related work conducted in this paper: Firstly, if the communication medium is shared with time-critical services, bandwidth overhead of TLS traffic should be kept to a minimum. Secondly, this indicates that Edge devices connected to a Fog node could benefit from TLS support, as that would immediately enable them to have a more secure communication channel with the Fog node. Together, these observations hint at a research opportunity for minimizing TLS traffic overhead for embedded devices. Indeed, this is something that we explore in Chapter 5.

## 4.2 [MSc. Thesis 1] TLS Extension Performance Impact

The TLS protocol is modular, and supports many extensions. Each extension has an identifying number, the most common of which are standardized by the Internet Assigned Numbers Authority (IANA) [18]. Some of these extensions are considered mandatory for TLS to operate, while others are optional. While some extensions cause only minor behavior changes in the protocol, others can completely change the outcome of a TLS handshake. A natural question that then arises is: how do these extensions impact protocol performance? And further: If the design is modular, can

we strip a TLS handshake down to only the bare minimum of extensions, perhaps reducing its footprint and bringing it within reach of lightweight devices?

To answer these questions, the MSc. thesis "TLS Extension Performance Impact" [17] by A. K. Mathiasen and E. Bejder presents an evaluation of several high-profile TLS extensions and their impact on protocol performance.

### Personal Contribution

This thesis is an original work by A. K. Mathiasen and E. Bejder, supervised by X. Fafoutis and myself. My role as a supervisor included defining the original research problem, as well as continuous supervision during the thesis project. This involved regular meetings, guidance on which extensions to evaluate, which performance metrics to use, and feedback on the thesis structure.

## 4.2.1 Extended Summary

This thesis measures the performance impact of many standardized TLS 1.3 [23] extensions. Its main goals are firstly to provide said performance analysis, secondly to give a foundation for further research into TLS 1.3 extensions, and finally to provide configuration recommendations based on the performance evaluation conducted in this work.

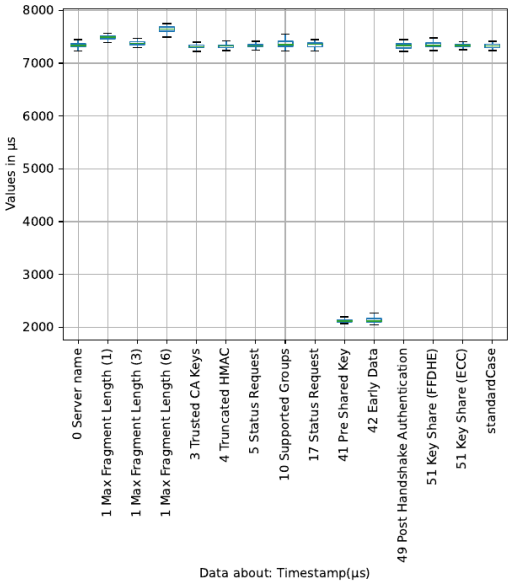
**Table 4.3:** Table of the tested TLS extensions (source: [17])

| IANA value | Extension name | Reference |
|---|---|---|
| 0 | `server_name` | RFC6066 [6] |
| 1 | `max_fragment_length` | RFC6066 [6] |
| 3 | `trusted_ca_keys` | RFC6066 [6] |
| 4 | `truncated_hmac` | RFC6066 [6] |
| 5 | `status_request` | RFC6066 [6] |
| 10 | `supported_groups`[1] | RFC6066 [6] |
| 17 | `status_request_v2` | RFC6961 [21] |
| 41 | `pre_shared_key` | RFC8446 [23] |
| 42 | `early_data` | RFC8446 [23] |
| 49 | `post_handshake_auth` | RFC8446 [23] |
| 51 | `key_share` | RFC8446 [23] |

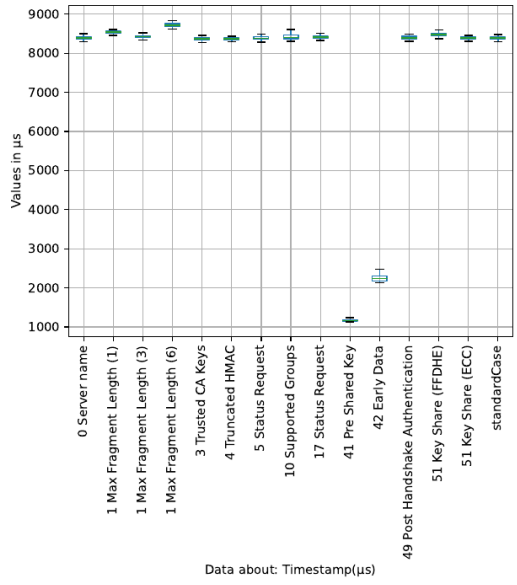
The extensions chosen for measurement are included in Table 4.3. Out of these, `server_name` and `key_share` are mandatory in any TLS-compliant implementation. Additionally, `pre_shared_key` is mandatory for Pre-Shared Key (PSK) agreement, while the `supported_groups` extension is mandatory for elliptic curve key exchanges.

The evaluation is conducted over an instrumented fork of the WolfSSL [33] library. This library was chosen due to its support for a wide variety of extensions, embedded-friendliness, active community, and comprehensive documentation. The

changes made to the source code introduce a performance metric data structure that captures relevant metrics during run-time. The choice to do this from within a modified WolfSSL library is motivated by the fact that WolfSSL manages its own memory and clears data structures when they are no longer used. In addition to the instrumented WolfSSL library, a data collection script and data processing script are used.



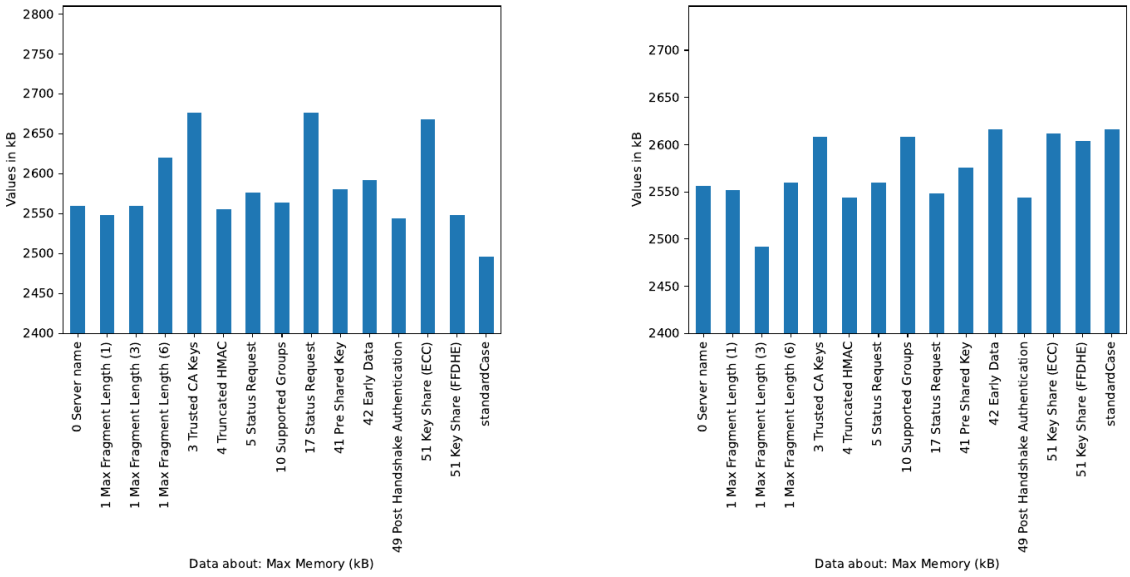

(a) **Client**.




(b) **Server**.

**Figure 4.2:** Timestamp distributions for the tested extensions. Here, timestamps indicate the duration of a handshake (source: [17]).

The performance evaluation focuses on three performance metrics: Round-Trip Time (RTT), memory usage, and transmission size (in bytes). The WolfSSL TLS handshake is measured on a modern x86 desktop machine. In Figure 4.2, timestamp distributions for the tested extensions, for both client and server are shown. The measured timestamps describe the duration of a TLS handshake. What stands out is that the choice of maximum TLS fragment length can result in longer execution times. Other than that, the choice of extension has little impact – except for the PSK and early data extensions. This makes sense; when a PSK is used to set up a TLS session, the handshake can omit certain computationally intensive steps, leading to shorter execution times. The early data extension is a completentary extension to the PSK extension, indicating that application data is already included in the first flight of messages. Similarly, the network transmission overhead depicted in Figure 4.4 shows that even though extensions generally only have small impact, the use of PSK and early data extensions relate to a significantly lower transmision overhead. This is




(a) **Client** (b) **Server**

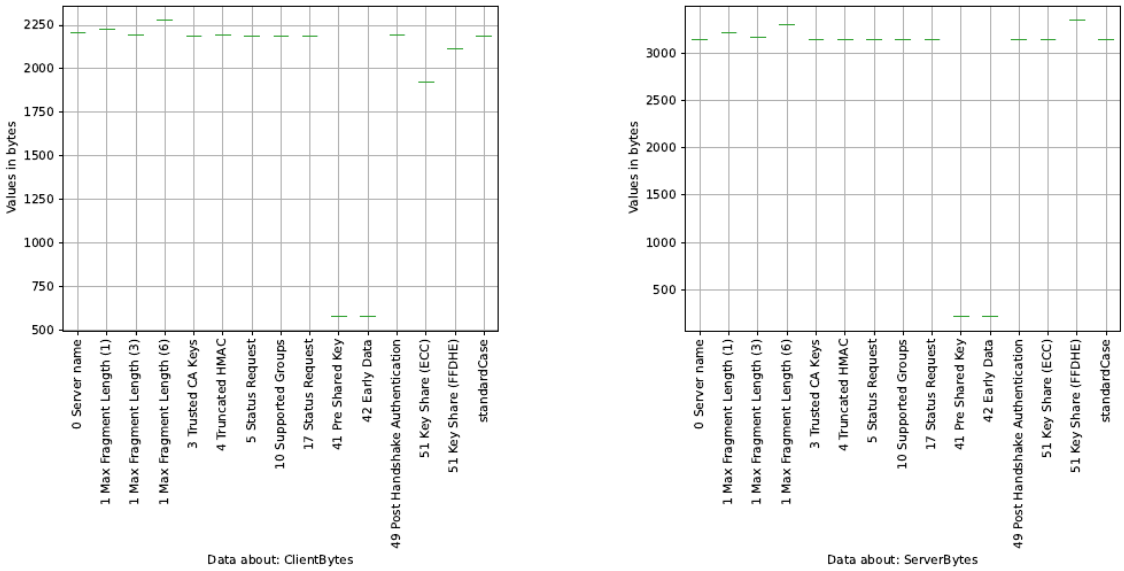
**Figure 4.3:** Peak memory usage of TLS client and server, with different extensions enabled (source: [17]).

because in TLS 1.3, these can be used with a fast session resumption protocol that requires fewer round-trips than a normal handshake. Memory usage is more varied, as can be seen in Figure 4.3. These vary between around 2.5 to 2.6 kB peak memory usage. The elliptic curve key share, status request, and trusted Certificate Authority (CA) key extensions require relatively more memory than the other extensions.

Finally, the thesis discusses possible directions for future work, including testing on embedded devices, in more realistic network conditions, testing different TLS libraries, and extending the collection of extensions on which tests are performed.

### 4.2.2 Closing Remarks

In this thesis project, the performance impact of TLS 1.3 extensions was evaluated. The evaluated metrics included time overhead, network bandwidth overhead, and memory usage overhead. It was found that TLS session resumption (the PSK and `early_data` extensions) required significantly less bandwidth, indicating that using session resumption can benefit embedded scenarios where network transmission costs are not negligible. This observation supports the research presented in Chapter 5, where we introduce an extension for the TLS session resumption protocol aimed at lightweight IoT and IIoT devices.




(a) Sum of bytes sent from the client

(b) Sum of bytes sent from the Server

**Figure 4.4:** The total amount of bytes sent over the network during a TLS handshake with different extensions enabled for client and server (source: [17]).

## 4.3 [MSc. Thesis 2] A Practical Study on Online Tracking Using TLS Session Resumption

Today, many websites employ tracking methods to track their users in one way or another. Some companies even specialize in this, and track users across websites to get a comprehensive image of their online behavior, which is useful for advertising purposes. Historically, these tracking methods utilize existing popular protocols to fingerprint their users. With session resumption in TLS 1.3, such an opportunity arises: trackers could craft special resumption tickets that the client will present to a server upon session resumption. This could unambiguously link two TLS sessions to each other. In this thesis project, the student was asked to explore the potential for this tracking mechanism, as it might pose a threat for IoT applications.

If TLS session resumption were to be used in the IoT and IIoT fields, then the possibility of tracking mechanisms poses a risk to security requirement A-07[2] identified in paper A. As such, it is important to analyze the impact this tracking can have, in pursuit of research goal G-II.

[2]Privacy-preserving authentication mechanisms

### Personal Contribution

This thesis is an original work by J. Pecl, supervised by N. Dragoni and myself. My contribution as supervisor consisted of proposing the original research problem as well as regular supervision meetings and guidance. I helped formulate the research goals, and provided a testing environment for testing TLS 1.3 session resumption mechanisms. Additionally, I provided guidance on the workings of TLS.

## 4.3.1 Extended Summary

This thesis explores the potential for tracking using TLS 1.3 session resumption tickets. First, the problem statement and research goals are explained. The TLS 1.3 protocol features a novel session resumption mechanism that has been received with some scepticiscm by the privacy-advocacy community, because it allows the server to give an opaque block of data to the client, which it will later (when session resumption happens) transmit back to to the client. This could be used to correlate two TLS sessions. Now, with TLS also moving towards the IoT and IIoT fields, this can pose a privacy risk for those devices as well. In this thesis, the following research questions are discussed:

- Comparing TLS session resumption tracking to other tracking mechanisms
- Making session resumption visible to users
- An update of an earlier study on TLS session resumption tracking [25]

Then, a systematic survey of online tracking mechanisms is presented. The vast majority of these tracking mechanisms are unique to the Web, but are very important to include as they provide a context in which we can evaluate the potential for TLS resumption tracking. Figure 4.5 summarizes the various tracking mechanisms. It turns out that TLS resumption tracking performs poorly compared to other tracking mechanisms, as the resumption ticket lifetime is limited to the (browser) cache lifetime. Additionally, the resumption ticket only provides a few hundred bytes of space, compared to Kilo- or even Megabytes for some other methods. Further, browsers often allow users to wipe their cache at will, thereby removing any stored session tickets. On the other hand, TLS session resumption tracking is very hard to identify, unlike most other methods. This stealthiness motivates the rest of the thesis, where a tool for detecting TLS session resumption tracking is developed.

After the survey, the development of this TLS session resumption tracking tool is described. The tool allows users to select a network interface which will then be monitored for TLS handshakes (specifically, ClientHello, ServerHello and NewSessionTicket messages on the TLS record layer). The tool is able to detect both TLS 1.2 and TLS 1.3 session resumptions. These differ in that TLS 1.2 uses the deprecated `session_ticket` and `session_id` extensions, while TLS 1.3 uses the

| Mechanism | Technology | Lifetime | Deletion | Detection | Accuracy | Delay | Avoidability | Cross-window | Cross-domain | Cross-session | Cross-browser | Cross-device | Storage size |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Session-only** | | | | | | | | | | | | | |
| Session identifiers | Web-server session | Web-session | | | | | ✓ | ✓ | ✗ | ✗ | ✗ | | - |
| Window.name DOM property | HTML5, Javascript | Web-session | | | | | ✓ | ✓ | ✗ | ✗ | ✗ | | 2MB |
| Web-form authentication | Web-server session | Web-session | | | | | ✓ | ✓ | ✓ | ✓ | ✓ | | - |
| **Storage based** | | | | | | | | | | | | | |
| HTTP cookies | HTTP header, JavaScript | ∞ | | | | | ✓ | ✓ | ✓ | ✗ | ✗ | | 4KB |
| HTML5 Local Storage | HTML5, Javascript | ∞ | | | | | ✓ | ✓ | ✓ | ✗ | ✗ | | 25MB |
| HTML5 Session Storage | HTML5, Javascript | Brows.session | | | | | ✗ | ✗ | ✗ | ✗ | ✗ | | 25MB |
| HTML5 WebSQL & IndexedDB | HTML5, Javascript | ∞ | | | | | ✓ | ✓ | ✓ | ✗ | ✗ | | - |
| *Flash cookies*[(2)] | Flash, Java | ∞ | | | | | ✓ | ✓ | ✓ | ✓ | ✗ | | 100KB |
| *Flash LocalConnection Object*[(2)] | Flash | ∞ | | | | | ✓ | ✓ | ✓ | ✓ | ✗ | | 40KB |
| **Cache-based** | | | | | | | | | | | | | |
| Cache cookies | HTML5, Javascript | Cache | | | | | ✓ | ✓ | ✗ | ✗ | ✗ | | - |
| Loading performance test | Server measure, JavaScript | Cache | | | | | ✓ | ✓ | ✗ | ✗ | ✗ | | - |
| HSTS Cache | HTTP header, Javascript | Cache | | | | | ✓ | ✓ | ✗ | ✗ | ✗ | | - |
| HTTP 301 Redirect cache | HTTP header | Cache | | | | | ✓ | ✓ | ✗ | ✗ | ✗ | | 48KB[(1)] |
| TLS Session Resumption | TLS protocol | Cache | | | | | ✓ | ✓ | ✗ | ✗ | ✗ | | 0.5KB[(3)] |
| QUIC Session Resumption | QUIC protocol | Cache | | | | | ✓ | ✓ | ✗ | ✗ | ✗ | | 0.5KB[(3)] |
| E-Tags / Last modified | HTTP header | Cache | | | | | ✓ | ✓ | ✗ | ✗ | ✗ | | 10KB |
| HTTP Authentication Cache | HTTP header, Javascript | Cache | | | | | ✓ | ✗ | ✗ | ✗ | ✗ | | 48KB[(1)] |
| **Fingerprinting** | | | | | | | | | | | | | |
| Network and location | IP, HTTP header, JavaScript | ∞ | | | | | ✓ | ✓ | ✓ | ✓ | ✓ | | - |
| Device fingerprinting | IP, HTTP headers, JavaScript | ∞ | | | | | ✓ | ✓ | ✓ | ✓ | ✗ | | - |
| OS instance fingerprinting | IP, HTTP headers, JavaScript | ∞ | | | | | ✓ | ✓ | ✓ | ✓ | ✗ | | - |
| Browser fingerprinting | HTML5, HTTP headers, JavaScript | ∞ | | | | | ✓ | ✓ | ✓ | ✗ | ✗ | | - |

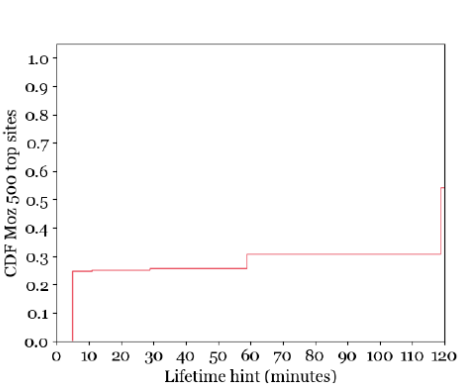
[(1)]defined by server (nginx max size here) [1] [(2)]No longer supported by main browsers
[(3)]Order of magnitude (tokens size may vary with server implementation)

**Figure 4.5:** An overview of Web tracking mechanisms. Grey indicates deprecation, while the different shades of red denote ease or intensity depending the metric, with light red being easy / hardly any, and dark red being difficult / very much so (source [19]).

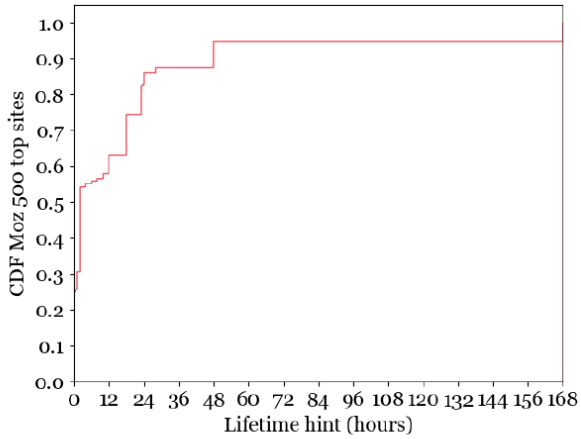
`pre_shared_key` extension. Note that TLS 1.2 has two resumption methods; session ID resumption is a separate mechanism from session ticket resumption. The tool was tested on a specially crafted website that opens two TLS sessions to different subdomains (this enables testing resumpion with third-party domains), and additionally closing the TLS session every few seconds to force renegotiations (and thus resumptions). Additionally, the tool is tested against a number of public websites. With some of these, it was found that different third-party domains received TLS 1.2 session resumptions with the same session ID, indicating that the corresponding servers were sharing session states. Additionally, some browser-specific behaviors were observed that are in violation of the TLS 1.3 RFC [23] recommendations.

In the third part of the thesis, an update to the original TLS session resumption tracking paper [25] is given, as the original study did not discuss the 1.3 iteration of the

TLS standard. Specifically the behavior of 500 popular websites, as per the Moz Top 500 [11], is studied. This dataset is further trimmed to 427 sites by removing duplicate entries (e.g. amazon.co.uk and amazon.de). First, the support for session resumption in TLS 1.2 is studied. Out of the 427 probed sites, 363 domains supported session resumption, and another 12 indicated they supported resumption but failed to deliver a session token. Only 38 sites indicated they did not support resumption, and 14 sites were not reachable during the study. Then, resumption for TLS 1.3 is studied. There, it is found that 34% deliver session tickets, 39% deliver `pre_shared_key` extensions, 18% deliver session IDs (this is a legacy field in TLS 1.3), and 9% do not deliver any resumption mechanisms at all. Note that in TLS 1.3, the `pre_shared_key` extension is used to advertise support for PSK resumption while the tickets themselves are delivered in a separate (`NewSessionTicket`) message. Thus, the fact that there is a discrepancy between these numbers suggests there were servers in the study that advertised support for session resumption but failed to deliver a ticket, possibly due to incompatible resumption method choices between client and server.




(a) Less than 120 minutes




(b) All lifetimes

**Figure 4.6:** Cumulative distribution of lifetime hints (source: [19]).

An additional aspect that is studied for each probed site, is the session lifetime. Two separate lifetime values are recorded for each website. Firstly, there is the *lifetime hint*, which is included with session tokens and indicates how long they can be used for. Secondly, there is the *observed lifetime* which is the maximal time during which a server actually accepts a resumption token before it refuses session resumption. For TLS 1.2, the shortest observed lifetime hint was 5 minutes, and was given by 24% of probed websites, and 86% of websites provide lifetime hints of 24 hours or less. Figure 4.6 shows the cumulative distribution of life-time hints in TLS 1.3. There is an observable difference in lifetime hints between the two versions: roughly 70% of TLS 1.2 session tickets included a life-time hint of 2 hours or less, while 68% of

TLS 1.3 PSKs supported lifetimes of at least 2 hours, and 24% supported 24 hours or more (as opposed to 3% in TLS 1.2). Despite these higher lifetime hints in TLS 1.3, the observed lifetime turned out virtually always lower, with only 5% of websites accepting resumptions after 24 hours. After this, the observed lifetime for TLS 1.2 session ID resumption is measured, and is found to be markedly lower: 30% did not resume after more than 30 seconds, 96% did not resume after 30 minutes and not a single site resumed after more than 2 hours. This shorter session duration can be attributed to the fact that for session ID-based resumption, the server needs to keep state on each session in its cache, while session tickets contain all necessary state within them. Further, it was found that regularly resuming ID-based sessions increased their lifetime in a number of cases, likely due to least-recently-used cache clearing policies.

**Table 4.4:** Popular browser behavior related to TLS session resumption. STK refers to TLS 1.2 session ticket resumption, PSK to 1.3 PSK resumption, PSK reuse indicates if PSKs can be reused, 3rd party indicates if session resumption on third-party domains is supported, and cache clear indicates if a browser clears TLS sessions with a user-invoked its cache clear (source: [19]).

| Browser | STK | PSK | PSK Reuse | 3rd party | Cache clear |
|---|---|---|---|---|---|
| Chrome | 1 hr | > 48 hrs | N | Y | N |
| Firefox | 15 mins | 15 mins | Y | Y | Y |
| Edge | 1 hr | > 48 hrs | N | Y | N |
| Opera | 1 hr | > 48 hrs | N | Y | N |
| Brave | 1 hr | > 48 hrs | N | Y | N |

Afterwards, an evaluation of the client (i.e. browser) settings is done. The specific metrics that are evaluated include the lifetime of TLS 1.2 session tickets and 1.3 pre-shared keys, if third-party resumption works when the same third-party resource is loaded from different first-party domains, and if clearing the browser cache also clears TLS sessions. The tested browsers include Firefox, Chrome, Edge, Brave, and Opera. The results of this study are summarized in Table 4.4. Interesting observations include that Firefox is the only browser to re-use pre-shared keys at all, and that Firefox is also the only browser to clear TLS sessions when the cache is cleared. This is an interesting finding, because it is technically possible for servers to respawn cookies after a cleared cache based on session correlation, although this has never been observed in the wild.

### 4.3.2 Closing Remarks

The findings made in this study indicate that tracking through TLS session resumption is possible, albeit not very powerful, at the very least within Web context. While it remains a risk, the impact can thus be considered fairly low. This is important to keep in mind while developing sensitive IoT applications using TLS, although most devices will only communicate with first-party servers, where this type of tracking

would be unnecessary to begin with. For those that do communicate with third-party servers, we can conclude that it is safest to disable session resumption if privacy is of the utmost concern. Further, its impact can be limited to the corralation of connection IDs by adjusting the session resumption protocol to not allow opaque data as a ticket.

CHAPTER 5

# ratchet TLS: Adapting TLS 1.3 for Lightweight Devices

Based on the findings in the previous chapter, we know that TLS 1.3 session resumption does not incur significant computational or memory overhead compared to other TLS extensions, while it reduces bandwidth overhead significantly. The TLS 1.3 early-data extension additionally gives the possibility to transmit application data already in the first flight of messages coming from the client, a feature called 0 Round-Trip Time (0-RTT) resumption. It would then make sense for lightweight devices to tolerate the occasional full TLS handshake for setting up a PSK, and maximizing usage of 0-RTT session resumption to minimize handshake overhead when a PSK is available. This would bring the benefits of TLS to the (I)IoT, making it easier to satisfy to security requirements DSS-04[1], A-06[2], and A-02[3] on lightweight devices. Unfortunately, it turns out that 0-RTT session resumption in TLS 1.3 requires early application data to be idempotent, i.e., early data is not allowed to change server state in any way. This is acceptable in a Web environment where early data will usually contain HTTP GET requests, which are supposed to be idempotent in any case. However, for IIoT scenarios it is much more likely that state-changing data is transmitted, such as e.g. periodic sensor readouts. There is thus a need for a more (I)IoT friendly TLS 0-RTT resumption handshake.

In this chapter, we introduce a TLS 1.3 extension that adapts the TLS 0-RTT session resumption protocol to drop the early data idempotency requirement. Additionally, it aims to further reduce bandwidth overhead and increases the number of possible resumptions with a single PSK, to minimize the number of times a full handshake has to be performed.

This chapter discusses paper E and paper F. The former introduces the rTLS protocol, while the latter provides an update to this protocol and adds a formal

[1]Secure data transport
[2]Mutual authentication
[3]Key distribution

verification of its security properties. The chapter closes by summarizing a MSc. thesis project that implements this protocol and empirically evaluates its performance.

The content in this chapter can be considered to be in pursuit of research goal G-III, as it extends existing TLS features and introduces features that allow 0-RTT handshakes to be used in IIoT scenarios.

## 5.1 [Paper E] rTLS: Lightweight TLS Session Resumption for Constrained IoT Devices

This work introduces the ratchet TLS (rTLS) TLS 1.3 extension. This extension adds a new form of 0-RTT session resumption, which decreases the bandwidth overhead and is not susceptible to replay attacks, unlike the standard 0-RTT protocol. The protocol extension builds on the Signal Protocol [26], from which we borrow the term "ratchet".

This work motivates the need for such a protocol extension, discusses TLS session resumption, defines the protocol, and provides a numerical analysis providing an estimate for the performance increase that rTLS can bring.

### Personal Contribution

This work was a collaboration between DTU, Itron Idealabs, and UniquID. Itron Idealabs and UniquID provided the use case and original TLS performance benchmarks. The protocol was subsequently defined and developed by me, with feedback from Itron and UniquID. The numerical estimates were also done by me. Additionally, I am responsible for all text in this paper.

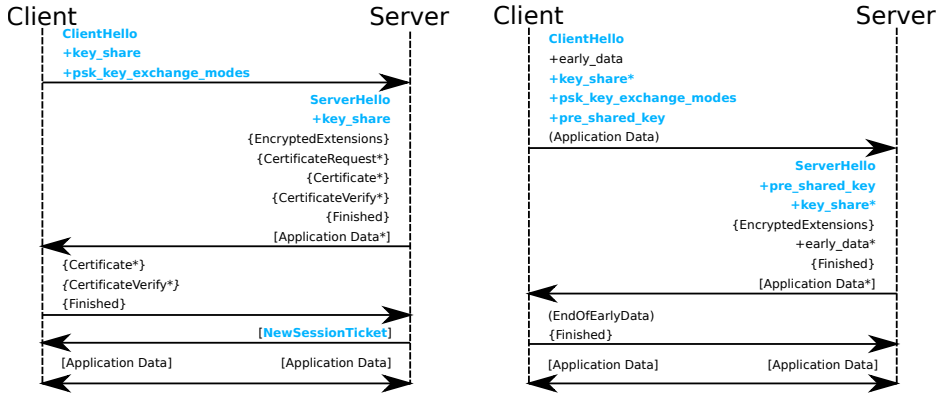
### 5.1.1 Extended Summary

First, a motivation for rTLS is given. A typical (full) TLS handshake takes anywhere from 1 to 4 KB of data traffic. To partially remedy this, the TLS 1.3 standard [23] includes a 0 Round-Trip Time (0-RTT) session resumption protocol. This allows two parties to share a PSK during the initial handshake, which can then, after the original TLS session has been closed, be used for 0-RTT session resumption. This protocol features an expedited TLS handshake with fewer round-trips, and allows the client to already transmit application data (referred to as early data) in its first flight of messages (hence "0 round-trips"). Unfortunately, this resumption protocol is not very useful for IoT applications because it does not allow for early data to change server-sided state, because it is vulnerable to replay attacks.

Another motivation is that bandwidth is deemed expensive in 5G networks and for modern IIoT devices it is thus of interest to reduce bandwidth usage as much as possible. Further, TLS is designed for the Web and assumes servers serve a potentially

infinite set of unknown clients. In (I)IoT scenarios however, the set of clients if often known a priori, and fairly static. An (I)IoT-oriented extension such as rTLS can then make the assumption that it is fine to store server-sided state in between sessions, something that is avoided in the standard TLS design.

Then, preliminaries are discussed. First, TLS 1.3 is briefly discussed, with an emphasis on the 0-RTT protocol and the `NewSessionTicket` data structure, which contains the PSK. The session ticket is generated by the server and transmitted to the client. It is assumed that this ticket contains all necessary state information for the server to resume the TLS session, and is encrypted with a key known only to the server. Figure 5.1 shows the communication pattern of both the initial handshake and a 0-RTT resumption handshake, with elements modified for rTLS in blue. Then, the Double Ratchet algorithm is discussed, this algorithm is part of the Signal Protocol [26] and enables highly secure asymmetric message exchange. An important concept in this protocol is a Key Derivation Function (KDF) chain, a feedback loop structure for KDFs. Such a structure provides key material for message encryption, with part of its output acting as input for the next iteration. This creates a ratchet-like structure (KDF chains are often referred to as "ratchets") where one can use a key to produce new keys for future messages, but cannot use it to produce previously used keys. A double ratchet combines two ratchets, called an "inner" and "outer" ratchet. The inner ratchet produces symmetric keys which are used for message encryption, while the "outer" ratchet takes input from Difie-Hellman (DH) handshakes




**(a)** The communication pattern of the initial handshake.

**(b)** The communication pattern of the resumption handshake.

**Figure 5.1:** Figure 5.1a and 5.1b depict the initial respectively resumption handshake communication patterns. + denotes an extension, * denotes an optional or situational component while {} and [] denote encryption with a derivation of the handshake or application secret, respectively. Elements that are used for rTLS are printed in blue. (source: [29]).

and provides key material which is used to reset the inner ratchet. Because DH handshakes rely on external entropy, this feeds fresh entropy into the inner ratchets thereby providing break-in protection.

Afterwards, the paper introduces the rTLS protocol. This protocol targets the following four design goals: to maximally rely on existing TLS features; to minimize the number of changes (to the original protocol); to minimize bandwidth overhead; and to provide stronger 0-RTT security properties. The `NewSessionTicket` structure is used to transmit the rTLS PSK, which includes a connection ID and a nonce. At the same time, the server initializes a ratchet which will in the future be used to decrypt early data for session resumptions with this connection ID. When the client receives the PSK, it also initializes a ratchet. Because the keys used to initialize the ratchet are derived from the shared TLS master secret, the client can then in the future encrypt early data using the ratchet output as key, knowing the server will be able to decrypt it with its own ratchet. The cryptographic primitives used for the ratches are chosen from the chosen TLS cipher suite, to ensure compatibility on both client and server.

For session resumption, the client can choose to include a (DH) `key_share` extension in the handshake, which will be used as external entropy to reset the ratchets on both client and server side when the resumption handshake is finished. Additionally, the client includes a `pre_shared_key` extension which includes the connection ID previously obtained from the server, along with its ratchet index, indicating how many times the client ratchet function has been called since it was last initialized. Finally, it includes early data encrypted with the most recently produced key from the client ratchet. When the server receives the client handshake message, it will reply with a `key_share` extension of its own if it was included by the client, and it can use the connection ID and ratchet index provided by the client to decrypt the early data. A brief summary of the extra protocol steps (on top of standard TLS operation is provided below:

**Initial Handshake**

1. **ID Generation**: The server generates a globally unique connection ID and transmits this to the client;
2. **Ratchet Initialization**: Both client and server initialize their ratchet structures;
3. **Persistent state storage** Both client and server store their state variables;

**Resumption Handshake (Client)**

1. **Ratchet step**: Executes the symmetric ratchet and derives the early data secret form this step;
2. **PSK exchange**: Transmits its connection ID and ratchet index to the server;

**Resumption Handshake (Server)**

1. **Access state**: The server finds the relevant state variables (ratchet) based on the received connection ID;

2. **Anti-replay condition**: The server ensures that its own ratchet index is less than or equal to the client's advertised ratchet index;

3. **Ratchet step**: The server executes its ratchet enough times so that its ratchet index will match the client ratchet index, with which it then derives the early data secret;

Further, a description of the necesary state variables is given: Both the client and server will need to store the connection ID and the ratchet, as well as their ratchet index. The server will need to keep a connection ID → ratchet mapping as well to identify the correct ratchet upon a resumption.

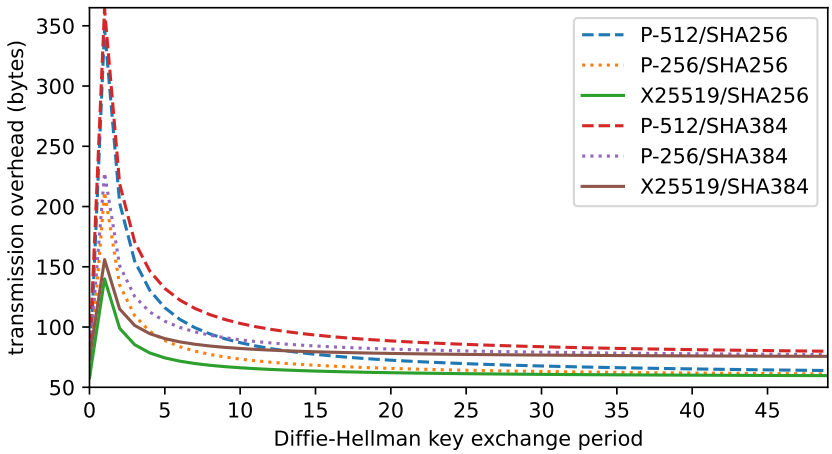



**Figure 5.2:** Average rTLS transmission overhead v. DH key exchange frequency (source: [29]).

Then, the proposed protocol extension is evaluated. This evaluation consists of two parts: first, an informal evaluation of its security properties is given, and afterwards a numerical estimate of the bandwidth and storage overhead is given based on the size of transmitted and stored data structures.

The security evaluation includes an informal argument on the various security properties that this protocol enjoys. These include replay protection and forward-secrecy, both of which are inherent properties of ratchet constructions. Additionally, it is argued that the protocol has break-in protection, a property provided by double ratchet structures.

The traffic overhead estimation is based on the size of the various transmitted data structures involved in session resumption. First, a "fixed cost" is defined for every resumption handshake, which consists of all handshake components that must be

included and are unavoidable in any resumption handshake without rigorous changes. Then, the session ticket or PSK is dissected into its individual parts, and size estimations for these based on the TLS standard documentation are given. The resumption handshake overhead can vary, depending on if a `key_share` extension is included by the client (indicative of a DH handshake), which significantly increases the data transmission cost. The exact overhead of a `key_share` extensions further depends on the chosen elliptic curve, with the X25519 curve being the most optimal choice. Because these key shares are so costly, one does not want to include them in every resumption handshake. On the other hand, they need to be included every so often to introduce new entropy into the ratchets and provide break-in protection. This is left as an implementation choice, and the paper defines this as the "DH exchange frequency". A higher DH exchange frequency implies higher transmission overhead. Figure 5.2 shows the traffic overhead plotted against this exchange frequency, with the x-axis indicating a frequency of every $x$ resumptions. From this figure, it can be seen that although the overhead is over 100 bytes when a key share is done for every resumption, it approaches roughty 60 bytes on average if a key share is included only once per 50 resumptions. A comparison with the overhead incurred from a standard TLS 1.3 handshake is also made, based on measurements done on the OpenSSL library. The estimate is that an rTLS PSK requires only roughly 11% of traffic overhead compared to a standard PSK, and that the total bandwidth cost of a handshake will reduced by half.

The storage overhead is computed with the assumption that a 4-byte connection ID is used, and that the ratchet KDF relies on the SHA-256 algorithm. It is estimated that with the assumptions made, roughly 270GB worth of state data is needed for $2^{32}$ maintained sessions – many more than needed for the vast majority of applications.

### 5.1.2 Closing Remarks

In this work the rTLS extension for TLS 1.3 was introduced. This extension changes the 0-RTT session resumption protocol to provide protection against replay attacks and to minimize bandwidth. The goal of this extension is to turn TLS into a feasible option for lightweight devices – such as sensors in sensor networks or other edge devices in industrial scenarios. With this extension, TLS is thus a step closer to being a suitable security layer candidate for devices in the IoT, IIoT and fog sphere.

The focus of this work is on the introduction and definition of rTLS, but due to time constraints within the research project, no formal security guarantees are given. Similarly, an empirical performance study of a proof-of-concept implementation did not make it into this paper. In the next two sections, these gaps are addressed. First, in paper F, an updated specification of rTLS is described and a formal security analysis is conducted on the protocol. Afterwards, a summary is given of a MSc. thesis project that implements an rTLS proof-of-concept for an empirical evaluation of its performance.

## 5.2 [Paper F] rTLS: Secure and Efficient TLS Session Resumption for the Internet of Things

This work is an extended version of Paper E, and thus starts with a repetition of most of the content first presented in that paper. As a major new contribution, a formal verification of its security properties is included. Additionally, several minor changes to the protocol are made based on lessons learned from a proof-of-concept implementation of the protocol, as well as efforts to give a formal proof of its security. The numerical estimates are also updated to a give a more accurate picture. Further, an extended version the preliminaries is given, and the presentation of the protocol is extended upon to minimize ambiguity in the specification. This paper can be considered the most recent complete version of the rTLS specification.

Because the changes made to the protocol mainly relate to minor technicalities, the main focus of this this summary is on the formal verification aspects presented in this paper, in order to avoid repetition with the summary of Paper E.

### Personal Contribution

I am the main author of the work presented in this paper, and am responsible for most of the writing, with the exception of parts of the description on the formal verification itself, which were done by S. Mödersheim, as he authored the final formal specification. The formal specification was produced in a long intensive process. The first iteration of a formal specification was presented in a DTU MSc. thesis [15] by A. Lalos (supervised by S. Mödersheim) and was the result of an intensive collaboration between A. Lalos, S. Mödersheim and myself. Afterwards, the specification was improved upon for publication in this paper together with Mödersheim. I gave regular feedback and double-checked the specification during this phase, while working on the rest of the paper.

### 5.2.1 Extended Summary

This work starts with a motivation to the one in paper E, after which it discusses the necessary preliminaries. These include an understanding of the the Double Ratchet algorithm and an understanding of the TLS 1.3 protocol. Compared to the original work, a more in-depth discussion on ratchets and double ratchets is given.

Then, the work presents the rTLS protocol itself. This presentation is largely the same as in paper E. The paper deviates slightly in terminology: what was formerly referred to as the “DH Exchange frequency” is now referred to as the “DH Exchange period” as that better captures the relation. One notable change is that now a more thorough discussion on rTLS key derivation together with how it fits into the TLS key schedule is given. Figure 5.3 depicts the TLS key schedule, with additions by the rTLS protocol marked in red. The resumption master secret together with a shared

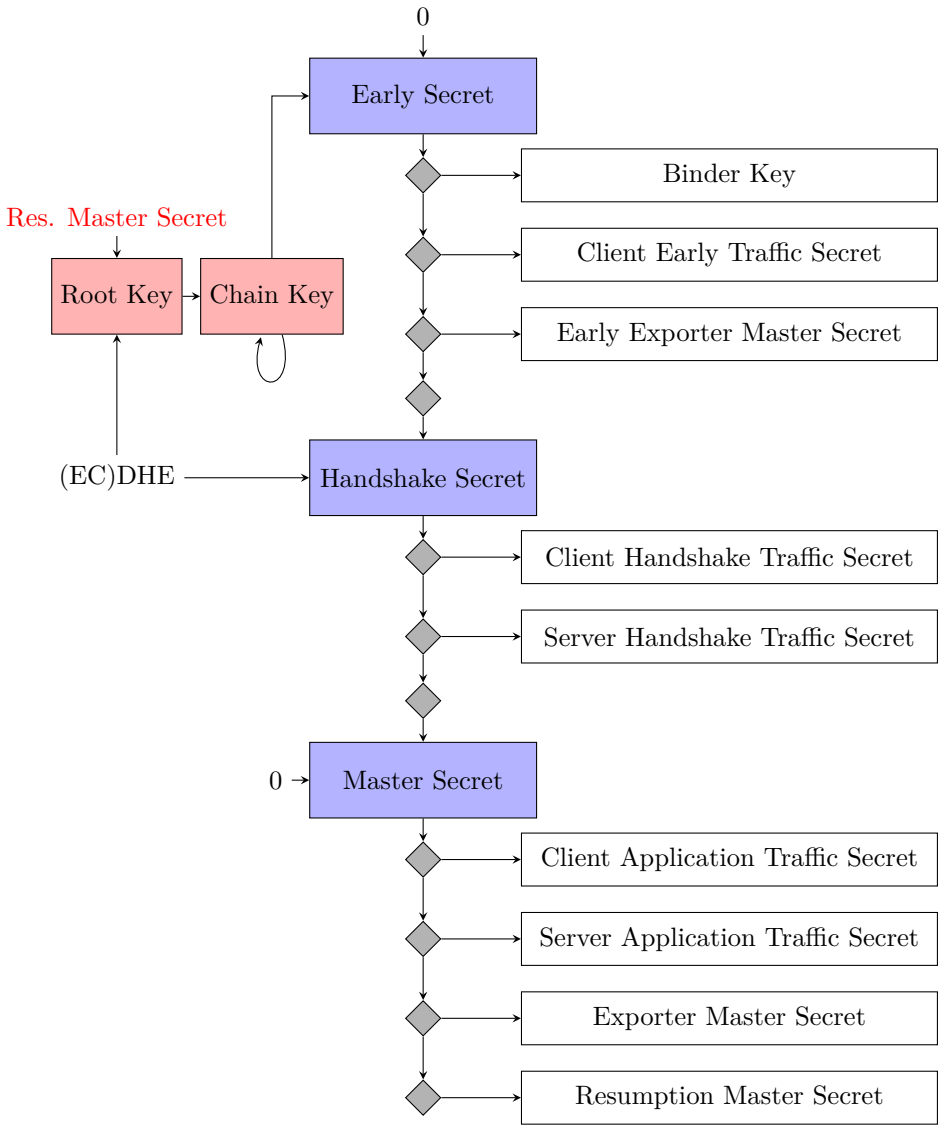



**Figure 5.3:** The rTLS Key schedule. Red indicates added KDF instances. Blue indicates a default TLS HKDF instance. Grey diamonds indicate applications of the KDF function to produce a key (source: [30]).

secret key from an elliptic curve DH exchange is used to generate a root key which acts as the initial key for the ratchet. Then, whenever the ratchet is spun a chain key is generated that acts as an input key for the early secret, upon which all other

secrets rely. This way, all derived keys for a session resumption rely on the ratchet (chain) key. Further, extra emphasis is given to the requirement that ratchet indices must be reset to 0 whenever a DH exchange occurs in a resumption.

Two new state variables have been added to the protocol. Both the client and server need to maintain a copy of their currenty private DH key, as well as the last received remote DH key. Together, these are necessary for deriving a root key when the ratchet needs to be reset.

After the presentation of the protocol, a security evaluation is given. Unlike in the earlier work, this consists of a formal specification for which several security properties are then verified in a Dolev-Yao-like intruding model. This is done using the Open Source Fixed-Point Model Checker (OFMC) tool for a bounded number of sessions, giving high certainty that they hold for the rTLS protocol. Note that the specification models resumption handshakes both with and without DH key exchanges, so that both can be verified. Because specifying rTLS in existing supported languages for OFMC turned out to be very dificult due to its stateful nature, a new notation was devised, for which a compiler is now being developed.

The formal specification starts from initial states for both the client and the server. They are initialized to share a resumption master secret, as well as a populated remote DH key for the client and a populated private DH key for the server. These are realistic assumptions to make as they can be shared during the initial handshake. Next, the specification for the resumption handshake is documented in detail, starting with the ClientHello message. Afterwards, the work similarly presents and discusses the ServerHello and Finished messages, which all taken together form the messages in a resumption handshake. For each message, only the cryptographically relevant parts are modeled, in order to keep clutter to a minimum.

The final part of the formal verification section describes the verification process itself. The first verification goal is secrecy, and the second goal is injective agreement, which means that when a honest party B receives a message from A, then A is either the intruder under their real alias or A did indeed send that message. Further, the injectivity implies that replayed messages are not accepted by B. For the verification, the number of explored sessions is bounded to 2, and the number of resumptions per session is also bounded to 2. Higher numbers were not practically feasible due to an exponential increase in the explored state space. Also, it is deemed unlikely that further sessions and resumptions would uncover further attacks due to the symmetry of all further repetitions. Additionally, it was tested that all expected steps could be taken in the model, so that the client and server were able to communicate. The OFMC tool reported no attacks, meaning that it can be stated with high certainty that the security properties hold. Finally, a brief argument is given proving that rTLS is not vulnerable to the recent selfie attack [16].

Finally, the performance evaluation has been adjusted to reflect the change in state variables. Compared to the predecessing paper, the minimum bandwidth overhead estimation remains the same, although an estimation for a more realistic handshake (i.e. including other extensions, ) has been added, and predicts roughly 400-600 bytes of overhead. The storage overhead has been adjusted upwards to 101 bytes, meaning

that 433GB of data is needed to track $2^{32}$ sessions on the server. This increase can be attributed to the addititon of new state variables.

### 5.2.2 Closing Remarks

The largest new contribution in this work is a formal security analysis of the rTLS protocol, showing that multiple security properties hold. Additionally this work improves upon the description of rTLS and provides updated performance estimates.

The security analysis adds a formal backing to the rTLS security claims, and shows that it is a feasible solution for the security problems that prohibit (I)IoT adoption of the TLS 0-RTT resumption protocol – or indeed any situation where the idempotency requirement of the standard resumption protocol is too strict. What remains, is to empirically evaluate this extension and observe the overhead introduced by it. A summary of this process is described in the next section.

## 5.3 [MSc. Thesis 3] rTLS: Proof-of-Concept and Empirical Evaluation

After the specification of the rTLS protocol extension and its security analysis, an empirical study was still needed. As this formed a well-defined research problem by itself, it was turned into a MSc. thesis project and was conducted by C. Xenofontos [34].

In this work, the proof-of-concept implementation of rTLS is presented, as well as a performance evaluation of the extension.

#### Personal Contribution

The MSc. thesis is an original work conducted by C. Xenofontos [34], and supervised by X. Fafoutis and myself. Since this project was also very related to my PhD, I was closely involved in its execution. My role as a supervisor included defining the original problem description, regular supervision meetings, guidance and advice on the workings of TLS and the double ratchet algorithm, proofreading, assistance with defining the performance metrics and guidance on the workings of rTLS.

### 5.3.1 Extended Summary

The work starts with an introduction of TLS and a motivation for rTLS, specifically aimed at the need for a proof-of-concept implementation to verify the performance claims. Then, the preliminaries are discussed. These include the TLS 1.3 protocol along with several attacks on the protcol that have been published over the years, the

0-RTT resumption protocol, the double ratchet algorithm, and the rTLS extension itself. After the preliminaries, the related work is discussed, including other lightweight TLS variants such as Datagram TLS (DTLS) and QUIC.

Then, the implementation of rTLS is described. First, it is established that due to the complexity of TLS, it makes more sense to extend an existing TLS library rather than implement a new TLS library from scratch. A brief comparison of various libraries is then conducted, with WolfSSL [33] being chosen as the target library due to its clean code, ample documentation, and great community support compared to the other contenders. Subsequently, a description of WolfSSL is given, introducing the various components that make up the library, as well as the source files that are relevant for the remainder of the thesis. Further, it is stressed that the version of rTLS developed in this thesis is purely for scientific analysis and should not be considered feature-complete or secure. A description of the working environment and tools used during development is also given.

The main goal of this implementation is to measure bandwidth overhead, meaning no specific measures are taken to optimize memory and computational overhead. The thesis describes the changes made to the initial handshake, i.e., the addition of a new PSK key exchange mode, the generation of a connection ID and its transmission to the client, and the initialization of the rTLS state variables for both client and server. To store these state variables, the rTLS code extends the `ssl` structure used by WolfSSL to maintain session context. Further, the wolfCrypt library is relied upon for cryptographic (KDF) calls. This alleviates some complexity as it provides us with an optimized and secure alternative to implementing it from scratch. Because the `ssl` structure is cleared when a session is closed, special functionality is added that can persistently store the rTLS state variables when this happens.

The session resumption implementaton is also described in detail. One deviation from the rTLS specification in papers E and F is that the optional DH key exchange is appended to the `pre_shared_key` structure instead of the `key_share` structure. This was decided upon because it turned out that adding the `key_share` structure to the resumption message added unnecessary bandwidth overhead that could be prevented by incorporating the DH exchange into the PSK field. This change can be included in the specification in future publications.

Afterwards, a number of changes to the source code of the example WolfSSL client and server implementations were made specifically to enable the testing of multiple session resumptions (both with and without DH exchanges). To capture the initial and resumption handshakes, a set of benchmarking scripts were written, and additionally the results were manually verified with Wireshark. The cumulative bandwidth cost of the “ClientHello”, “ServerHello”, “Encrypted Extensions”, “Finished” (both for client and server), and the “End of Early Data” messages is measured. For each possible DH key exchange period (up to 50), 255 session resumptions are recorded, after which the average bandwidth cost is computed. The results of these tests are shown in Figure 5.4. When a DH key exchange is included for every handshake, an overhead of 556 bytes is measured. As the DH key exchange period increases, this overhead reduces to 524.75 bytes on average for one DH key exchange per 50 resump-

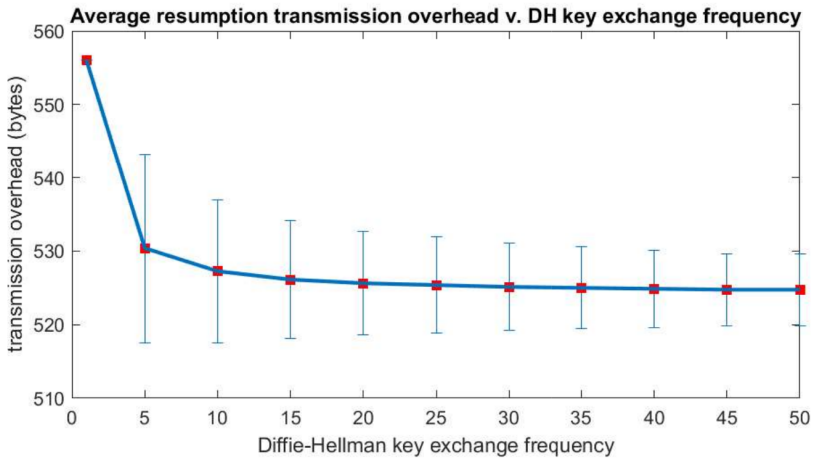



**Figure 5.4:** The average total bandwidth overhead of rTLS resumption handshakes (source: [34]).

tions. The pattern looks very similar to the estimations made in paper F, and indeed the measured overhead falls in the predicted range of 400-600 bytes. The standard TLS resumption bandwidth overhead (measured identically to the rTLS overhead described above) is also measured, and sits at a constant 746 bytes. The use of rTLS can thus save on average around 221 bytes of bandwidth overhead per session resumption, if a DH key exchange period of 50 is chosen. After comparing rTLS to standard TLS, the thesis further compares the obtained measurements to the estimations made in paper E, and points out some inaccuracies in this estimation based on the practical experience obtained from working with the WolfSSL implementation.

Finally, a discussion on rTLS, the implementation process, and future work is presented. These include developing a fully functional prototype, as well as optimizing for memory and computational overhead.

### 5.3.2 Closing Remarks

This work provided a promising first empirical validation of the estimates given in papers E and F, through a proof-of-concept implementation of rTLS on top of the WolfSSL TLS library. Together with the other work presented in this chapter, this forms a multifaceted validation of rTLS, bolstering trust in its security and performance benefits.

These results pave the way for future research on rTLS, including a prototype implementation as alluded to in the future work section of this MSc. thesis, but also other avenues can be explored. We will briefly discuss these in the following chapter.

# CHAPTER 6

# Conclusion

With the industrial sector transitioning into the Industry 4.0 era, previously disconnected industrial systems will find themselves exposed to the Internet. This brings many new opportunities, but with it also come new risks. Malicious actors might try to compromise these systems, and dependencies on third parties characteristic to Cloud computing can put latency, safety, and availabilty requirements into jeopardy. The recent Fog computing paradigm decentralizes Cloud aspects and brings them closer to the network edge, and the introduction of Fog nodes brings the power and flexibility from IT to the IIoT and OT domains. To enable a safe and secure Fog ecosystem, research into Fog computing security is needed. In this thesis, we collected IIoT security requirements identified by the scientific literature, to get an overview of the security properties that Fog nodes will need to fullfil. Further, we proposed a distributed PTP node architecture that protects PTP networks from compromised subsystems, and synergizes with Fog node capabilities. As the final contribution we introduced, analyzed, and tested rTLS, a novel TLS extension that enables the IIoT to safely use fast TLS session resumption, lowering the resource usage necessary to use the protocol. These contributions together form a necessary step towards a more secure Fog ecosystem.

In the remainder of this chapter, we will summarize the contributions of each paper and thesis that has been discussed in this dissertation, after which we close the chapter with a discussion on future work and research opportunities.

## 6.1 Contributions

The contributions collected in this thesis aim to improve the security of Fog and IIoT systems. Because the Fog computing concept spans a vast spectrum – from lightweight embedded devices to large Cloud-like datacenters, it is very challenging to present a single security solution that encompasses all these aspects. Instead, the works in this thesis focus on concrete improvements and adjustments to existing technologies, to make them more suitable for Fog and IIoT systems. This was done using the security requirements collected in paper A as an indication of where the security needs lie, and subsequently investigating several technologies that hold potential to address these issues. The outcome of this research can be grouped into three main contributions, which are summarized below.

I Papers A and B present a large systematic survey of security requirements identified by the academic literature in the IIoT, Industry 4.0, and Fog fields. These requirements are categorized by overarching themes such as authentication and access control. Further, the work presents a quantitative overview of IIoT and Fog security research in the past decade. Finally, the work includes a discussion of Fog computing opportunities towards satisfying the identified security requirements.

II Paper C describes a deep study into the security of the PTP protocol, and introduces a redundant multi-domain PTP node architecture protecting PTP networks against compromised subsystems. This contribution is complementary to the security additions made in the 2019 iteration of the PTP standard. The architecture includes convergence algorithms for aggregating offsets collected from the various domains into one reliable offset that can be used to correct its RTC. Further, its performance is evaluated in a set of simulated scenarios with various network topologies. Paper D fits partly into this contribution by describing a use case where PTP-enabled devices connect with a Fog node (through TSN).

III Papers E and F introduce the rTLS protocol extension. The former defines the rTLS protocol, while the latter includes minor improvements to the protocol itself and a formal security analysis. This extension modifies TLS 1.3 0-RTT session resumption, removing the idempotency requirement on early data sent with resumption handshakes. This makes the 0-RTT handshake suitable for IIoT applications. Further, the extension requires fewer resources than a standard resumption handshake. The extension is empirically evaluated in MSc. thesis 3. The privacy impact of TLS resumption handshakes is analyzed in Msc. thesis 2, and a performance measurement of a collection of common TLS extensions is given in MSc. thesis 1. This provides a more comprehensive view of the suitability of TLS for lightweight scenarios, while motivating the need rTLS as a less resource-intensive alternative to standard session resumption. Paper D also partially fits into this contribution, as in includes a security analysis of the use case that describes a need for TLS.

## 6.2 Future work

There are many open research questions and opportunities for future work in the IIoT and Fog security domain. Contribution I discusses many more opportunities than can possibly be addressed in one thesis. These remain subjects for potential future work. A few examples of this include the use of Fog nodes to extend traditional Public Key Infrastructure (PKI), or as multi-factor authentication hubs for e.g. authenticating configuration changes when equipped with smart card readers or biometric sensors. Further, Fog nodes could serve as a decentralized AC infrastructure, removing the

single point of failure in traditional centralized AC policy systems. Due to their highly virtualized nature, Fog nodes also hold potential to run software or configuration upgrades for peripheral devices in sandboxed environments, while monitoring for anomalous behavior and dynamically rolling back if operational requirements are violated. The interested reader is referred to paper A for a comprehensive discussion on research opportunties for Fog security in general. In the remainder of section, we will focus on future work related to Contributions II and III.

Contribution II marks a first step in redundant PTP node design, and includes testing of a fully redundant PTP setup. However, the resource investment needed for this architecture is not trivial. It is possible that a hybrid setup where only parts of the PTP network are redundant offers similar benefits for a smaller investment. Further, there are some attack categories that might not be mitigated even with the combined security features of the 2019 PTP standard and a redundant node architecture, and future research is needed to identify and protect against these. Finally, our proposed architecture is for PTP slave nodes, but there are other node types such as boundary clocks that can also benefit from a distributed architecture. This poses an opportunity for future work in this domain.

Although the rTLS protocol extension is already empirically validated in MSc. thesis 3, these tests were conducted with a limited proof-of-concept implementation on an x86 platform. Further research is needed to accurately capture the performance of this extension on lightweight devices and other architectures (e.g. ARM and RISC-V devices). There is an ongoing effort to port the proof-of-concept implementation to embedded platforms and conduct performance tests on these, which will enable a more accurate evaluation of the resource requirements for this extension. Additionally, the bandwidth overhead can be reduced even more by replacing the existing resumption message structure altogether. After the original introduction of rTLS , the DTLS 1.3 standard has been published, featuring a similar resumption protocol to that of TLS 1.3. Porting the rTLS extension to DTLS would remove the dependency on TCP, lowering resource requirements and potentially increasing performance. Finally, there are other efforts optimizing TLS for IIoT applications, such as compact TLS (cTLS) [24], presenting research opportunities to adapt rTLS for use in these projects.

The contributions presented in this thesis provide concrete improvements to existing technologies, preparing them for the Industry 4.0 era. Although the state of the art is getting closer, the road to widespread adoption of Industry 4.0 paradigms is still full of open challenges, and many more steps like the ones in this thesis are needed to make it there.

# Bibliography


[1] M. Barzegaran, N. Desai, J. Qian, K. Tange, B. Zarrin, P. Pop, and J. Kuusela. "Fogification of electric drives: An industrial use case." In: *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 2020. DOI: 10.1109/ETFA46521.2020.9212010.

[2] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli. "Fog Computing and Its Role in the Internet of Things." In: *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*. MCC '12. ACM, 2012. ISBN: 978-1-4503-1519-7. DOI: 10.1145/2342509.2342513.

[3] OpenFog Consortium. *OpenFog Reference Architecture for Fog Computing*. 2017. URL: https://iiconsortium.org/pdf/OpenFog_Reference_Architecture_2_09_17.pdf (visited on December 24, 2021).

[4] Danfoss. *Danfoss Electric Drives*. URL: https://www.danfoss.com/en/products/dds/ (visited on December 24, 2021).

[5] M. De Donno, K. Tange, and N. Dragoni. "Foundations and Evolution of Modern Computing Paradigms: Cloud, IoT, Edge, and Fog." In: *IEEE Access* (2019). DOI: 10.1109/ACCESS.2019.2947652.

[6] D. Eastlake 3rd. *The Transport Layer Security (TLS) Protocol Version 1.3*. 2011. DOI: 10.17487/RFC6066. URL: https://rfc-editor.org/rfc/rfc6066.txt (visited on December 24, 2021).

[7] P. H Feiler, D. P Gluch, and J. J Hudak. *The architecture analysis & design language (AADL): An introduction*. Technical report CMU/SEI-2006-TN-011. Carnegie-Mellon Univ Pittsburgh PA Software Engineering Inst, 2006.

[8] IEC. *Functional safety of electrical/electronic/programmable electronic safety-related systems - Part 1: General requirements*. 2010.

[9] *IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*. eng. 2008. DOI: 10.1109/IEEESTD.2008.4579760.

[10] *IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*. eng. 2020. DOI: 10.1109/IEEESTD.2020.9120376.

[11] Moz Inc. *Top 500 Most Popular Websites*. URL: https://moz.com/top500 (visited on December 24, 2021).

[12] R. Kaiser and S. Wagner. *The PikeOS concept: History and design*. Technical report. SYSGO AG, 2007. URL: https://www.sysgo.com/.

[13] B. Kitchenham and S. Charters. *Guidelines for performing Systematic Literature Reviews in Software Engineering*. Technical report EBSE-2007-01. EBSE Technical Report, 2007.

[14] E. Kyriakakis, K. Tange, N. Reusch, E. O. Zaballa, X. Fafoutis, M. Schoeberl, and N Dragoni. "Fault-tolerant Clock Synchronization using Precise Time Protocol Multi-Domain Aggregation." In: *2021 IEEE 24th International Symposium on Real-Time Distributed Computing (ISORC)*. IEEE, 2021. DOI: 10.1109/ISORC52013.2021.00025.

[15] A. Lalos. "A Formal Library of IoT Protocols." 2021. URL: https://findit.dtu.dk/en/catalog/2685752487 (visited on December 25, 2021).

[16] G. Lowe. "Selfie: reflections on TLS 1.3 with PSK." In: *Journal of Cryptology* 34.27 (2021). DOI: 10.1007/s00145-021-09387-y.

[17] A. K. Mathiasen and E. Bejder. "TLS Extension Performance Impact." 2021. URL: https://findit.dtu.dk/en/catalog/2692890744 (visited on December 24, 2021).

[18] Y. Nir, R. Salz, and N. Sullivan. *Transport Layer Security (TLS) Extensions*. 2021. URL: https://www.iana.org/assignments/tls-extensiontype-values/tls-extensiontype-values.xhtml (visited on December 24, 2021).

[19] J. Pecl. "A Practical Study on Online Tracking Using TLS Session Resumption." 2021. URL: https://findit.dtu.dk/en/catalog/2691679817 (visited on December 24, 2021).

[20] K. Petersen, S. Vakkalanka, and L. Kuzniarz. "Guidelines for conducting systematic mapping studies in software engineering: An update." In: *Inf. Softw. Technol.* (2015). DOI: 10.1016/j.infsof.2015.03.007.

[21] Y. Pettersen. *The Transport Layer Security (TLS) Protocol Version 1.3*. 2013. DOI: 10.17487/RFC6961. URL: https://rfc-editor.org/rfc/rfc6961.txt (visited on December 24, 2021).

[22] *Profinet system description–system manual*. Technical report Issue A5E00298288-04. Siemens Simatic, 2008.

[23] E. Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.3*. August 2018. DOI: 10.17487/RFC8446. URL: https://rfc-editor.org/rfc/rfc8446.txt (visited on December 24, 2021).

[24] E. Rescorla, R. Barnes, and H. Tschofenig. *Compact TLS 1.3 (IETF draft)*. URL: https://datatracker.ietf.org/doc/draft-rescorla-tls-ctls/.

[25] E. Sy, C. Burkert, H. Federrath, and M. Fischer. "Tracking Users across the Web via TLS Session Resumption." In: *Proceedings of the 34th Annual Computer Security Applications Conference*. ACSAC '18. Association for Computing Machinery, 2018. DOI: 10.1145/3274694.3274708.

[26] OpenWhisper Systems. *Signal*. URL: https://www.signal.org (visited on December 24, 2021).

[27] K. Tange, M. De Donno, X. Fafoutis, and N. Dragoni. "A Systematic Survey of Industrial Internet of Things Security: Requirements and Fog Computing Opportunities." In: *IEEE Communications Surveys Tutorials* (2020). DOI: 10.1109/COMST.2020.3011208.

[28] K. Tange, M. De Donno, X. Fafoutis, and N. Dragoni. "Towards a Systematic Survey of Industrial IoT Security Requirements: Research Method and Quantitative Analysis." In: *Proceedings of the Workshop on Fog Computing and the IoT*. IoT-Fog '19. ACM, 2019. DOI: 10.1145/3313150.3313228.

[29] K. Tange, D. Howard, T. Shanahan, S. Pepe, X. Fafoutis, and N. Dragoni. "rTLS: Lightweight TLS Session Resumption for Constrained IoT Devices." English. In: *Proceedings of the 22nd International Conference on Information and Communications Security*. Springer, 2020. DOI: 10.1007/978-3-030-61078-4_14.

[30] K. Tange, S. A Mödersheim, A Lalos, X. Fafoutis, and N. Dragoni. "rTLS: Secure and Efficient TLS Session Resumption for the Internet of Things." In: *Sensors* (2021). DOI: 10.3390/s21196524.

[31] A. Varga and R. Hornig. "An Overview of the OMNeT++ Simulation Environment." In: *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*. Simutools '08. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008. DOI: 10.4108/ICST.SIMUTOOLS2008.3027.

[32] W. Wallner. *LibPTP: A Library for PTP Simulation*. 2016. URL: https://github.com/ptp-sim/libPTP (visited on December 24, 2021).

[33] WolfSSL. *WolfSSL Embedded SSL/TLS Library*. URL: https://www.wolfssl.com/.

[34] C. Xenofontos. "rTLS: Proof-of-Concept and Empirical Evaluation." 2021. URL: https://findit.dtu.dk/en/catalog/2692305257 (visited on December 25, 2021).

Paper A

# Towards a Systematic Survey of Industrial IoT Security Requirements: Research Method and Quantitative Analysis

K. Tange, M. De Donno, X. Fafoutis, and N. Dragoni. "Towards a Systematic Survey of Industrial IoT Security Requirements: Research Method and Quantitative Analysis." In: *Proceedings of the Workshop on Fog Computing and the IoT*. IoT-Fog '19. ACM, 2019. DOI: 10.1145/3313150.3313228

# Towards a Systematic Survey of Industrial IoT Security Requirements: Research Method and Quantitative Analysis

Koen Tange, Michele De Donno, Xenofon Fafoutis
kpta@dtu.dk,mido@dtu.dk,xefa@dtu.dk
Technical University of Denmark

Nicola Dragoni
ndra@dtu.dk
Technical University of Denmark and
AASS, Örebro University

## ABSTRACT

Industry 4.0 and, in particular, Industrial Internet of Things (IIoT) represent two of the major automation and data exchange trends of the $21^{st}$ century, driving a steady increase in the number of smart embedded devices used by industrial applications. However, IoT devices suffer from numerous security flaws, resulting in a number of large scale cyber-attacks. In this light, Fog computing, a relatively new paradigm born from the necessity of bridging the gap between Cloud computing and IoT, can be used as a security solution for the IIoT. To achieve this, the first step is to clearly identify the security requirements of the IIoT that can be subsequently used to design security solutions based on Fog computing. With this in mind, our paper represents a preliminary work towards a systematic literature review of IIoT security requirements. We focus on two key steps of the review: (1) the research method that will be used in the systematic work and (2) a quantitative analysis of the results produced by the study selection process. This lays the necessary foundations to enable the use of Fog computing as a security solution for the IIoT.

## CCS CONCEPTS

• **Security and privacy**; • **Computer systems organization** → **Embedded and cyber-physical systems**; Real-time systems; Real-time system architecture; • **General and reference** → *Surveys and overviews*;

## KEYWORDS

Industrial Internet of Things, IIoT, Industry 4.0, Security, Fog Computing, Systematic Literature Review

**ACM Reference Format:**
Koen Tange, Michele De Donno, Xenofon Fafoutis and Nicola Dragoni. 2019. Towards a Systematic Survey of Industrial IoT Security Requirements: Research Method and Quantitative Analysis. In *Workshop on Fog Computing and the IoT (IoT-Fog '19), April 15–18, 2019, Montreal, QC, Canada.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3313150.3313228


*IoT-Fog '19, April 15–18, 2019, Montreal, QC, Canada*

ACM ISBN 978-1-4503-6698-4/19/04…$15.00
https://doi.org/10.1145/3313150.3313228

## 1 INTRODUCTION

Today, we are living in the $4^{th}$ industrial revolution, also referred to as Industry 4.0. Due to the increasing availability, affordability, and proficiency of sensors, processors, and Wireless Sensor Network (WSN) technologies, the number of embedded devices used in industrial applications is steadily increasing. This leads to a growth in the interest for the Industrial Internet of Things (IIoT), a large network of devices, systems, and applications communicating and sharing intelligence with each other, the external environment, and with humans [30]. According to Accenture [30], the IIoT could be worth 7.1 trillion US dollars to the United States and more than 1.2 trillion to Europe by 2030.

In this wave of excitement, IoT security represents one of the biggest weak points holding back the adoption of the IIoT. As a matter of fact, IoT devices are often poorly secured [34] and thus easy targets for malware taking advantage of them to run devastating cyber attacks, such as Distributed Denial of Service (DDoS) [31] (e.g., Mirai [32]) or sabotage attacks (e.g., StuxNet [66], CrashOverride/Industroyer [68]).

In this scenario, a relatively new computing paradigm has attracted attention: Fog computing [18]. Fog computing is a system-level architecture born from the necessity of bridging the gap between IoT and Cloud computing, by distributing resources and services along the continuum from Cloud to IoT [96]. Among others, one of the promises of Fog computing is to present a possible solution to the IoT security problem.

The first step for improving security of the IIoT is to clearly define its main security requirements. To the best of our knowledge, the last surveys discussing security requirements of the IIoT date back to 2015 and 2016 [102, 103]. However, as we show later in this paper, the field has grown exponentially since then. Thus, we believe that a systematic and up-to-date survey on the security requirements of IIoT is becoming a necessity.

### 1.1 Contribution of the Paper

In this paper, we present a preliminary study towards a systematic literature review work that aims at identifying security requirements of the IIoT.

Systematic studies are meant to give an overview of a research area, following a structured methodology with respect to searching and study selection [98]. An essential part of a systematic literature review consists of defining the research method adopted to select relevant studies that are later used to extract qualitative results on the topic. In the paper, we focus on this methodological phase of the systematical literature review and we provide a quantitative analysis of the output produced by the research so far. Thus, the

paper can be considered as the first step towards a complete systematic literature review work, in which the selected papers will be used to extract qualitative results about security requirements in the IIoT. Once the requirements are delineated, it will be possible to focus on how Fog computing can meet them.

### 1.2 Outline of the Paper

The paper is organized as follows. Section 2 briefly mentions related work and motivates the need for a systematic review. Section 3 describes the research method used. Section 4 presents a quantitative analysis of the results obtained during the research phase. Section 5 concludes the paper.

## 2 RELATED WORK

To the best of our knowledge, the most recent work focused on reviewing IIoT security is [74], where the focus lies on threat characterization by looking at existing attacks. However, this work does not explicitly discuss security requirements, opting to leave them as implied by the described threats.

Another recent study [46] focuses on Industry 4.0 system architecture as a whole and observes that there is an increase in security-focused architectural proposals, but does not discuss security in depth.

Some older surveys dated back to 2015 and 2016 also mention IIoT security requirements [102, 103], but they refrain from discussing such requirements in-depth.

## 3 RESEARCH METHOD

In this section, we present the research method that will be used in the systematic literature review on security requirements for the IIoT that will extend this work.

We adopt the research method detailed by Petersen et al. [98], and utilize the suggested template for describing our approach. In the next subsections, we elaborate on research questions, search strategy, study selection, and validity concerns.

### 3.1 Research Questions

The main aim of this work is to identify security requirements for the IIoT. Our end goal is to investigate which ones can be solved by Fog computing. In addition, we want to provide an overview of the research activity in the field: how research activity has developed throughout the years, how this research was published, and what its geographical distribution is.

Thus, our research questions can be formulated as follows:

- **RQ1:** how are publications related to IIoT security spread throughout the years?
- **RQ2:** how is IIoT security research activity geographically distributed?
- **RQ3:** what are the most popular publication venues for IIoT security research?
- **RQ4:** what are the security requirements of the IIoT?
- **RQ5:** which of these security requirements can be solved by Fog computing?

Note that RQ4 and RQ5 are questions we aim to answer in our completed study, so they are not discussed in this preliminary work.

Answering these questions will aid in getting a better understanding of the current security landscape for the IIoT, while at the same time identifying various concrete research opportunities related to security for Fog computing. Each of these can then be traced back to concrete security requirements relevant to the Industry 4.0 paradigm.

### 3.2 Search Strategy

We utilize the adjusted PICOC criteria for software engineering [60] in order to identify relevant keywords. In particular:

- **Population:** we consider the IIoT as the application area in which our research is conducted. However, this is a very broad population, therefore, we take into account only studies addressing IIoT security.
- **Intervention:** this criterion does not apply to our research questions, as we are interested in *any* work in the IIoT domain that describes security requirements.
- **Comparison:** we compare the security requirements identified by different studies by taking into account such factors as the number of studies that mention them, related threats, and proposed solutions.
- **Outcomes:** we present the identified security requirements as well as the properties of their mitigation, allowing us to discuss which requirements call for further research.
- **Context:** As we do not empirically compare the available works, this criterion does not apply to our study.

With these criteria in mind, we have formulated the following keywords: *IIoT, Industrial Internet of Things, Industry 4.0,* and *security*.

We considered as sources the following databases: ACM Digital Library, IEEE Xplore, Elsevier/ScienceDirect. In this domain, we believe that the combination of these three sources provides an accurate representation of the research that has been conducted globally.

We divided the search into two stages. First, we queried the databases for articles related to IIoT/Industry 4.0 in general, based on their titles. This provided an overview of the amount of research conducted in this field. After that, we narrowed down our search to only include works related to security, by excluding articles not containing the word "security" in their abstract. The queries are summarized in Table 1. The search results for both queries are listed in Table 2.

### 3.3 Study Selection

The study selection process was done in multiple phases. Firstly, the JabRef [1] reference management software was used to identify and delete duplicates. Two duplicates were found, leaving the number of considered papers for the subsequent phases at 173.

In the second phase, we independently reviewed titles and abstracts of each article in order to reduce selection bias. Each article

[1] https://www.jabref.org

**Table 1: Queries used for our search, expressed in pseudo-code**

| Query | Description |
|---|---|
| Q1 | *in title:* IIoT OR "Industrial Internet of Things" OR "Industry 4.0" |
| Q2 | (*in title:* IIoT OR "Industrial Internet of Things" OR "Industry 4.0") AND *in abstract:* security |

**Table 2: Number of papers returned from our queries**

| Source | Q1 | Q2 |
|---|---|---|
| ACM | 36 | 6 |
| IEEE Xplore | 1462 | 160 |
| Scopus | 219 | 9 |
| **Total** | 1717 | 175 |

was marked as being relevant, not relevant, or of doubtful relevance. Articles were voted for inclusion when the work covers cyber-security challenges and/or solutions for Industry 4.0, and it was published before 2019, since that is the year in which this study is conducted. Articles were voted for exclusion when the work was not related to Industry 4.0 security, a duplicate, or was not presented in legible English.

The following rules were used for filtering out articles based on title and abstract review (this has been done jointly by two authors of the paper):

- when both authors considered an article relevant, the article was included for the next phase;
- when one author expressed doubt and the other author considered an article relevant, the article was included for the next phase;
- when both authors expressed doubt, a joint review was done considering also other sections of the article (e.g. introduction, outline, conclusion) in order to determine its relevance. If this review did not clear up doubts for either of the authors, the article was given the benefit of the doubt and included for the next phase;
- when one author considered an article relevant, while the other considered it to not be relevant, the article was marked for joint review as described in the previous rule;
- when one author considered an article not relevant, while the other considered it to be doubtful, the article was marked for joint review as with the previous rules;
- when both authors considered an article not relevant, the article was excluded.

After the individual title and abstract reviews, 35 articles were excluded and 41 were marked as doubtful entries requiring a joint review. These were then jointly reviewed, leading to an additional 18 exclusions. The remaining 120 papers ( [1–9, 11–17, 19–29, 33, 35–53, 55–59, 61–65, 67, 69–73, 75–95, 97, 99–102, 104–134]) were considered for full-text reading, overall reducing the number of papers to analyse by 93% compared to results of Q1 and 30% compared to Q2.

The next phase, consisting of reading the full text of each selected paper, is currently in progress. It is already clear that some articles are not relevant and will be excluded but, at present, we are unable to provide relevant numbers on this. During the full-text reading phase, we extract information relevant to the stated research questions, and use this to create a comprehensive picture of the security challenges and corresponding requirements for the IIoT.

### 3.4 Validity Evaluation

Every study that is subject to manual selection is vulnerable to researcher bias in the filtering process. In order to reduce this issue, we performed the filtering process twice: two authors of this paper selected studies independently, and the results of the filtering process were based on a systematic approach combining the selections of both authors, and in some cases a joint review.

Furthermore, we have described our research process in detail, and have taken care to list the criteria by which we have filtered studies. This is done to increase the repeatability of this work.

Finally, it is worth mentioning that our approach does not suffer from the Matthew's effect, as opposed to querying databases that rank papers based on citation count [10].

## 4 RESULTS

In this section, we provide a quantitative analysis of the set of studies resulting from the presented research method.

### 4.1 Spread of publications throughout the years (RQ1)

Figure 1 shows the number of publications between 2013 and 2018. Security research for the IIoT starts first appearing around 2013, growing slowly over the next 3 years. In 2017, a drastic increase in activity can be seen. One possible reason is that 2016 saw several serious IoT related security incidents (such as Mirai [32] and Crashoverride/Industroyer [68]), which served to illustrate the importance of security on these devices. In 2018, the growth in activity

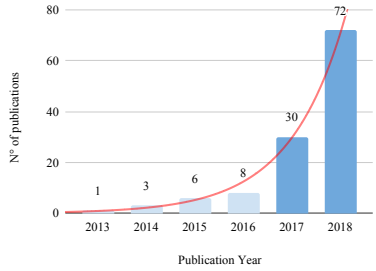



**Figure 1: Number of publications per year**

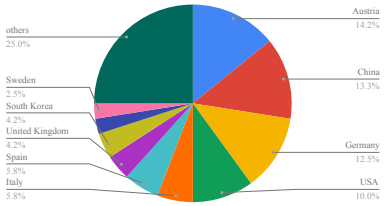



**Figure 2: Demographic: geographical distribution of research activity based on first author country of affiliation**

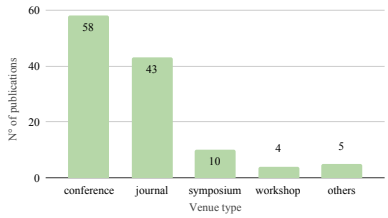



**Figure 3: Popularity of different venue types**

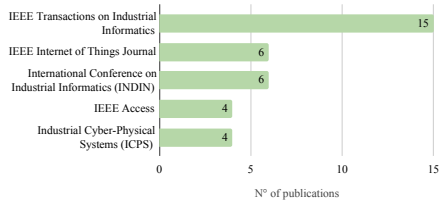



**Figure 4: Popularity of different specific publication venues**

continued, showing that the research community deems IIoT security to be of high importance.

## 4.2 Geographical Distribution of IIoT Security Research (RQ2)

The geographical distribution of research activity is shown in Figure 2. Data was obtained by extracting the country of affiliation of the first author of the considered studies.

German-speaking countries are strongly represented, making for a total of 26.7% of contributions. One possible explanation is that one of our search terms, *Industry 4.0*, was originally coined by the German government [54], thus, it might have seen higher adoption in German-speaking countries.

This raises the question of whether our search terms were successful in providing a good global sample of studies in this field. We believe they were, since the field we are considering is very narrow; we specifically searched for *Industrial* challenges in order to be able to extract security requirements unique to this field. However, we acknowledge that this might be a threat to the theoretical validity of our contribution that should be further investigated. We plan to address this issue in our future work, as stated in Section 5.

China and United States of America are the two other major contributors. This can possibly be attributed to the size of their industries and thus the relevance of research in this area. However, interestingly, 62.5% of the studies originate from Europe, showing that this topic is also regarded as highly relevant in countries with smaller industries.

The 'others' group consists of the 30 countries that have 2 or fewer publications in this field: France, Portugal, Czech Republic, Brazil, Australia, Greece, Belgium, Singapore, Ireland, Pakistan, Japan, Qatar, Turkey, Malaysia, Ukraine, Taiwan, Netherlands, Canada, Hungary, New Zealand, and Iran.

## 4.3 Venue Types for Publication (RQ3)

We have grouped the studies based on the venue type of their publication, which is shown in Figure 3. As can be seen, conference proceedings are the most popular dissemination method, followed by journals. The 'others' category consists of venue types in which 2 or fewer publications were published: congresses, summits, and forums.

Looking at the specific venues of publication (Figure 4), it can be seen that the IEEE Transactions on Industrial Informatics journal is by far the most popular venue, with 15 publications. One noteworthy observation here is that, out of all considered studies, only 5 were published in venues that were focused on security. The vast majority of IIoT security-related work appears to be published in venues targeting industrial systems or IoT instead.

## 5 CONCLUSION

In this preliminary study, we have described a systematic search and filtering of IIoT security studies, and laid the groundwork for extracting security requirements and putting them in a Fog computing perspective (RQ4 and RQ5). We also answered a number of questions about the IIoT security research domain itself, adding perspective to developments in this field. Of course, as in any mapping study, it is challenging to take all studies of the field into account, but it is more important to have a good representation of studies rather than a high number of studies [98].

Future work will be based on two phases. First, we will further improve the study selection by means of reverse snowball sampling. This will ensure that we end up with a good sample of relevant studies, mitigating bias that might have been introduced by the search terms. Second, we plan to address the remaining research questions, and provide a content review of the selected studies. We will use the extracted research requirements to discuss what research opportunities might exist within this field, as well as discussing the role that can be played by Fog computing as a security solution for the IIoT.

## ACKNOWLEDGMENTS

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 764785, FORA – Fog computing for Robotics and Industrial Automation.

## REFERENCES

[1] M. Aazam, S. Zeadally, and K. A. Harras. 2018. Deploying Fog Computing in Industrial Internet of Things and Industry 4.0. *IEEE Transactions on Industrial Informatics* 14, 10 (2018), 4674–4682. https://doi.org/10.1109/TII.2018.2855198

[2] D. Airehrour, J. Gutierrez, and S. K. Ray. 2016. Securing RPL routing protocol from blackhole attacks using a trust-based mechanism. In *2016 26th International Telecommunication Networks and Applications Conference (ITNAC)*. IEEE, 115–120. https://doi.org/10.1109/ATNAC.2016.7878793

[3] R. Al-Ali, R. Heinrich, P. Hnetynka, A. Juan-Verdejo, S. Seifermann, and M. Walter. 2018. Modeling of Dynamic Trust Contracts for Industry 4.0 Systems. In *Proceedings of the 12th European Conference on Software Architecture: Companion Proceedings (ECSA '18)*. ACM, Article 45, 4 pages. https://doi.org/10.1145/3241403.3241450

[4] F. Al-Turjman and S. Alturjman. 2018. Context-Sensitive Access in Industrial Internet of Things (IIoT) Healthcare Applications. *IEEE Transactions on Industrial Informatics* 14, 6 (2018), 2736–2744. https://doi.org/10.1109/TII.2018.2808190

[5] P. Autenrieth, C. Lörcher, C. Pfeiffer, T. Winkens, and L. Martin. 2018. Current Significance of IT-Infrastructure Enabling Industry 4.0 in Large Companies. In *2018 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC)*. IEEE, 1–8. https://doi.org/10.1109/ICE.2018.8436244

[6] Z. Bakhshi, A. Balador, and J. Mustafa. 2018. Industrial IoT security threats and concerns by considering Cisco and Microsoft IoT reference models. In *2018 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*. IEEE, 173–178. https://doi.org/10.1109/WCNCW.2018.8368997

[7] N.C. Batista, R. Melício, and V.M.F. Mendes. 2017. Services enabler architecture for smart grid and smart living services providers under industry 4.0. *Energy and Buildings* 141 (2017), 16–27. https://doi.org/10.1016/j.enbuild.2017.02.039

[8] E. Bauer, O. Schluga, S. Maksuti, A. Bicaku, D. Hofbauer, I. Ivkic, M. G. Tauber, and A. Wöhrer. 2017. Towards a security baseline for IaaS-cloud back-ends in Industry 4.0. In *2017 12th International Conference for Internet Technology and Secured Transactions (ICITST)*. IEEE, 427–432. https://doi.org/10.23919/ICITST.2017.8356438

[9] A. Bécue, Y. Fourastier, I. Praça, A. Savarit, C. Baron, B. Gradussofs, E. Pouille, and C. Thomas. 2018. CyberFactory#1 — Securing the industry 4.0 with cyber-ranges and digital twins. In *2018 14th IEEE International Workshop on Factory Communication Systems (WFCS)*. IEEE, 1–4. https://doi.org/10.1109/WFCS.2018.8402377

[10] J. Beel and B. Gipp. 2009. Google Scholar's Ranking Algorithm: An Introductory Overview. In *Proceedings of the 12th International Conference on Scientometrics and Informetrics (ISSI'09*. Springer, 439–446.

[11] M. Beltrán, M. Calvo, and S. González. 2017. Federated system-to-service authentication and authorization combining PUFs and tokens. In *2017 12th International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC)*. IEEE, 1–8. https://doi.org/10.1109/ReCoSoC.2017.8016157

[12] N. Benias and A. P. Markopoulos. 2017. A review on the readiness level and cyber-security challenges in Industry 4.0. In *2017 South Eastern European Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM)*. IEEE, 1–5. https://doi.org/10.23919/SEEDA-CECNSM.2017.8088234

[13] A. Bicaku, S. Maksuti, S. Palkovits-Rauter, M. Tauber, R. Matischek, C. Schmittner, G. Mantas, M. Thron, and J. Delsing. 2017. Towards trustworthy end-to-end communication in industry 4.0. In *2017 IEEE 15th International Conference on Industrial Informatics (INDIN)*. IEEE, 889–896. https://doi.org/10.1109/INDIN.2017.8104889

[14] A. Bicaku, C. Schmittner, M. Tauber, and J. Delsing. 2018. Monitoring Industry 4.0 applications for security and safety standard compliance. In *2018 IEEE Industrial Cyber-Physical Systems (ICPS)*. IEEE, 749–754. https://doi.org/10.1109/ICPHYS.2018.8390801

[15] S. Blanch-Torné, F. Cores, and R. M. Chiral. 2015. Agent-based PKI for Distributed Control System. In *2015 World Congress on Industrial Control Systems Security (WCICSS)*. IEEE, 28–35. https://doi.org/10.1109/WCICSS.2015.7420319

[16] G. Bloom, B. Alsulami, E. Nwafor, and I. C. Bertolotti. 2018. Design patterns for the industrial Internet of Things. In *2018 14th IEEE International Workshop on Factory Communication Systems (WFCS)*. IEEE, 1–10. https://doi.org/10.1109/WFCS.2018.8402353

[17] A. Bluschke, W. Bueschel, M. Hohmuth, F. Jehring, R. Kaminski, K. Klamka, S. Koepsell, A. Lackorzynski, T. Lackorzynski, M. Matthews, P. Rietzsch, A. Senier, P. Sieber, V. Ulrich, R. Wiggers, and J. Wolter. 2018. fastvpn - Secure and Flexible Networking for Industry 4.0. In *Broadband Coverage in Germany; 12th ITG-Symposium*. VDE, 1–8. https://imld.de/en/research/research-projects/fastvpn/

[18] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli. 2012. Fog Computing and Its Role in the Internet of Things. In *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing (MCC '12)*. ACM, 13–16. https://doi.org/10.1145/2342509.2342513

[19] H. Boyes, B. Hallaq, J. Cunningham, and T. Watson. 2018. The industrial internet of things (IIoT): An analysis framework. *Computers in Industry* 101 (2018), 1–12. https://doi.org/10.1016/j.compind.2018.04.015

[20] R. Chaturvedi. 2017. UL testing standards to mitigate cybersecurity risk ~ UL's approach with complement to the other standards for SICE 2017. In *2017 56th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*. IEEE, 728–730. https://doi.org/10.23919/SICE.2017.8105618

[21] M. Cheminod, L. Durante, L. Seno, F. Valenza, A. Valenzano, and C. Zunino. 2017. Leveraging SDN to improve security in industrial networks. In *2017 IEEE 13th International Workshop on Factory Communication Systems (WFCS)*. IEEE, 1–7. https://doi.org/10.1109/WFCS.2017.7991960

[22] G. Chen and W. S. Ng. 2017. An efficient authorization framework for securing industrial Internet of Things. In *TENCON 2017 - 2017 IEEE Region 10 Conference*. IEEE, 1219–1224. https://doi.org/10.1109/TENCON.2017.8228043

[23] M. Chen, Y. Miao, Y. Hao, and K. Hwang. 2017. Narrow Band Internet of Things. *IEEE Access* 5 (2017), 20557–20577. https://doi.org/10.1109/ACCESS.2017.2751586

[24] S. R. Chhetri, N. Rashid, S. Faezi, and M. A. A. Faruque. 2017. Security Trends and Advances in Manufacturing Systems in the Era of Industry 4.0. In *Proceedings of the 36th International Conference on Computer-Aided Design (ICCAD '17)*. IEEE Press, 1039–1046. https://doi.org/10.1109/ICCAD.2017.8203896

[25] K. R. Choo, S. Gritzalis, and J. H. Park. 2018. Cryptographic Solutions for Industrial Internet-of-Things: Research Challenges and Opportunities. *IEEE Transactions on Industrial Informatics* 14, 8 (2018), 3567–3569. https://doi.org/10.1109/TII.2018.2841049

[26] M. W. Condry and C. B. Nelson. 2016. Using Smart Edge IoT Devices for Safer, Rapid Response With Industry IoT Control Operations. *Proc. IEEE* 104, 5 (2016), 938–946. https://doi.org/10.1109/JPROC.2015.2513672

[27] H. Cui, R. H. Deng, J. K. Liu, X. Yi, and Y. Li. 2018. Server-Aided Attribute-Based Signature With Revocation for Resource-Constrained Industrial-Internet-of-Things Devices. *IEEE Transactions on Industrial Informatics* 14, 8 (2018), 3724–3732. https://doi.org/10.1109/TII.2018.2813304

[28] B. Czybik, S. Hausmann, S. Heiss, and J. Jasperneite. 2013. Performance evaluation of MAC algorithms for real-time Ethernet communication systems. In *2013 11th IEEE International Conference on Industrial Informatics (INDIN)*. IEEE, 676–681. https://doi.org/10.1109/INDIN.2013.6622965

[29] A. K. Das, M. Wazid, N. Kumar, A. V. Vasilakos, and J. J. P. C. Rodrigues. 2018. Biometrics-Based Privacy-Preserving User Authentication Scheme for Cloud-Based Industrial Internet of Things Deployment. *IEEE Internet of Things Journal* 5, 6 (2018), 4900–4913. https://doi.org/10.1109/JIOT.2018.2877690

[30] P. Daugherty and B. Berthon. 2015. *Winning with the Industrial Internet of Things: How to Accelerate the Journey to Productivity and Growth*. Technical Report. Dublin: Accenture.

[31] Michele De Donno, Nicola Dragoni, Alberto Giaretta, and Angelo Spognardi. 2017. Analysis of DDoS-Sapable IoT Malwares. In *Federated Conference on Computer Science and Information Systems (FedCSIS)*. IEEE, 807–816.

[32] Michele De Donno, Nicola Dragoni, Alberto Giaretta, and Angelo Spognardi. 2018. DDoS-Capable IoT Malwares: Comparative Analysis and Mirai Investigation. *Security and Communication Networks* 2018 (2018).

[33] J. Delsing. 2017. Local Cloud Internet of Things Automation: Technology and Business Model Features of Distributed Internet of Things Automation Solutions. *IEEE Industrial Electronics Magazine* 11, 4 (2017), 8–21. https://doi.org/10.1109/MIE.2017.2759342

[34] Nicola Dragoni, Alberto Giaretta, and Manuel Mazzara. 2017. The Internet of Hackable Things. In *Proceedings of 5th International Conference in Software Engineering for Defence Applications*, Paolo Ciancarini, Stanislav Litvinov, Angelo Messina, Alberto Sillitti, and Giancarlo Succi (Eds.). Springer, 129–140.

[35] M. H. Eldefrawy, N. Pereira, and M. Gidlund. 2018. Key Distribution Protocol for Industrial Internet of Things without Implicit Certificates. *IEEE Internet of Things Journal* (2018). https://doi.org/10.1109/JIOT.2018.2865212 (early access).

[36] C. Esposito, A. Castiglione, F. Palmieri, and A. D. Santis. 2018. Integrity for an Event Notification Within the Industrial Internet of Things by Using Group Signatures. *IEEE Transactions on Industrial Informatics* 14, 8 (2018), 3669–3678. https://doi.org/10.1109/TII.2018.2791956

[37] G. Falco, C. Caldera, and H. Shrobe. 2018. IIoT Cybersecurity Risk Modeling for SCADA Systems. *IEEE Internet of Things Journal* 5, 6 (2018), 4486–4495. https://doi.org/10.1109/JIOT.2018.2822842

[38] X. Feng, J. Wu, J. Li, and S. Wang. 2018. Efficient Secure Access to IEEE 21451 Based Wireless IIoT Using Optimized TEDS and MIB. In *IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society*. IEEE, 5221–5227. https://doi.org/10.1109/IECON.2018.8591182

[39] H. Flatt, S. Schriegel, J. Jasperneite, H. Trsek, and H. Adamczyk. 2016. Analysis of the Cyber-Security of industry 4.0 technologies based on RAMI 4.0 and identification of requirements. In *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 1–4. https://doi.org/10.1109/ETFA.2016.7733634

[40] J. L. Flores and I. Mugarza. 2018. Runtime Vulnerability Discovery as a Service on Industrial Internet of Things (IIoT) Systems. In *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, Vol. 1. IEEE, 948–955. https://doi.org/10.1109/ETFA.2018.8502660

[41] F. Fraile, T. Tagawa, R. Poler, and A. Ortiz. 2018. Trustworthy Industrial IoT Gateways for Interoperability Platforms and Ecosystems. *IEEE Internet of Things Journal* 5, 6 (2018), 4506–4514. https://doi.org/10.1109/JIOT.2018.2832041

[42] J. Fu, Y. Liu, H. Chao, B. K. Bhargava, and Z. Zhang. 2018. Secure Data Storage and Searching for Industrial IoT by Integrating Fog Computing and Cloud Computing. *IEEE Transactions on Industrial Informatics* 14, 10 (2018), 4519–4528. https://doi.org/10.1109/TII.2018.2793350

[43] G. George and S. M. Thampi. 2018. A Graph-Based Security Framework for Securing Industrial IoT Networks From Vulnerability Exploitations. *IEEE Access* 6 (2018), 43586–43601. https://doi.org/10.1109/ACCESS.2018.2863244

[44] A. Hassanzadeh, S. Modi, and S. Mulchandani. 2015. Towards effective security control assignment in the Industrial Internet of Things. In *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*. IEEE, 795–800. https://doi.org/10.1109/WF-IoT.2015.7389155

[45] A. Hoeller and R. Toegl. 2018. Trusted Platform Modules in Cyber-Physical Systems: On the Interference Between Security and Dependability. In *2018 IEEE European Symposium on Security and Privacy Workshops (EuroS PW)*. IEEE, 136–144. https://doi.org/10.1109/EuroSPW.2018.00026

[46] F. Hofer. 2018. Architecture, Technologies and Challenges for Cyber-physical Systems in Industry 4.0: A Systematic Mapping Study. In *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM '18)*. ACM, Article 1, 10 pages. https://doi.org/10.1145/3239235.3239242

[47] F. Hofer. 2018. Enhancing Security and Reliability for Smart- Systems' Architectures. In *2018 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*. IEEE, 150–153. https://doi.org/10.1109/ISSREW.2018.000-8

[48] P. Hu. 2015. A System Architecture for Software-Defined Industrial Internet of Things. In *2015 IEEE International Conference on Ubiquitous Wireless Broadband (ICUWB)*. IEEE, 1–5. https://doi.org/10.1109/ICUWB.2015.7324414

[49] Y. Huang and W. Sun. 2018. An AHP-Based Risk Assessment for an Industrial IoT Cloud. In *2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*. IEEE, 637–638. https://doi.org/10.1109/QRS-C.2018.00112

[50] F. Januário, C. Carvalho, A. Cardoso, and P. Gil. 2016. Security challenges in SCADA systems over Wireless Sensor and Actuator Networks. In *2016 8th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*. IEEE, 363–368. https://doi.org/10.1109/ICUMT.2016.7765386

[51] N. Jazdi. 2014. Cyber physical systems in the context of Industry 4.0. In *2014 IEEE International Conference on Automation, Quality and Testing, Robotics*. IEEE, 1–4. https://doi.org/10.1109/AQTR.2014.6857843

[52] S. Jeong, W. Na, J. Kim, and S. Cho. 2018. Internet of Things for Smart Manufacturing System: Trust Issues in Resource Allocation. *IEEE Internet of Things Journal* 5, 6 (2018), 4418–4427. https://doi.org/10.1109/JIOT.2018.2814063

[53] P. Kadera and P. Novák. 2017. Performance Modeling Extension of Directory Facilitator for Enhancing Communication in FIPA-Compliant Multiagent Systems. *IEEE Transactions on Industrial Informatics* 13, 2 (2017), 688–695. https://doi.org/10.1109/TII.2016.2601918

[54] H. Kagermann, W. Wahlster, and J. Helbig. 2013. *Recommendations for Implementing the Strategic Initiative INDUSTRIE 4.0 – Securing the Future of German Manufacturing Industry*. Final Report of the Industrie 4.0 Working Group. acatech – National Academy of Science and Engineering, München. http://forschungsunion.de/pdf/industrie_4_0_final_report.pdf

[55] E. Kail, A. Banati, E. Lászlo, and M. Kozlovszky. 2018. Security Survey of Dedicated IoT Networks in the Unlicensed ISM Bands. In *2018 IEEE 12th International Symposium on Applied Computational Intelligence and Informatics (SACI)*. IEEE, 000449–000454. https://doi.org/10.1109/SACI.2018.8440945

[56] A. Karati, S. H. Islam, and M. Karuppiah. 2018. Provably Secure and Lightweight Certificateless Signature Scheme for IIoT Environments. *IEEE Transactions on Industrial Informatics* 14, 8 (2018), 3701–3711. https://doi.org/10.1109/TII.2018.2794991

[57] S. Katsikeas, K. Fysarakis, A. Miaoudakis, A. Van Bemten, I. Askoxylakis, I. Papaefstathiou, and A. Plemenos. 2017. Lightweight amp; secure industrial IoT communications via the MQ telemetry transport protocol. In *2017 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 1193–1200. https://doi.org/10.1109/ISCC.2017.8024687

[58] B. Kim and Y. Kang. 2018. Abnormal Traffic Detection Mechanism for Protecting IIoT Environments. In *2018 International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE, 943–945. https://doi.org/10.1109/ICTC.2018.8539533

[59] Y. Kim, Y. Lee, and J. Kim. 2018. RIPPLE: Adaptive fine-grained access control in multi-hop LLNs. In *2018 International Conference on Information Networking (ICOIN)*. IEEE, 863–868. https://doi.org/10.1109/ICOIN.2018.8343245

[60] Barbara Kitchenham and Stuart Charters. 2007. *Guidelines for performing Systematic Literature Reviews in Software Engineering*. Technical Report EBSE-2007-01. EBSE Technical Report.

[61] T. Kobzan, S. Schriegel, S. Althoff, A. Boschmann, J. Otto, and J. Jasperneite. 2018. Secure and Time-sensitive Communication for Remote Process Control and Monitoring. In *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, Vol. 1. IEEE, 1105–1108. https://doi.org/10.1109/ETFA.2018.8502539

[62] K. K. Kolluru, C. Paniagua, J. van Deventer, J. Eliasson, J. Delsing, and R. J. DeLong. 2018. An AAA solution for securing industrial IoT devices using next generation access control. In *2018 IEEE Industrial Cyber-Physical Systems (ICPS)*. IEEE, 737–742. https://doi.org/10.1109/ICPHYS.2018.8390799

[63] F. Kurtz, C. Bektas, N. Dorsch, and C. Wietfeld. 2018. Network Slicing for Critical Communications in Shared 5G Infrastructures - An Empirical Evaluation. In *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*. IEEE, 393–399. https://doi.org/10.1109/NETSOFT.2018.8460110

[64] E. Laarouchi, D. Cancila, and H. Chaouchi. 2017. Safety and degraded mode in civilian applications of unmanned aerial systems. In *2017 IEEE/AIAA 36th Digital Avionics Systems Conference (DASC)*. IEEE, 1–7. https://doi.org/10.1109/DASC.2017.8102040

[65] M. Langfinger, M. Schneider, D. Stricker, and H. D. Schotten. 2017. Addressing security challenges in industrial augmented reality systems. In *2017 IEEE 15th International Conference on Industrial Informatics (INDIN)*. IEEE, 299–304. https://doi.org/10.1109/INDIN.2017.8104789

[66] R. Langner. 2011. Stuxnet: Dissecting a cyberwarfare weapon. *IEEE Security & Privacy* 9, 3 (2011), 49–51.

[67] A. Laszka, W. Abbas, Y. Vorobeychik, and X. Koutsoukos. 2018. Synergistic Security for the Industrial Internet of Things: Integrating Redundancy, Diversity, and Hardening. In *2018 IEEE International Conference on Industrial Internet (ICII)*. IEEE, 153–158. https://doi.org/10.1109/ICII.2018.00025

[68] R Lee. 2017. *CRASHOVERRIDE: Analysis of the threat to electric grid operations*. Technical Report. Dragos Inc.

[69] C. Lesjak, H. Bock, D. Hein, and M. Maritsch. 2016. Hardware-secured and transparent multi-stakeholder data exchange for industrial IoT. In *2016 IEEE 14th International Conference on Industrial Informatics (INDIN)*. IEEE, 706–713. https://doi.org/10.1109/INDIN.2016.7819251

[70] C. Lesjak, D. Hein, M. Hofmann, M. Maritsch, A. Aldrian, P. Priller, T. Ebner, T. Ruprechter, and G. Pregartner. 2015. Securing smart maintenance services: Hardware-security and TLS for MQTT. In *2015 IEEE 13th International Conference on Industrial Informatics (INDIN)*. IEEE, 1243–1250. https://doi.org/10.1109/INDIN.2015.7281913

[71] C. Lesjak, D. Hein, and J. Winter. 2015. Hardware-security technologies for industrial IoT: TrustZone and security controller. In *IECON 2015 - 41st Annual Conference of the IEEE Industrial Electronics Society*. IEEE, 002589–002595. https://doi.org/10.1109/IECON.2015.7392493

[72] C. Lesjak, T. Ruprechter, H. Bock, J. Haid, and E. Brenner. 2014. ESTADO — Enabling smart services for industrial equipment through a secured, transparent and ad-hoc data transmission online. In *The 9th International Conference for Internet Technology and Secured Transactions (ICITST-2014)*. IEEE, 171–177. https://doi.org/10.1109/ICITST.2014.7038800

[73] C. Lesjak, T. Ruprechter, J. Haid, H. Bock, and E. Brenner. 2014. A secure hardware module and system concept for local and remote industrial embedded system identification. In *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*. IEEE, 1–7. https://doi.org/10.1109/ETFA.2014.7005086

[74] Marianna Lezzi, Mariangela Lazoi, and Angelo Corallo. 2018. Cybersecurity for Industry 4.0 in the current literature: A reference framework. *Computers in Industry* 103 (2018), 97 – 110. https://doi.org/10.1016/j.compind.2018.09.004

[75] F. Li, J. Hong, and A. A. Omala. 2017. Efficient certificateless access control for industrial Internet of Things. *Future Generation Computer Systems* 76 (2017), 285–292. https://doi.org/10.1016/j.future.2016.12.036

[76] X. Li, J. Niu, M. Z. A. Bhuiyan, F. Wu, M. Karuppiah, and S. Kumari. 2018. A Robust ECC-Based Provable Secure Authentication Protocol With Privacy Preserving for Industrial Internet of Things. *IEEE Transactions on Industrial Informatics* 14, 8 (2018), 3599–3609. https://doi.org/10.1109/TII.2017.2773666

[77] X. Li, J. Peng, J. Niu, F. Wu, J. Liao, and K. R. Choo. 2018. A Robust and Energy Efficient Authentication Protocol for Industrial Internet of Things. *IEEE Internet of Things Journal* 5, 3 (2018), 1606–1615. https://doi.org/10.1109/JIOT.2017.

2787800

[78] Z. Li, J. Kang, R. Yu, D. Ye, Q. Deng, and Y. Zhang. 2018. Consortium Blockchain for Secure Energy Trading in Industrial Internet of Things. *IEEE Transactions on Industrial Informatics* 14, 8 (2018), 3690–3700. https://doi.org/10.1109/TII.2017.2786307

[79] L. Liang, Y. Liu, Y. Yao, T. Yang, Y. Hu, and C. Ling. 2017. Security challenges and risk evaluation framework for industrial wireless sensor networks. In *2017 4th International Conference on Control, Decision and Information Technologies (CoDIT)*. IEEE, 0904–0907. https://doi.org/10.1109/CoDIT.2017.8102711

[80] C. Lin, D. He, X. Huang, K. R. Choo, and A. V. Vasilakos. 2018. BSeIn: A blockchain-based secure mutual authentication with fine-grained access control system for industry 4.0. *Journal of Network and Computer Applications* 116 (2018), 42–52. https://doi.org/10.1016/j.jnca.2018.05.005

[81] M. Ma, D. He, N. Kumar, K. R. Choo, and J. Chen. 2018. Certificateless Searchable Public Key Encryption Scheme for Industrial Internet of Things. *IEEE Transactions on Industrial Informatics* 14, 2 (2018), 759–767. https://doi.org/10.1109/TII.2017.2703922

[82] Z. Ma, A. Hudic, A. Shaaban, and S. Plosz. 2017. Security Viewpoint in a Reference Architecture Model for Cyber-Physical Production Systems. In *2017 IEEE European Symposium on Security and Privacy Workshops (EuroS PW)*. IEEE, 153–159. https://doi.org/10.1109/EuroSPW.2017.65

[83] S. Maksuti, A. Bicaku, M. Tauber, S. Palkovits-Rauter, S. Haas, and J. Delsing. 2017. Towards flexible and secure end-to-end communication in industry 4.0. In *2017 IEEE 15th International Conference on Industrial Informatics (INDIN)*. IEEE, 883–888. https://doi.org/10.1109/INDIN.2017.8104888

[84] G. Marchetto, R. Sisto, J. Yusupov, and A. Ksentinit. 2018. Formally verified latency-aware VNF placement in industrial Internet of things. In *2018 14th IEEE International Workshop on Factory Communication Systems (WFCS)*. IEEE, 1–9. https://doi.org/10.1109/WFCS.2018.8402355

[85] S. Marksteiner. 2018. Reasoning on Adopting OPC UA for an IoT-Enhanced Smart Energy System from a Security Perspective. In *2018 IEEE 20th Conference on Business Informatics (CBI)*, Vol. 02. IEEE, 140–143. https://doi.org/10.1109/CBI.2018.10060

[86] D. W. McKee, S. J. Clement, J. Almutairi, and J. Xu. 2017. Massive-Scale Automation in Cyber-Physical Systems: Vision amp;amp; Challenges. In *2017 IEEE 13th International Symposium on Autonomous Decentralized System (ISADS)*. IEEE, 5–11. https://doi.org/10.1109/ISADS.2017.56

[87] D. W. McKee, S. J. Clement, J. Almutairi, and J. Xu. 2018. Survey of advances and challenges in intelligent autonomy for distributed cyber-physical systems. *CAAI Transactions on Intelligence Technology* 3, 2 (2018), 75–82. https://doi.org/10.1049/trit.2018.0010

[88] A. Melis, D. Berardi, C. Contoli, F. Callegati, F. Esposito, and M. Prandini. 2018. A Policy Checker Approach for Secure Industrial SDN. In *2018 2nd Cyber Security in Networking Conference (CSNet)*. IEEE, 1–7. https://doi.org/10.1109/CSNET.2018.8602927

[89] H. Mouratidis and V. Diamantopoulou. 2018. A Security Analysis Method for Industrial Internet of Things. *IEEE Transactions on Industrial Informatics* 14, 9 (2018), 4093–4100. https://doi.org/10.1109/TII.2018.2832853

[90] N. Moustafa, E. Adi, B. Turnbull, and J. Hu. 2018. A New Threat Intelligence Scheme for Safeguarding Industry 4.0 Systems. *IEEE Access* 6 (2018), 32910–32924. https://doi.org/10.1109/ACCESS.2018.2844794

[91] J. Moyne, S. Mashiro, and D. Gross. 2018. Determining a security roadmap for the microelectronics industry. In *2018 29th Annual SEMI Advanced Semiconductor Manufacturing Conference (ASMC)*. IEEE, 291–294. https://doi.org/10.1109/ASMC.2018.8373213

[92] I. Mugarza, J. Parra, and E. Jacob. 2018. Cetratus: Towards a live patching supported runtime for mixed-criticality safe and secure systems. In *2018 IEEE 13th International Symposium on Industrial Embedded Systems (SIES)*. IEEE, 1–8. https://doi.org/10.1109/SIES.2018.8442088

[93] E. T. Nakamura and S. L. Ribeiro. 2018. A Privacy, Security, Safety, Resilience and Reliability Focused Risk Assessment Methodology for IIoT Systems Steps to Build and Use Secure IIoT Systems. In *2018 Global Internet of Things Summit (GIoTS)*. IEEE, 1–6. https://doi.org/10.1109/GIOTS.2018.8534521

[94] M. Niedermaier, F. Fischer, and A. von Bodisco. 2017. PropFuzz — An IT-security fuzzing framework for proprietary ICS protocols. In *2017 International Conference on Applied Electronics (AE)*. IEEE, 1–4. https://doi.org/10.23919/AE.2017.8053600

[95] P. O'Donovan, C. Gallagher, K. Bruton, and D. T.J. O'Sullivan. 2018. A fog computing industrial cyber-physical system for embedded low-latency machine learning Industry 4.0 applications. *Manufacturing Letters* 15 (2018), 139–142. https://doi.org/10.1016/j.mfglet.2018.01.005 Industry 4.0 and Smart Manufacturing.

[96] OpenFog Consortium Architecture Working Group and others. 2017. *OpenFog Reference architecture for Fog Computing*. Technical Report. OpenFog Consortium. https://www.openfogconsortium.org/wp-content/uploads/OpenFog_Reference_Architecture_2_09_17-FINAL.pdf

[97] T. Pereira, L. Barreto, and A. Amaral. 2017. Network and information security challenges within Industry 4.0 paradigm. *Procedia Manufacturing* 13 (2017), 1253–1260. https://doi.org/10.1016/j.promfg.2017.09.047 Manufacturing Engineering Society International Conference 2017, MESIC 2017, 28-30 June 2017, Vigo (Pontevedra), Spain.

[98] K. Petersen, S. Vakkalanka, and L. Kuzniarz. 2015. Guidelines for conducting systematic mapping studies in software engineering: An update. *Information and Software Technology* 64 (2015), 1–18.

[99] D. Preuveneers, W. Joosen, and E. Ilie-Zudor. 2016. Data Protection Compliance Regulations and Implications for Smart Factories of the Future. In *2016 12th International Conference on Intelligent Environments (IE)*. IEEE, 40–47. https://doi.org/10.1109/IE.2016.15

[100] D. Preuveneers, W. Joosen, and E. Ilie-Zudor. 2017. Identity Management for Cyber-physical Production Workflows and Individualized Manufacturing in Industry 4.0. In *Proceedings of the Symposium on Applied Computing (SAC '17)*. ACM, 1452–1455. https://doi.org/10.1145/3019612.3019861

[101] P. Radanliev, D. De Roure, S. Cannady, R. M. Montalvo, R. Nicolescu, and M. Huth. 2018. Economic impact of IoT cyber risk - Analysing past and present to predict the future developments in IoT risk analysis and IoT cyber insurance. In *Living in the Internet of Things: Cybersecurity of the IoT - 2018*. IET, 1–9. https://doi.org/10.1049/cp.2018.0003

[102] A. Sadeghi, C. Wachsmann, and M. Waidner. 2015. Security and Privacy Challenges in Industrial Internet of Things. In *Proceedings of the 52Nd Annual Design Automation Conference (DAC '15)*. ACM, Article 54, 6 pages. https://doi.org/10.1145/2744769.2747942

[103] A. Sajid, H. Abbas, and K. Saleem. 2016. Cloud-Assisted IoT-Based SCADA Systems Security: A Review of the State of the Art and Future Challenges. *IEEE Access* 4 (2016), 1375–1384. https://doi.org/10.1109/ACCESS.2016.2549047

[104] C. Sandberg and B. Hunter. 2017. Cyber security primer for legacy process plant operation. In *2017 Petroleum and Chemical Industry Technical Conference (PCIC)*. IEEE, 97–102. https://doi.org/10.1109/PCICON.2017.8188728

[105] J. Schuette and G. S. Brost. 2018. LUCON: Data Flow Control for Message-Based IoT Systems. In *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*. IEEE, 289–299. https://doi.org/10.1109/TrustCom/BigDataSE.2018.00052

[106] A. Seitz, D. Henze, D. Miehle, B. Bruegge, J. Nickles, and M. Sauer. 2018. Fog Computing as Enabler for Blockchain-Based IIoT App Marketplaces - A Case Study. In *2018 Fifth International Conference on Internet of Things: Systems, Management and Security*. IEEE, 182–188. https://doi.org/10.1109/IoTSMS.2018.8554484

[107] G. Settanni, F. Skopik, A. Karaj, M. Wurzenberger, and R. Fiedler. 2018. Protecting cyber physical production systems using anomaly detection to enable self-adaptation. In *2018 IEEE Industrial Cyber-Physical Systems (ICPS)*. IEEE, 173–180. https://doi.org/10.1109/ICPHYS.2018.8387655

[108] G. Shaabany and R. Anderl. 2018. Security by Design as an Approach to Design a Secure Industry 4.0-Capable Machine Enabling Online-Trading of Technology Data. In *2018 International Conference on System Science and Engineering (ICSSE)*. IEEE, 1–5. https://doi.org/10.1109/ICSSE.2018.8520195

[109] V. Sharma, G. Choudhary, Y. Ko, and I. You. 2018. Behavior and Vulnerability Assessment of Drones-Enabled Industrial Internet of Things (IIoT). *IEEE Access* 6 (2018), 43368–43383. https://doi.org/10.1109/ACCESS.2018.2856368

[110] L. Shu, M. Mukherjee, M. Pecht, N. Crespi, and S. N. Han. 2018. Challenges and Research Issues of Data Management in IoT for Large-Scale Petrochemical Plants. *IEEE Systems Journal* 12, 3 (2018), 2509–2523. https://doi.org/10.1109/JSYST.2017.2700268

[111] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund. 2018. Industrial Internet of Things: Challenges, Opportunities, and Directions. *IEEE Transactions on Industrial Informatics* 14, 11 (2018), 4724–4734. https://doi.org/10.1109/TII.2018.2852491

[112] V. Sklyar and V. Kharchenko. 2017. Challenges in assurance case application for industrial IoT. In *2017 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, Vol. 2. IEEE, 736–739. https://doi.org/10.1109/IDAACS.2017.8095187

[113] Z. A. Solangi, Y. A. Solangi, S. Chandio, M. bt. S. Abd. Aziz, M. S. bin Hamzah, and A. Shah. 2018. The future of data privacy and security concerns in Internet of Things. In *2018 IEEE International Conference on Innovative Research and Development (ICIRD)*. IEEE, 1–4. https://doi.org/10.1109/ICIRD.2018.8376320

[114] T. K. Sung. 2018. Industry 4.0: A Korea perspective. *Technological Forecasting and Social Change* 132 (2018), 40–45. https://doi.org/10.1016/j.techfore.2017.11.005

[115] M. Traub, H. Vögel, E. Sax, T. Streichert, and J. Härri. 2018. Digitalization in automotive and industrial systems. In *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*. IEEE, 1203–1204. https://doi.org/10.23919/DATE.2018.8342198

[116] M. H. u. Rehman, E. Ahmed, I. Yaqoob, I. A. T. Hashem, M. Imran, and S. Ahmad. 2018. Big Data Analytics in Industrial IoT Using a Concentric Computing Model. *IEEE Communications Magazine* 56, 2 (2018), 37–43. https://doi.org/10.1109/

MCOM.2018.1700632

[117] T. Ulz, T. Pieber, C. Steger, S. Haas, H. Bock, and R. Matischek. 2017. Bring your own key for the industrial Internet of Things. In *2017 IEEE International Conference on Industrial Technology (ICIT)*. IEEE, 1430–1435. https://doi.org/10.1109/ICIT.2017.7915575

[118] T. Ulz, T. Pieber, C. Steger, S. Haas, and R. Matischek. 2018. Secured remote configuration approach for industrial cyber-physical systems. In *2018 IEEE Industrial Cyber-Physical Systems (ICPS)*. IEEE, 812–817. https://doi.org/10.1109/ICPHYS.2018.8390811

[119] B. van Lier. 2017. The industrial internet of things and cyber security: An ecological and systemic perspective on security in digital industrial ecosystems. In *2017 21st International Conference on System Theory, Control and Computing (ICSTCC)*. IEEE, 641–647. https://doi.org/10.1109/ICSTCC.2017.8107108

[120] R. Vanickis, P. Jacob, S. Dehghanzadeh, and B. Lee. 2018. Access Control Policy Enforcement for Zero-Trust-Networking. In *2018 29th Irish Signals and Systems Conference (ISSC)*. IEEE, 1–6. https://doi.org/10.1109/ISSC.2018.8585365

[121] K. Wallis, F. Kemmer, E. Jastremskoj, and C. Reich. 2017. Adaption of a Privilege Management Infrastructure (PMI) Approach to Industry 4.0. In *2017 5th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*. IEEE, 101–107. https://doi.org/10.1109/FiCloudW.2017.71

[122] K. Matthias Weber, Niklas Gudowsky, and Georg Aichholzer. 2018. Foresight and technology assessment for the Austrian parliament — Finding new ways of debating the future of industry 4.0. *Futures* (2018). https://doi.org/10.1016/j.futures.2018.06.018

[123] E. Weippl and P. Kieseberg. 2017. Security in cyber-physical production systems: A roadmap to improving IT-security in the production system lifecycle. In *2017 AEIT International Annual Conference*. IEEE, 1–6. https://doi.org/10.23919/AEIT.2017.8240552

[124] F. Xiao, L. Sha, Z. Yuan, and R. Wang. 2018. VulHunter: A Discovery for unknown Bugs based on Analysis for known patches in Industry Internet of Things. *IEEE Transactions on Emerging Topics in Computing* (2018). https://doi.org/10.1109/TETC.2017.2754103 (early access).

[125] P. Xu, S. He, W. Wang, W. Susilo, and H. Jin. 2018. Lightweight Searchable Public-Key Encryption for Cloud-Assisted Wireless Sensor Networks. *IEEE Transactions on Industrial Informatics* 14, 8 (2018), 3712–3723. https://doi.org/10.1109/TII.2017.2784395

[126] Q. Yan, W. Huang, X. Luo, Q. Gong, and F. R. Yu. 2018. A Multi-Level DDoS Mitigation Framework for the Industrial Internet of Things. *IEEE Communications Magazine* 56, 2 (2018), 30–36. https://doi.org/10.1109/MCOM.2018.1700621

[127] C. Yin, J. Xi, R. Sun, and J. Wang. 2018. Location Privacy Protection Based on Differential Privacy Strategy for Big Data in Industrial Internet of Things. *IEEE Transactions on Industrial Informatics* 14, 8 (2018), 3628–3636. https://doi.org/10.1109/TII.2017.2773646

[128] M. Yousif. 2016. Manufacturing and the Cloud. *IEEE Cloud Computing* 3, 4 (2016), 4–5. https://doi.org/10.1109/MCC.2016.77

[129] M. Yu, M. Zhu, G. Chen, J. Li, and Z. Zhou. 2016. A cyber-physical architecture for industry 4.0-based power equipments detection system. In *2016 International Conference on Condition Monitoring and Diagnosis (CMD)*. IEEE, 782–785. https://doi.org/10.1109/CMD.2016.7757942

[130] S. Zanero. 2017. Cyber-Physical Systems. *Computer* 50, 4 (2017), 14–16. https://doi.org/10.1109/MC.2017.105

[131] L. Zhou and H. Guo. 2018. Anomaly Detection Methods for IIoT Networks. In *2018 IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI)*. IEEE, 214–219. https://doi.org/10.1109/SOLI.2018.8476769

[132] L. Zhou, K. Yeh, G. Hancke, Z. Liu, and C. Su. 2018. Security and Privacy for the Industrial Internet of Things: An Overview of Approaches to Safeguarding Endpoints. *IEEE Signal Processing Magazine* 35, 5 (2018), 76–87. https://doi.org/10.1109/MSP.2018.2846297

[133] M. Zolanvari, M. A. Teixeira, and R. Jain. 2018. Effect of Imbalanced Datasets on Security of Industrial IoT Using Machine Learning. In *2018 IEEE International Conference on Intelligence and Security Informatics (ISI)*. IEEE, 112–117. https://doi.org/10.1109/ISI.2018.8587389

[134] E. Zugasti, M. Iturbe, I. Garitano, and U. Zurutuza. 2018. Null is Not Always Empty: Monitoring the Null Space for Field-Level Anomaly Detection in Industrial IoT Environments. In *2018 Global Internet of Things Summit (GIoTS)*. IEEE, 1–6. https://doi.org/10.1109/GIOTS.2018.8534574

Paper B

# A Systematic Survey of Industrial Internet of Things Security: Requirements and Fog Computing Opportunities

K. Tange, M. De Donno, X. Fafoutis, and N. Dragoni. "A Systematic Survey of Industrial Internet of Things Security: Requirements and Fog Computing Opportunities." In: *IEEE Communications Surveys Tutorials* (2020). DOI: 10.1109/COMST.2020.3011208

# A Systematic Survey of Industrial Internet of Things Security: Requirements and Fog Computing Opportunities

Koen Tange, *Student Member, IEEE,* Michele De Donno, *Student Member, IEEE,* Xenofon Fafoutis, *Senior Member, IEEE,* and Nicola Dragoni

***Abstract*—A key application of the Internet of Things (IoT) paradigm lies within industrial contexts. Indeed, the emerging Industrial Internet of Things (IIoT), commonly referred to as Industry 4.0, promises to revolutionize production and manufacturing through the use of large numbers of networked embedded sensing devices, and the combination of emerging computing technologies, such as Fog/Cloud Computing and Artificial Intelligence. The IIoT is characterized by an increased degree of inter-connectivity, which not only creates opportunities for the industries that adopt it, but also for cyber-criminals. Indeed, IoT security currently represents one of the major obstacles that prevent the widespread adoption of IIoT technology. Unsurprisingly, such concerns led to an exponential growth of published research over the last few years. To get an overview of the field, we deem it important to systematically survey the academic literature so far, and distill from it various security requirements as well as their popularity. This paper consists of two contributions: our primary contribution is a systematic review of the literature over the period 2011-2019 on IIoT Security, focusing in particular on the security requirements of the IIoT. Our secondary contribution is a reflection on how the relatively new paradigm of Fog computing can be leveraged to address these requirements, and thus improve the security of the IIoT.**

***Index Terms*—Industrial Internet of Things, Cyber-security, Security Requirements, Fog Computing**

## ACRONYMS

| | |
|---|---|
| **3GPP** | The 3rd Generation Partnership Project |
| **5G** | Fifth generation cellular network technology |
| **ABS** | Attribute Based Signatures |
| **AC** | Access Control |
| **ACL** | Access Control List |
| **BYOK** | Bring Your Own Key |
| **CIA** | Confidentiality, Integrity, Availability |
| **CI** | Critical Infrastructure |
| **DDoS** | Distributed Denial of Service |
| **DHT** | Distributed Hash Table |
| **DID** | Decentralized Identifier |
| **DoS** | Denial of Service |
| **DTLS** | Datagram Transport Layer Security |
| **ENISA** | European Union Agency for Network and Information Security |
| **GDPR** | General Data Protection Regulation |
| **ICS** | Industrial Internet Consortium |
| **IDS** | Intrusion Detection System |
| **IETF** | Internet Engineering Task Force |
| **IIoT** | Industrial Internet of Things |
| **IoT** | Internet of Things |
| **LLN** | Low-power Lossy network |
| **LPWAN** | Low Power Wide Area Network |
| **M2M** | Machine-to-Machine |
| **MQTT** | Message Queue Telemetry Transport |
| **NB-IoT** | Narrow Band IoT |
| **NFC** | Near Field Communication |
| **NFV** | Network Function Virtualization |
| **OPC UA** | The OPC Unified Architecture |
| **OT** | Operational Technology |
| **OWASP** | Open Web Application Security Project |
| **PKI** | Public Key Infrastructure |
| **PLC** | Programmable Logic Controller |
| **PUF** | Physically Uncloneable Function |
| **SCADA** | Supervisory Control And Data Acquisition |
| **SDN** | Software Defined Networking |
| **SSI** | Self Sovereign Identity |
| **TEE** | Trusted Execution Environment |
| **TLS** | Transport Layer Security |
| **TPM** | Trusted Platform Module |
| **TSN** | Time Sensitive Networking |
| **WPAN** | Wireless Personal Area Network |
| **WSN** | Wireless Sensor Networks |
| **ZTN** | Zero Trust Networking |

K. Tange, M. De Donno, X. Fafoutis and N. Dragoni are with the Embedded Systems Engineering section, DTU Compute, Technical University of Denmark. Email: {kpta, mido, xefa, ndra}@dtu.dk

Additionally, N. Dragoni is with Örebro University, Sweden

## I. INTRODUCTION

INDUSTRY 4.0, also referred to as $4^{th}$ industrial revolution, represents a new industrial era, whereby due to the increasing availability, affordability, and capability of sensors, processors, and communication technologies, the number of embedded devices used in industrial applications is rapidly increasing. This leads to a growth in the interest for the Industrial Internet of Things (IIoT): a large network of devices, systems, and applications communicating and sharing intelligence with each other, the external environment, and with humans [1]. According to Accenture [1], the IIoT could be worth 7.1 trillion US dollars to the United States and more than 1.2 trillion to Europe by 2030.

In this wave of excitement, Internet of Things (IoT) security represents one of the biggest weak points holding back the adoption of the IIoT. As a matter of fact, IoT devices are

often poorly secured [2] and thus easy targets for malware taking advantage of them to run devastating cyber-attacks, such as Distributed Denial of Service (DDoS) [3] (e.g., Mirai [4] affected consumer IoT) or sabotage attacks. Threats are not limited to the consumer IoT. In fact, traditional industrial environments have been subject to attacks in the past, sometimes with devastating results (e.g., StuxNet [5] or CrashOverride/Industroyer [6]). It is thus apparent that without security, IIoT will never be able to deliver its full potential. As a result, recent years have seen an unprecedented growth of research in IIoT security.

In this landscape, a relatively new computing paradigm has attracted attention: Fog computing [7]. Fog computing is a system-level architecture born from the necessity of bridging the gap between IoT and Cloud computing, by distributing resources and services along the continuum from Cloud to IoT [8]. Among others, one of the promises of Fog computing is to present a possible solution to the (I)IoT security problem.

### A. Contribution

In this article, we present a systematic survey on the security requirements of the IIoT. As we quantitatively demonstrate in Section VI, the field of IIoT security has grown rapidly over the last few years, and this momentum motivates this article and the need for an up-to-date systematic survey.

In particular, as our primary contribution, we survey the literature on IIoT security over the period 2011-2019, which corresponds to more than 200 papers. In turn, we identify, categorize, and discuss the IIoT security requirements that have been identified by the research community, highlighting the research interest attracted by each of them over the target period. In addition, we provide statistics with regard to the geographical distribution and the publication venue of the surveyed papers.

As a secondary contribution, in the final part of the article, we discuss how the Fog computing paradigm can be used to address these requirements. Our reflection identifies numerous research opportunities at the intersection of Fog computing and IIoT security, along with open challenges and limitations still (partially) unsolved.

### B. Outline

The paper is organized as follows. We first establish common ground by discussing the difference between IoT and IIoT, and providing a glimpse into recent IoT security surveys. Section III briefly mentions related work and motivates the need for a systematic literature review. Section IV describes the research method used in the review and formalizes the research questions. Section V surveys the security requirements resulting from the systematic review. Section VI presents a quantitative analysis of the results obtained during the research phase. Section VII discusses the role that Fog computing might play in meeting the IIoT security requirements. Finally, Section VIII concludes the paper.

## II. IoT and IIoT

Before we discuss the results of our systematic survey in depth, it is helpful to establish a common understanding of how IoT and IIoT differ. In this section, we first explore this difference, then, we provide an overview of recent IoT security surveys.

We find Table I, taken from the ENISA "Good practices for Security of Internet of Things in the context of Smart Manufacturing" [9] report, to be helpful in outlining the differences between IoT and IIoT, and use this as a guideline throughout our work. That said, the difference is not a precise, clear-cut one, and we sometimes do deviate from these guidelines, when it is abundantly clear that a scenario concerns the IIoT without meeting relevant criteria from that table.

In general, it is accepted that IIoT is a subset of IoT: IoT typically covers consumer devices in retail and lifestyle, IIoT focuses mainly on Operational Technology (OT), the smart manufacturing process, smart logistics, and smart cities.

It should not be surprising that the safety and security requirements in IIoT are generally stricter than those found in a typical IoT scenario. Even so, we find significant overlap in used terminology in the literature, and IIoT having stricter requirements does not necessarily mean that any proposed security solution for the IoT is not applicable to the IIoT. This is echoed by Yu et al. [10], who, in a short survey on the differences between IIoT and IoT security, find that for the most part, the challenges overlap. At the same time, as will become evident throughout this study, the field is broad, and scenarios covered in the literature differ wildly. Often, one can imagine a more general IoT cousin to a specific IIoT scenario quite easily. The security requirements distilled from said IIoT scenario would thus often also apply to its IoT cousin. Vice-versa, it is likely that works are covering the IoT scenario, these would identify requirements that have not been covered in the available literature for the IIoT. This is especially true for requirements derived out of common challenges such as resource constraints and key distribution. Therefore, we recommend readers with an interest in any given IIoT scenario to also search the available literature for the more general IoT case, and consider if the requirements found in those works uncover security liabilities that have not been addressed in existing IIoT work.

### A. IoT Security Surveys

There exist ample surveys investigating the state of IoT security, and we will briefly look at several relatively recent surveys, discussing how their identified security requirements might relate to the IIoT.

In [11], the authors survey the literature for real IoT attacks and present a taxonomy. They also identify integrity, anonymity, confidentiality, privacy, access control and authorization, authentication, resilience, and self-organization as security requirements for IoT systems in general. These are all represented in the requirements collected in this work as well, and reiterate that generic IoT solutions can work for IIoT systems, if they do not violate scenario-specific constraints. Neshenko et al. [12] provide a much more thorough study

TABLE I
"INDICATIVE DIFFERENCES IN TERMS OF SELECTED ASPECTS BETWEEN IOT AND IIOT" (TAKEN FROM [9]).

| Selected Characteristics | Internet of Things | Industrial Internet of Things |
|---|---|---|
| Focus | Protection of personal data and assets | Prevention of process interruption, safety |
| Priorities | Confidentiality, Integrity, Availability | Availability, Integrity, Confidentiality |
| Device Failure Implications | No critical consequences | Interruption of processes, impact on production, potential physical threats |
| Reaction to threat | Possible shut down and remediation | Maintenance of operation |
| Upgrades and Patch Management | Possible during operation time, no reasons for significant delays | Need to be scheduled and performed during down time, which may postpone the upgrade for a considerable amount of time. |
| Lifecycle of the device | Relatively frequent upgrades of equipment | Long lifespan of the devices (over 15 years) |
| Conditions of deployment | Regular environment | Harsh environment (temperature, vibration, etc) |

of IoT vulnerabilities and attacks, but do not relate these to security requirements. Nevertheless, we can see that the familiar topics of authentication and access control, assurance, and confidentiality return implicitly throughout the text. The threats described by the authors include problems such as false data injection, improper patch management, and improper encryption. Many of these can be directly connected to the security requirements listed in this work.

In [13], the authors provide a top-down survey of IoT security. They discuss security requirements for healthcare, smart grids, manufacturing, smart homes, transport, and smart cities. Some of these are also considered to be in the IIoT domain [9], and indeed the security requirements identified in these sections overlap with the ones collected in this survey, albeit on a higher level of abstraction. In each investigated domain, they list a subset of these as requirements. For smart grids, they identify availability, confidentiality, integrity, non-repudiation, and privacy, and additionally list challenges we also identify in our work: heterogeneity, scalability, privacy, and so on. What is apparent through their work, is that the main way in which the requirements for the various domains differ is in their priority, for instance, privacy and confidentiality weigh higher in healthcare than in transport. Further, the authors make the insightful observation that one specific challenge for the IIoT that is not as apparent in general IoT networks, is that its crucial safety requirements often compete with security in terms of resources. It is perhaps the balance that must be found between these two aspects that sets the IIoT apart from normal IoT systems. Indeed, whenever resource constraints are not an issue, or when safety constraints are less strict, standard IoT solutions often suffice.

## III. RELATED WORK

To the best of our knowledge, the most recent works focused on reviewing IIoT security are [14] and [15]. The former focuses primarily on threats characterization by looking at existing attacks, while the latter mainly reviews the differences between information technology and operational technology in an Industry 4.0 setting, and discusses the challenges. However, neither of these works explicitly discuss security requirements, opting to leave them as implied by the described threats and challenges. Another recent study [16] focuses on Industry 4.0 system architecture as a whole and observes that there is an increase in security-focused architectural proposals, but does not discuss security in depth. Some older surveys dated back to 2015 and 2016 mention IIoT security requirements [17], [18], but they also refrain from an in-depth discussion.

Recently, Hansch et al. [19] published a study identifying and mapping security requirements to an OPC UA model, allowing easier machine-based verification. While they provide many security requirements, they are based on a limited set of use cases, and no thorough explanation for their derivation is given. Moreover, they are of a less abstract level than the ones we attempt to derive in this work.

As a result, we deem it necessary to provide an up-to-date, systematic survey that specifically addresses IIoT security requirements.

## IV. RESEARCH METHOD

In this section, we present the research method that is used in this systematic literature review on security requirements for the IIoT.

We adopt the research method detailed by Petersen et al. [20] and utilize the suggested template for describing our approach. In the next subsections, we elaborate on research questions, search strategy, study selection, and validity concerns.

### A. Research Questions

The main aim of this work is to identify security requirements for the IIoT. This can then guide us in identifying which of these show potential to be solved by Fog computing. In addition, we want to provide an overview of the research activity in the field: how research activity has developed throughout the years, how this research was published, and what its geographical distribution is.

Thus, our research questions can be formulated as follows:

- **RQ1:** what are the security requirements of the IIoT?
- **RQ2:** how are publications related to IIoT security spread throughout the years?
- **RQ3:** how is IIoT security research activity geographically distributed?
- **RQ4:** what are the most popular publication venues for IIoT security research?

Answering these questions will aid in getting a better understanding of the current security landscape for the IIoT, while at the same time identifying various concrete research opportunities related to Fog computing. Each of these can then be traced back to concrete security requirements relevant to the Industry 4.0 paradigm.

TABLE II
QUERIES USED FOR OUR SEARCH, EXPRESSED IN PSEUDO-CODE

| Query | Description |
|---|---|
| Q1 | *in title:* IIoT OR "Industrial Internet of Things" OR "Industry 4.0" |
| Q2 | (*in title:* IIoT OR "Industrial Internet of Things" OR "Industry 4.0") AND *in abstract:* security |

TABLE III
NUMBER OF PAPERS OBTAINED

| Source | Q1 | Q2 |
|---|---|---|
| ACM | 60 | 12 |
| IEEE Xplore | 2702 | 323 |
| ScienceDirect | 369 | 21 |
| **Total** | *3158* | *356* |

### B. Search Strategy

We utilize the adjusted PICOC criteria for software engineering [21] in order to identify relevant keywords. In particular:

- **Population:** we consider the IIoT as the application area in which our research is conducted. However, this is a very broad population, therefore, we take into account only studies addressing IIoT security.
- **Intervention:** this criterion does not apply to our research questions, as we are interested in *any* work in the IIoT domain that describes security requirements.
- **Comparison:** we compare the security requirements identified by different studies by taking into account such factors as the number of studies that mention them, related threats, and proposed solutions.
- **Outcomes:** we present the identified security requirements as well as the properties of their mitigation, allowing us to discuss which requirements call for further research.
- **Context:** as we do not empirically compare the available works, this criterion does not apply to our study.

With these criteria in mind, we have formulated the following keywords: *IIoT, Industrial Internet of Things, Industry 4.0,* and *Security*.

We considered as sources the following databases: ACM Digital Library, IEEE Xplore, Elsevier/ScienceDirect. In this domain, we believe that the combination of these three sources provides an accurate representation of the research that has been conducted globally.

We divided the search into two stages. First, we queried the databases for articles related to IIoT/Industry 4.0 in general, based on their titles. This provided an overview of the amount of research conducted in this field. After that, we narrowed down our search to only include works related to security, by excluding articles not containing the word "security" in their abstract. The queries are summarized in Table II. The search results for both queries are listed in Table III. The queries have been executed in March 2020.

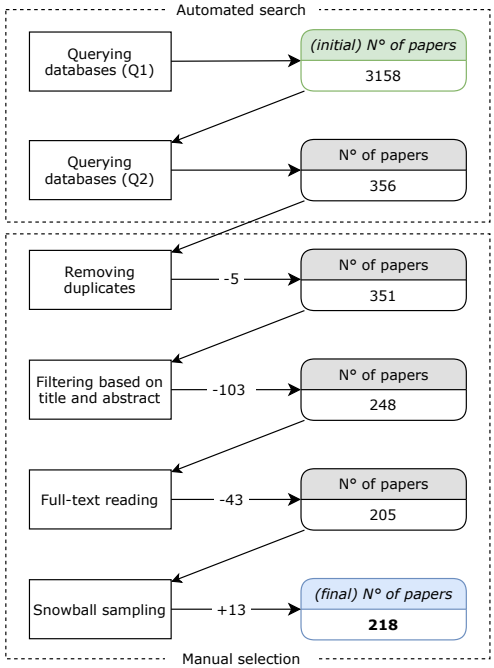



Fig. 1. A schematic representation of the entire study selection process.

### C. Study Selection

Starting from 356 papers resulting from our queries, we further filtered the studies with multiple phases.

Firstly, the JabRef [1] reference management software was used to identify and delete duplicates. Five duplicates were found, leaving the number of considered papers for the subsequent phases at 351.

Subsequently, we independently reviewed the titles and abstracts of each article in order to reduce selection bias. Each article was marked as being relevant, not relevant, or of doubtful relevance. Articles were voted for inclusion when the work covered cyber-security challenges and/or solutions for Industry 4.0, and it was published before 2020, since that is the year in which this study is conducted. We do not believe that filtering on a minimum publication date is necessary at this time, due to the relatively young age of this field. Articles

[1] https://www.jabref.org

were voted for exclusion when the work was not related to Industry 4.0 security, contained duplicate content, or was not presented in legible English.

The following rules were used for filtering out articles based on title and abstract review (this has been done jointly by two authors of the paper):

- when both authors considered an article relevant, the article was included for the next phase;
- when one author expressed doubt and the other author considered an article relevant, the article was included for the next phase;
- when both authors expressed doubt, a joint review was done considering also other sections of the article (e.g. introduction, outline, conclusion) to determine its relevance. If this review did not clear up doubts for either of the authors, the article was given the benefit of the doubt and included for the next phase;
- when one author considered an article relevant, while the other considered it to not be relevant, the article was marked for joint review as described in the previous rule;
- when one author considered an article not relevant, while the other considered it to be doubtful, the article was marked for joint review as with the previous rules;
- when both authors considered an article not relevant, the article was excluded.

After the individual title and abstract reviews, 68 articles were excluded and 69 were marked as doubtful entries requiring a joint review. These were then jointly reviewed, leading to an additional 35 exclusions. The remaining 248 papers were considered for full-text reading, overall reducing the number of papers to analyse by 92% compared to results of Q1 and 30% compared to Q2.

In the full-reading phase, we extracted information relevant to the stated research questions, as well as identifying the challenges discussed in the papers. We then used this data to provide a comprehensive picture of the security challenges and corresponding requirements for the IIoT. In this phase, it became clear that a number of papers were not relevant to our work, resulting in the discarding of other 43 papers. Additionally, we identified 13 papers of interest through reverse snowball sampling and added these to the selection.This brings the final number of papers considered in this survey to 218.

The entire study selection process and related numbers are summarized in Figure 1.

### *D. Validity Evaluation*

Every study that is subject to manual selection is vulnerable to researcher bias in the filtering process. In order to reduce this issue, we performed the filtering process twice: two authors of this paper selected studies independently, and the results of the filtering process were based on a systematic approach combining the selections of both authors, and in some cases a joint review.

Also, to mitigate possible selection bias, we have performed reverse snowball sampling, allowing for the introduction of papers originally not considered due to not being captured by our search queries.

Furthermore, we have described our research process in detail, and have taken care to list the criteria by which we filtered studies. This is done to increase the repeatability of this work.

Finally, it is worth mentioning that our approach does not suffer from the Matthew's effect, as opposed to querying databases that rank papers based on citation count [22].

## V. IIoT Security Requirements (RQ1)

In this section, we present the security requirements that we found to have been discussed in the selected literature. We describe why these requirements are deemed relevant and summarize some of the proposed solutions. We also discuss why these requirements are difficult to satisfy for Industry 4.0 applications, which gives the insight needed to see why the discussed security requirements are hard to meet with conventional solutions. Furthermore, they provide a set of motivational factors for why the research discussed in this section is necessary.

We observed that the focus of the investigated literature is mainly on Industry 4.0, even if in this field highly varying scenarios are considered. For example, some articles discuss petrochemical plant management [246], while others focus on drones [177], [192], [199], and so on. Each of these scenarios has its own threat model and will thus also differ in terms of security requirements from the others, to a certain degree. However, we note that the majority of them show considerable overlap, and that even the ones that are unique to one particular scenario, might still translate into a research opportunity, or might be possibly addressed with Fog computing. Therefore, we have attempted to include all such requirements in this section, and mention their relevance to particular scenarios, to provide context.

In the rest of this section, we discuss all IIoT security requirements found in this study, grouped by the overarching categories to which they belong. Figure 2[2] depicts a hierarchical structure of the various subsections, together with all the works related to each subsection. References were picked and positioned using the following heuristics: firstly, if a work is mentioned in a subsection (be it in a table or the text itself), it is included in the level 1 node representing that subsection (e.g. Authentication); secondly, if a work is mentioned in a topic *within* a subsection (e.g. Key Distribution), it is included in the level 2 node representing that subsubsection in the mind-map. Additionally, in order to minimize redundancy in the mind-map, the following rule was followed: when a reference is included for both a subsection (e.g. Network Security) and one or more of its subsubsections (e.g. Wireless), then preference is given to the latter, and the reference is removed from the subsection itself. This does *not* eliminate redundancy between nodes of the same level (e.g., a reference can still be included for both Key Distribution and Mutual Authentication), but it does allow for a representative overview of works relevant to any topic.

[2]In electronic versions of this work, nodes and references in this map are clickable, allowing for easier navigation through the document.


Fig. 2. A clickable mind-map giving an overview of the categories (subsections) and specific topics (subsubsections) discussed in Section V. References in this mind-map were chosen for inclusion when explicitly mentioned in the portion of text represented by each node, or when deemed relevant to the category, based on a full-text review.

Finally, in Section V-J, we close this section with a summary and an analysis of the obtained results.

Except for Section V-A and Section V-J, every section contains a table relating the most important security requirements of that category to a collection of works that we deemed the most relevant to these topics. Additionally, every table shows the research interest (low, medium, high, very high) of the scientific community for each security requirement in that category. This interest is inferred from the percentage of works identifying or addressing the specific security requirement compared to all the (unique) papers related to that category. The number of papers addressing a specific category is taken from Figure 2 as the number of papers appearing in the corresponding level 1 (i.e., subsection) and all level 2 (i.e., subsubsections) nodes, but removing duplicates. For instance, the total number of papers discussing Network Security is given by the count of the references appearing in Figure 2 for the nodes Network Security, Latency and timeliness, Availability, and Wireless, without duplicates. It is important to note that a number of works identify multiple security requirements, thus, appear in multiple subsections; as such, the calculated percentages do not represent disjoint partitions of the set of investigated works, thus their sum

TABLE IV
INTEREST LEVELS ASSIGNED TO EACH SECURITY REQUIREMENT IN RELATION TO ITS CATEGORY.

| Relative interest | Range (x) |
|---|---|
| Low | 0% ≤ x ≤ 7% |
| Medium | 7% < x ≤25% |
| High | 25% < x ≤45% |
| Very High | 45% < x ≤100% |

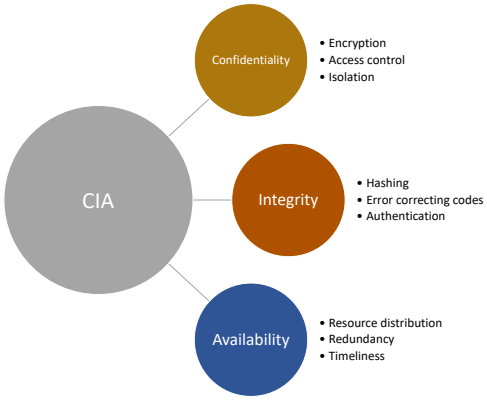



Fig. 3. The CIA triad, with some examples for each property.

will not result in 100%. The range of percentages assigned to each interest level are shown in Table IV and have been chosen based on the distribution of percentages assigned to security requirements across all categories. As an example of the interest level, consider a category AB discussed by 50 papers, and a security requirement AB-01 identified by 5 of these 50 papers, the research interest for the requirement AB-01 will be *medium*, with a percentage of 10%. The aim of these tables is to give the interested reader a stepping stone to more in-depth works for each requirement, but also the topics itself.

### A. The CIA Triad

The Confidentiality, Integrity, Availability (CIA) triad is a well-known information security model, and can be considered as a set of extremely abstract security goals or requirements. A subset of these lie at the root of every other security requirement. Figure 3 provides a graphical representation of the triad and shows some examples of solutions related to each property. We briefly describe the three properties as they are described by [130] below:

- **Confidentiality** pertains to protecting information in all its forms. This includes data encryption, access control, network isolation, but also privacy aspects.
- **Integrity** concerns consistency, accuracy, authenticity, and more generally the overall trustworthiness of entities.
- **Availability** concerns operational guarantees of the system. This covers topics such as redundancy and decentralization, but also guarantees that tasks will be performed within hard deadlines.

Typically, the CIA triad is used in information security, meaning that the three properties relate to information only. However, it is equally applicable in other domains, such as cyber-physical systems [132]. Indeed, many of the works we investigated explicitly mention the triad (e.g. [25], [63], [89], [97]). Traditionally the focus in industrial environments has been first on availability, second on integrity, and last on confidentiality. However, with Internet-connected systems, this requires reconsideration, and all three aspects should be brought up to an acceptable level. Thus, with the development of new IIoT and Industry 4.0 solutions, confidentiality and integrity should be weighed equally to availability.

While these three aspects are a very good starting point and are certainly important to keep in mind when specifying the security goals for any system, it is not always useful to reduce concrete requirements back to elements of the CIA triad, if one already has more (e.g. contextual) information that might help with deriving an unambiguous security goal. For example, it is easy to state that data at rest should be kept confidential, but such a requirement does not convey the conditions that a confidentiality mechanism should satisfy. Moreover, it leaves a lot of room for interpretation (e.g., confidential to which parties?). On the other end of the spectrum, very fine-grained requirements are only possible if one is developing for a specific scenario.

In the next subsections, we strive to find a middle ground where we describe security requirements at a high enough level to see where the challenges in achieving them lie, but at the same time refrain from going too deep into any scenario, although we might refer to them as anecdotal evidence supporting the legitimacy of a requirement.

### B. Authentication

Authentication of remote entities (both humans and machines, or even applications) is a key concern for many forms of IoT communication [31]. Within the context provided by IoT and IIoT applications, this brings some extra challenges [15], [17], [49]. There is a need for extremely lightweight authentication schemes, with little overhead in terms of computation time and transfer size, among other things.

A second but very important concern is verifying the integrity and authenticity of data, e.g. to ensure that a configuration file was created by an authorized party, and not modified since. Also here, the IIoT domain has special requirements that prevent the adoption of commonly used authentication mechanisms. Many topics in this section therefore also concern integrity, albeit not explicitly mentioned in every instance.

Wang and Wang [82] name some other typical challenges (mainly aimed at wireless industrial communication) that need to be taken into account when investigating authentication and integrity methods. They consider extreme resource constraints, the open broadcast nature of wireless communications (i.e. anyone can read and send messages on certain frequencies), extremely large network sizes, and lack of infrastructure support.

TABLE V
AUTHENTICATION-RELATED SECURITY REQUIREMENTS, SOURCES THAT IDENTIFY THESE, AND THEIR INTEREST LEVEL RELATIVE TO THE CATEGORY. THE RELATIVE INTEREST LEVEL IS BASED ON THE PERCENTAGE OF WORKS ADDRESSING THE SPECIFIC SECURITY REQUIREMENT COMPARED TO THE TOTAL NUMBER OF PAPERS FOR THAT CATEGORY.

| ID | Security requirement | Related sources | Relative interest | % within category |
|---|---|---|---|---|
| A-01 | multi-factor authentication | [24], [26], [72], [73], [77], [78], [82] | Medium | 9% |
| A-02 | key distribution | [29], [53], [55], [59]–[63], [67], [85] | Medium | 13% |
| A-03 | node addition, revocation, rekeying | [29], [30], [63], [67], [82] | Low | 6% |
| A-04 | decentralized key management | [24], [53], [55], [62], [67] | Low | 6% |
| A-05 | transitive authentication | [62] | Low | 1% |
| A-06 | mutual authentication | [28], [48], [52], [70]–[74], [76]–[78], [81], [87] | Medium | 17% |
| A-07 | privacy-preserving authentication | [72], [77], [78], [87], [93], [94] | Medium | 8% |
| A-08 | minimization of user interaction | [70]–[72], [76]–[80] | Medium | 10% |
| A-09 | non-repudiation | [84], [86], [89] | Low | 4% |
| A-10 | attestation | [17], [39], [95], [97] | Low | 5% |

As an example of authentication challenges in existing systems, we consider the Message Queue Telemetry Transport (MQTT) protocol. This is a widely deployed protocol for data exchange in the industrial domain, and features some very basic and insecure authentication methods [57]. According to Katsikeas et al. [34], the protocol allows authentication through a simple username and password combination, which are communicated in plaintext. A second authentication method sometimes used is through a unique client identifier, which is easily spoofable. While it is possible to secure these methods by complementing MQTT with Transport Layer Security (TLS) or IPSec, those two protocols are too resource-intensive for many IIoT applications, and lighter alternatives are necessary, such as TinyTLS [56] or DTLS [58]. Now that industrial networks are becoming increasingly connected to the internet, this becomes more and more important.

The importance of sufficiently secure authentication mechanisms is reflected by the fact that positions 2 and 3 in the Open Web Application Security Project (OWASP) IoT Top 10 [47] on IoT vulnerabilities concern attack vectors where (a lack of) authentication is an important aspect. Its importance is also underlined by the popularity of this topic with recent research efforts, with many papers addressing or identifying the above issues (see Table V-B). These works identify several authenticity properties that can be considered requirements in various use-cases in the IIoT domain. We describe these in more detail in the following subsections.

For a comprehensive survey on IoT authentication algorithms, we refer the interested reader to Ferrag et al. [31]. The authors cover many authentication algorithms and compare them based on computational efficiency, threat protection, and more. Kail et al. [33] provide another survey covering multiple industrial protocols aimed specifically at Low Power Wide Area Network (LPWAN) technologies.

Next, we discuss a number of authentication-related topics in the following subsections. First, we look at key distribution, after which we discuss mutual authentication and multi-factor authentication. Then, we address non-repudiation as a requirement, followed by anonymous authentication and privacy preservation in authentication algorithms. As a final topic, we discuss attestation techniques through trusted hardware.

*1) Key distribution:* Key distribution is a challenging requirement for many applications in the IoT [68], and naturally extends to the IIoT. With devices being set up and used in hostile environments, possibly being very mobile, dynamically joining and leaving networks, and possibly being very constrained in resources, there is a pressing need for efficient, flexible, and dynamic key management mechanisms. Airehrour et al. [59] argue that traditional Public Key Infrastructure (PKI) is outdated, stating that "it was at no time designed to handle the complications of managing industrial-scale networks of 50 billion devices that IoT promises to usher in.". This raises the question of whether all IoT devices should exist in the same authentication domain, and if centralized authentication authorities such as PKIs are even a sensible choice for that many devices. We will not attempt to answer these questions here.

In order to deal with dynamic environments, some naturally implied requirements for key management solutions are that they can handle node addition, revocation, as well as rekeying [63]. Resource-constrained devices will have issues with key generation, computationally intensive algorithms, and transmission of large/many messages. Moreover, in an industrial setting, device owners might not trust the manufacturer to generate keys for them, and will want to do this themselves [67]. Availability can be an issue as well. In Critical Infrastructure (CI) environments, an authentication authority has to be reachable at all times. Because of this, Blanch-Torne et al. [62] state that it is not sufficient to rely on one central authority for authentication. Additionally, they also identify transitive authentication (if A knows B and B knows C, B can introduce A to C) as a requirement in some scenarios.

In [63], the authors propose a key management solution that aims for little transmission overhead by requiring only one transmitted message for one-way-authentication. While this makes for an energy-efficient protocol, it appears to not be very scalable or dynamic, since all nodes need to be known beforehand, and addition, revocation and rekeying are not thoroughly discussed.

Ulz et al [67] propose a Bring Your Own Key (BYOK) approach, to address the trust issue between device owners and manufacturers. However, it does require devices to have Near Field Communication (NFC) capabilities, and key distribution requires a human to physically move between a central server and the device.

Another approach is suggested by [62], where there is no centralized authority, but a Distributed Hash Table (DHT) that takes care of identity propagation and lookups. Their solution is a distributed one, and also provides transitive authentication. It is scalable and dynamic, but the protocol is not designed with energy-efficiency in mind, and can require a considerable number of messages at times.

While the above-mentioned sources address the identified requirements to some extent, none of them addresses multiple at once. Clearly, there is still plenty of opportunity for novel research in this area. One potential solution to several key distribution challenges that might become viable in the future, is quantum key distribution [65], [66]. In such a system, it is impossible to eavesdrop on a transmission without altering its payload, meaning that any eavesdropping attempt can be detected.

Blockchain technologies are another promising candidate, showing potential to overcome several key challenges. Bartolomeu et al. [61] discuss Self Sovereign Identity (SSI) techniques for IIoT, which build on top of blockchains to provide Decentralized Identifiers (DIDs). These systems have as a property that all entities carry their own identification data, eliminating the need for a centralized root of trust. They discuss the challenges faced by several frameworks capable of providing DIDs, some of the most prevalent being the need for a common data model for interaction between parties, and a lack of research in their application to Machine-to-Machine (M2M) authentication. A different approach is taken in [60], where the blockchain-based BCTrust protocol is extended with key management functionality. One challenge with blockchain is that due to the immutability of blockchains, revocation or alteration of data is impossible. The standard solution is to add append modifications at the end of the chain, but there exist some early results showing that small scale changes are possible using Chameleon hashing schemes. This comes at the cost of some security [64], but further research is needed.

*2) Mutual Authentication:* In [82], mutual authentication is identified as one of the requirements for any practical authentication scheme, and Kolluru et al. [76] state that mutual authentication between any two IoT devices is necessary, as many of them are exposed to external environments. Moreover, because of this many-to-many requirement, a user/password system is neither user-friendly nor flexible enough. It is also difficult to handle in dynamic environments. They thus identify the need for authentication mechanisms that can be used between any pair of devices, with minimal user interaction. Autenrieth et al. [71] even state that fully automated mutual authentication is a requirement. Some recent work that aims to facilitate this is done in [74], and uses trusted components such as Physically Uncloneable Functions (PUFs). PUFs are functions implemented in hardware in a way that aims to make them very hard to copy, thus being able to act as a device "fingerprint". Another way to facilitate M2M authentication in settings where the participating devices are geographically nearby, is by using physical context such as luminosity or temperature. Loske et al. [79] survey the available literature on this so-called context-aware authentication. If the transmissions are wireless, devices can also be identified through their radio frequency fingerprint [81].

One way of minimizing interaction is by relying on biometrics for identification and authentication (although one should be careful to not use biometrics for authorization). One property of biometric-based authentication schemes is that they cannot be used for M2M authentication, as biometrics are always derived from living beings. Therefore, these types of protocols might not be feasible in every industrial context, although they adapt well to some (e.g. smart healthcare [73]). In [72], a two-factor mutual authentication method is proposed, combining smart cards and biometrics, although recent work shows that their protocol is not secure against various attacks [75]. Li et al. [77], [78] use a combination of user/password and biometrics instead as a two-factor approach, while Deebak et al. [73] combine smart cards, passwords, and biometrics. The proposed methods claim to be very lightweight – but reliance on biometrics by itself requires specialized hardware (or some non-trivial computational capacity to process e.g. audio or video signals), which might not always be an option. Further, it typically requires physical proximity, although recent work [80] shows that remote biometric authentication is a possibility.

Another way to minimize user interaction is by deriving identities through analysis of behavioral patterns. This shares the property that it cannot be used for M2M authentication with biometric-based methods. The Fifth generation cellular network technology (5G) authentication scheme for smart devices proposed in [70] uses Cloud-based learning to dynamically identify and authenticate users based on behavioral patterns, showing another approach for minimizing user interaction. This concept has also been used in the field of intelligent vehicles whereby drivers are identified by their driving behavior [69].

*3) Non-repudation:* Non-repudation is a message property, ensuring that the author of a message is not able to later repudiate (i.e. deny) their authorship of that message. Non-repudiation can also extend to concepts other than messages (e.g. an entity cannot repudiate their accountability for an action that was started/requested by them).

Fraile et al. [84] provide some concrete examples showing why non-repudiation can be considered a security requirement. Firstly, users might perform illegal actions, and the system needs a way to track these actions. If these actions are reputable, the system becomes susceptible to log injection attacks, an observation echoed by Ankele et al. [83]. Another example mentioned is the situation where a manufacturer finds out that their configuration files on some hardware have been deleted, after the hardware vendor has performed updates to this system. Without a non-repudiation mechanism in place, the deletion of these configuration files cannot be unambiguously traced back to the software update. Another example can be found in [89], where the challenges in applying the Assurance Case methodology for the IIoT are laid out. Assurance Cases are structured arguments, for use during e.g. software development, that show that certain properties of a system hold. The authors of this work identify non-repudation as a requirement for the assurance of security properties of a system. The blockchain-based authentication

and access control scheme described in [87] also states that non-repudiation is an essential property.

Li et al. [86] propose a certificateless authentication scheme for Wireless Sensor Networks (WSN) environments. The advantage of their approach is that, because some of the heavier computations can be moved to third parties (e.g. a gateway), the computational requirements on sensor nodes themselves can remain low. Their protocol achieves non-repudiation by ensuring that messages are publicly verifiable. Certificateless schemes are a fairly popular topic in this domain. More recent examples of work on similar schemes for IIoT are [85], [88], [92] (broken in [91]), and [90].

*4) Anonymity and Privacy:* Anonymous authentication is verifying the authenticity of an entity without disclosing that entity's identity. This is necessary in situations where one wants to protect the privacy of users. Lin et al. [87] identify the need to protect users from being identifiable when an adversary has access to the authentication service. Cui et al. [93] also mention privacy-preserving access control. One example of a threat due to lack of anonymous authentication is provided by [72]: an adversary could conduct traffic analysis to create profiles on sensitive assets in an industrial environment, and possibly derive sensitive data from those profiles. Paliwal [94] proposes a hash-based privacy preserving authentication scheme specialized for WSNs scenarios. In this work, a variety of requirements are identified for schemes for WSNs, although these mostly relate to low-level properties that generalize to any secure authentication scheme, such as resistance against replay attacks. Because of this, we consider these to be too low-level to be included in our analysis as is, but rather as implied by other requirements.

In [87], a public blockchain-backed authentication mechanism is proposed, thereby turning user anonymity into a hard requirement. The work in [93] does not rely on a blockchain, but relies on a server to provide computational aid (in a secure manner). While both proposed schemes use Attribute Based Signatures (ABS) as cryptographic constructs, the two approaches cater to different goals: blockchains are widely considered to be resilient and highly available systems, which can be useful in scenarios that require these aspects, while server-aided encryption schemes target low-power devices with very limited computational ability or battery life.

*5) Attestation:* Attestation is a method for detection of unintended and malicious changes to software [17]. Doing this remotely can provide guarantees on the integrity and authenticity of a piece of software that is being run on a remote system, and therefore allows one to place more trust in a remote system than is possible in a scenario without remote attestation.

Because attestation aims to enable these higher levels of trust, it poses very strong security requirements on hardware. At the same time, remote attestation methods implemented purely in software typically have to rely on very strong assumptions that are hard to achieve in practice [17]. Attestation can be done in a practical setting through the use of Trusted Execution Environment (TEE)s provided by trusted hardware, such as ARM TrustZone [247], Intel SGX [248], or implementations of the Trusted Platform Module (TPM) standards [249]. Not all of these might run on low-end hardware, but some recent embedded controllers contain trusted hardware components [250] that also enable attestation to some extent.

References [17], [95], and [97] all identify the need for remote attestation, in order to increase the system's resilience against intruders. Especially in contexts where parts of an overall system are deployed in hostile environments, where it is important that the correct functioning of the software is continuously verified. Additionally, Laaki et al. [96] also identify the possibility for hardware attestation to protect the digital twin representation of proprietary hardware setups.

As mentioned in [17], there has not been a lot of activity on trusted hardware in this domain as of yet, with most of the available attestation protocols proposed so far aiming for a more general-purpose scenario, not taking into account aspects that make integrity and authentication protocols for the IIoT a challenging domain.

### C. Access Control

Access Control (AC) is necessary in a wide variety of situations; already when a device allows for two modes of interaction, one for normal user behavior and one for system administrators to deploy updates, a rudimentary form of access control is needed. Furthermore, a lack of adequate privilege separation has been identified as one of the most severe shortcomings in existing systems, such as the Supervisory Control And Data Acquisition (SCADA) protocol [100].

AC invariably relies on authentication methods, as one needs to authenticate users in order to enforce access policies. It is therefore not surprising that AC mechanisms inherit many of the authentication requirements described in Section V-B. The challenges in access control relate to resource consumption, but also availability. In highly distributed scenarios, it should not happen that AC policies are unavailable due to a connection failure.

Aiming to minimize energy consumption for lightweight devices, Li et al. [86] propose a certificateless signature scheme as well as an AC framework for WSNs. This is made possible by relying on a (collection of) trusted systems in the network that are powerful enough to perform a part of needed cryptographic operations. The lightweight devices then cooperate with the trusted systems to create cryptographic signatures. Some natural security requirements are mentioned, such as the CIA triad and non-repudiation. Beltran et al. [24] also target low-power devices, but they explore a setting in which these resource-constrained systems interact with Cloud services. In this scenario, they identify the need for identification, authentication, authorization, and accounting mechanisms. Furthermore, they state that depending on the particular application, fine-grained authorization control might be needed, or the ability to handle dynamically changing privileges. In some other situations, they state it is useful to manage access policies centrally. However, in order to be compatible with many systems from different developers, some form of federation is needed too. In order to address these issues, they propose a token-based federated authentication scheme that makes use of PUFs to meet the energy constraints of

TABLE VI
ACCESS CONTROL-RELATED SECURITY REQUIREMENTS, SOURCES THAT IDENTIFY THESE, AND THEIR INTEREST LEVEL RELATIVE TO THE CATEGORY. THE RELATIVE INTEREST LEVEL IS BASED ON THE PERCENTAGE OF WORKS ADDRESSING THE SPECIFIC SECURITY REQUIREMENT COMPARED TO THE TOTAL NUMBER OF PAPERS FOR THAT CATEGORY.

| ID | Security requirement | Related sources | Relative interest | % within category |
|---|---|---|---|---|
| AC-01 | handle dynamic changes | [24], [99], [104], [108] | Medium | 25% |
| AC-02 | fine-grained AC | [24], [53], [99], [104], [106] | High | 31% |
| AC-03 | centralized AC | [24], [86] | Medium | 12% |
| AC-04 | decentralized AC | [24], [87], [99], [102]–[104], [107], [109] | Very High | 50% |
| AC-05 | privacy-preserving AC | [87], [102] | Medium | 12% |
| AC-06 | transparency | [86], [87], [103] | Medium | 19% |
| AC-07 | compatibility | [107] | Low | 6% |

low-power devices. The resulting authentication scheme is flexible enough to act as a building block for many types of authorization mechanisms.

The blockchain-based authentication protocol proposed in [87] also contains an AC framework, and tackles the availability and single point of failure challenges through use of a blockchain, and a DHT containing AC policies. An additional feature of this work is that it respects the privacy of users through the use of ABS techniques. A different approach is taken by He et al. [102]. In this work, ring signatures are used to construct a distributed lightweight AC framework. This framework specifically targets WSNs and achieves user anonymity by grouping users with similar rights, ensuring that AC authorities cannot differentiate between signatures from users in the same group. Lahbib et al. [104] also propose a blockchain system, identifying the need for dynamic access control and distributed governance. They utilize smart contracts and leverage the non-repudiation and integrity inherent to blockchain systems to propose a resource management framework, with fine-grained AC built in. Yao et al. [109] share the sentiment that distributed AC is needed, but propose a Fog solution based on attribute credentials.

Kim et al [103] consider a scenario where nodes in multihop Low-power Lossy network (LLN)s want to communicate with each other. They also identify the need for federation, but from a reliability perspective. In order to guarantee the availability of a system, it cannot rely on a single point of failure for access control enforcement. At the same time, they identify the need for a transparent scheme, that is also scalable. Decentralized protocols such as the one proposed in their work, can increase scalability, as changes are propagated much more organically through the network, than with a centralized structure, avoiding congestion issues.

In the work presented by Chen et al. [99], AC and authorization are also identified as one of the major challenges for the IIoT. They propose an access control framework for a scenario where the owner of an IIoT device has the right to control the AC policies of their device, and wants to set up fine-grained policies. At the same time, a large number of IIoT devices are shared by multiple entities that can interact with them based on these policies.

Preuveneers et al. [107] argue that identity management is crucial for AC purposes, and propose a framework handling identities, authentication, and authorization in a networked production scenario. They also raise the point of compatibility with legacy devices, which is worth considering in any IoT environment.

Vanickis et al. [108] make the observation that due to the increase in frequency and sophistication of security attacks in recent years, there is a need to include risk assessment in the process of specifying AC policies, and that as a result of these trends there is a growing interest in Zero Trust Networking (ZTN) protocols as opposed to perimeter-based security. The principle behind ZTN is to treat the intranet with the same level of trust as the Internet. Their proposed policy enforcement framework is built upon this principle, and is able to provision firewalls across different segments of a network.

### *D. Maintainability*

Maintainability concerns the ability to configure, reconfigure, and update (parts of) a system. In Industry 4.0, these concepts become crucial as the software and configuration of IIoT systems must have the ability to be changed, in order to provide protection against previously unknown security threats [116]. Updateability can be considered a countermeasure against security attacks, since it allows for continuous changes to firewall configurations as threats are identified, as well as software patches for newly discovered software vulnerabilities. As we will see in this section, the challenges relating to maintenance are again related to resource constraints and the dynamism of IIoT environments, making traditional maintenance solutions insufficient to adequately address the needs in this domain.

In [110], George et al. state that the availability of security updates is a critical concern for IIoT devices, but that due to some IIoT systems being so lightweight and the infrastructure not being fixed, it is extremely difficult to always patch all devices in a network. To mitigate this, they describe an approach that ensures update deployment on high-risk vulnerabilities, to reduce the risk of serious attacks on the infrastructure. For this, they propose a number of risk mitigation strategies that can be used to help identify the devices most in need of updates. Yadav et al. [114] also identify the timely application of patches to all vulnerable systems in a network as a problem, and propose a patch prioritization method to mitigate this.

In addition, some IIoT systems require the ability to be updated without any disturbance to the service they provide. Mugarza et al. [111] propose a secure updating mechanism for mixed-criticality systems. However, their approach requires the ability to run and monitor updated binaries in a sandboxed mode. Not every device has the resources for this. They follow

TABLE VII
MAINTAINABILITY-RELATED SECURITY REQUIREMENTS, SOURCES THAT IDENTIFY THESE, AND THEIR INTEREST LEVEL RELATIVE TO THE CATEGORY. THE RELATIVE INTEREST LEVEL IS BASED ON THE PERCENTAGE OF WORKS ADDRESSING THE SPECIFIC SECURITY REQUIREMENT COMPARED TO THE TOTAL NUMBER OF PAPERS FOR THAT CATEGORY.

| ID | Security requirement | Related sources | Relative interest | % within category |
|---|---|---|---|---|
| M-01 | software updateability | [52], [111]–[113] | High | 27% |
| M-02 | configuration updateability | [52], [67], [111] | Medium | 20% |
| M-03 | disturbance-free updates | [38], [111] | Medium | 20% |
| M-04 | usability of update process | [113] | Low | 7% |
| M-05 | traceability | [36], [113] | Medium | 13% |
| M-06 | compatibility | [36], [113] | Medium | 13% |
| M-07 | transparency | [113] | Low | 7% |
| M-08 | secure status transfer | [36]–[38], [117] | High | 27% |

up on this research with an application of their system to a smart city scenario [112]. The proposed update process is in accordance with several safety standards, a requirement identified in Section V-E.

According to Seitz et al. [113], updating IIoT systems is often complex and cumbersome, and requires an expert technician to perform the update, which can be a lengthy process. This does not scale with the increase in connected devices, and therefore the update process must be streamlined and simplified, with minimal possibilities for errors due to human behavior. Their suggestion is a marketplace, not unlike those seen on smartphones. In addition to usability, they state that update management of devices should be possible both on-site and remotely, and updates and installations must be logged so that they are traceable, for transparency and in case of problems. While their proposed solution appears as a global and decentralized marketplace, this might not be a good fit for every type of device, especially when the functionality of such a device is secret. Moreover, it raises questions about how much power can be given to app developers and where the trust in a system should lie, which are topics that can be highly dependant on a specific scenario, and are worthy of further investigation on their own.

Another problem is mentioned by Ulz et al. in [67], wherein they state that cryptographic keys also require the possibility to be updated securely. This can be interpreted as a requirement relating to the maintainability of a system's configuration, and is an argument against the deployment of hardcoded keys at manufacturing time, which sometimes happens in production environments. In a later work, they take this notion further, and propose a hardware device, that can be temporarily attached to a system to allow for secure updating and reconfiguration [52]. The updates are verified and installed in an isolated environment provided by the special hardware, for increased security and traceability, but still allows for remote queuing and deployment of updates, to some extent. However, this approach might not be practical in environments where it is hard to physically reach all deployed systems.

*1) Smart maintenance:* Industry 4.0 enables smart maintenance, which is essentially predictive maintenance of (parts of) devices based on remote data collection about their usage. This allows for a more streamlined production line where system downtime and maintenance costs are reduced to a minimum. Its relevancy is underlined by the inclusion of continuous maintenance and maintenance frequency being used as measurable safety indicators in a meta-model proposal for automated security dependability detection within IIoT systems [25]. Priller et al. [117] provide a case study on this subject detailing a number of smart maintenance security requirements, notably the ability to update as well as secure communication channels themselves.

Lesjak et al. [36] reason that smart maintenance requires secure communication channels, as status information of machines is sensitive data. Moreover, the maintainer needs the ability to verify the validity of this data. They argue that there are systems for which it is essential that they are exposed to the Internet as little as possible, and propose solutions using NFC to permit secure transmission of data to the maintainer, as well as identity provisioning over NFC [37]. The specific requirements identified in this work are the need to support legacy devices, prevent data leakage, protect against Internet access, protect the validity of the maintenance data (towards the maintainer), and protect transparency of the communicated data (towards the customer). In a later study, Lesjak et al. [38] propose an MQTT-based approach where they add a further requirement that data transmission must not cause safety-critical interference, so that operational functionality remains unaffected.

### *E. Resilience*

The Industrial Internet Consortium (ICS) has published an IIoT security framework [123] in which they define resilience as "the emergent property of a system that behaves in a manner to avoid, absorb and manage dynamic adversarial conditions while completing the assigned missions, and reconstitute the operational capabilities after causalities". This definition overlaps with several aspects of system trustworthiness such as safety and reliability, but also security. Indeed, [15] and [118] identify resilience as an important security challenge for the IIoT. The implication that resilience requirements bring to the security domain are that security technologies should provide the capability to continue normal system operations if parts of the system are considered compromised. This could for example be done by rerouting tasks to other capable components, or through other means, often belonging to one of three canonical approaches identified by Laszka et al. [126]: redundancy, diversity, and hardening.

The manner in which this requirement should be satisfied, depends heavily on the scenario. In a WSN, it might be

TABLE VIII
RESILIENCE-RELATED SECURITY REQUIREMENTS, SOURCES THAT IDENTIFY THESE, AND THEIR INTEREST LEVEL RELATIVE TO THE CATEGORY. THE RELATIVE INTEREST LEVEL IS BASED ON THE PERCENTAGE OF WORKS ADDRESSING THE SPECIFIC SECURITY REQUIREMENT COMPARED TO THE TOTAL NUMBER OF PAPERS FOR THAT CATEGORY.

| ID | Security requirement | Related sources | Relative interest | % within category |
|---|---|---|---|---|
| R-01 | continuation of operation with compromised subsystems | [15], [84], [118], [121], [126] | High | 31% |
| R-02 | operation with intermittent connectivity | [84], [125] | Medium | 12% |
| R-03 | standards compliance | [25], [112], [119], [120], [127] | High | 31% |

acceptable to simply deploy enough sensors to guarantee some redundancy, meaning that a small number of compromised sensors can be kept contained and their output discarded until the issue has been addressed. In a power plant however, it might be catastrophic to disable one generator entirely if one of its components has been compromised. Instead, it might be possible to provide the compromised components' functionality in some other way, or temporarily reroute energy from other generators to guarantee some level of operations.

Fraile et al. discuss device driver security in a connected virtualized factory environment [84]. They identify multiple resilience-related issues, one being that intermittent connectivity might cause loss of history if status information should be continuously sent to a centralized database or the Cloud. Their proposed solution is to keep local databases that keep a short-term history that can be synchronized with a back-end once connectivity is restored. Another identified issue is to avoid system failure, in case of a compromised device driver. The authors propose redundancy and smart fallback mechanisms to adapt to possible threats. The difficulty in a fallback mechanism is that it requires the exact same configuration and as much as possible of the current system state of the normal system, in order to allow for rapid recovery. This is not only difficult because state replication can introduce considerable overhead, but it also means that the fallback system is vulnerable to the same threats as the normal system. To mitigate this issue, the authors propose introducing some diversity in the fallback system. The proposed solutions in this work are all rather specific to the considered scenario and architecture, but use elements that are common in resiliency mechanisms in general.

When looking at low-energy devices, WSNs have been identified as a way to increase the robustness of SCADA systems against network failures, due to their distributed and self-organizing nature [125]. However, major concerns exist regarding their ability to communicate securely, and the ability to interface with some proprietary SCADA protocols. The authors also identify a number of challenges relating to the security of WSNs and propose a decentralized multi-agent architecture to remedy a number of these.

In [25] and [120], a number of measurable indicator points are identified, among which those relating to resiliency. In the latter, they then use these indicator points to propose a method for automated standard compliance testing in the Industry 4.0 domain. Standard compliance is a powerful aid in verifying the resilience, reliability, and safety of a system, and can be applied to a wide spectrum of devices. Related to standards compliance, Bauer et al. [119] investigate European Union Agency for Network and Information Security (ENISA) guidelines on secure Cloud services, and extract a number of measurable security metrics that relate service level agreement objectives between Cloud providers and their (industrial) customers to concrete responsibilities. These metrics could also be used in compliance testing. In this work, reliability and redundancy are also identified as measurable indicators. Similarly, Leander et al. [127] investigate the applicability of the IEC 62443 cybersecurity standards [124] in Industry 4.0 applications. For a short survey on the security standards relevant to Industry 4.0, we refer to [122].

### *F. Data security and data sharing*

In today's world, data security is critical in nearly any digital environment, and the IIoT is no different. Many of the works investigated in this survey identify confidentiality of data as a security requirement in some form (e.g [17], [36], [49], [71], [133], [136]). Traditionally however, availability and integrity are considered more favorable than confidentiality for industrial environments [129], [132], as they have a measurable economic impact. This is not a sustainable viewpoint in an era of connected devices, and is changing fast now that companies seek to connect their systems to the Internet.

In a survey among companies, Autenrieth et al. [71] found that they too consider data security to be one of the critical factors for migration to Industry 4.0, a finding confirmed by another study conducted by Moyne et al. [136]. In this work, the authors additionally state that companies are hesitant to adopt data-sharing based technologies (Cloud, smart maintenance, fault detection and prevention, etc.) as there is no evidence of these technologies being safe or secure when it comes to protecting intellectual property, as a result of which they identify the need for a standardized way to achieve intellectual property protection in the presence of data sharing mechanisms. The sentiment that companies are reluctant to rely on Cloud providers for data storage and sharing is shared by Esposito et al. [29]. However, they also note that most data breaches come from within companies, and not Cloud providers. They propose a cloud storage solution that aims to minimize the attack surface both in the Cloud and within the company. They identify data loss mitigation as another requirement, identifying four key elements for an effective solution: prevention, identification, notification, documentation.

The challenges in this domain relate to three colliding factors: Firstly, due to the heterogeneity of devices, data security mechanisms need to be able to operate with extremely few resources. Secondly, due to the criticality of some IIoT applications, the data security requirements are very high.

TABLE IX
DATA SECURITY AND DATA SHARING RELATED SECURITY REQUIREMENTS, SOURCES THAT IDENTIFY THESE, AND THEIR INTEREST LEVEL RELATIVE TO THE CATEGORY. THE RELATIVE INTEREST LEVEL IS BASED ON THE PERCENTAGE OF WORKS ADDRESSING THE SPECIFIC SECURITY REQUIREMENT COMPARED TO THE TOTAL NUMBER OF PAPERS FOR THAT CATEGORY.

| ID | Security requirement | Related sources | Relative interest | % within category |
|---|---|---|---|---|
| DSS-01 | data loss mitigation | [29] | Low | 2% |
| DSS-02 | data confidentiality | [36], [50], [52], [53], [55], [98], [105], [130], [135], [139], [149] | Medium | 19% |
| DSS-03 | standardization | [136] | Low | 2% |
| DSS-04 | secure data transport | [34], [38], [40], [137], [142], [143], [145] | Low | 7% |
| DSS-05 | secure external data storage | [29], [65], [66], [98], [131], [139], [141], [146]–[156], [159], [161] | High | 34% |
| DSS-06 | data flow control | [162]–[164] | Low | 5% |
| DSS-07 | data protection legislation compliance | [50], [98], [165] | Low | 5% |

Thirdly, many smart capabilities are enabled by the sharing of data, but in industrial contexts, data is often sensitive and confidentiality is of utmost concern, which poses a dilemma.

Data security covers a wider area than just encryption techniques. One of the vital aspects of the Industry 4.0 paradigm is making smart use of available data. This inevitably involves sharing data with other entities, that can be anywhere from a part of the system to being outside the organization boundaries. As an example, consider the discussion on the sharing of device usage metrics in Section V-D1. Even if no other data is shared, usage metrics will have to be sent to the device manufacturer to enable smart maintenance, but might also be used to deduce sensitive information such as production volume. A similar example would be data analysis for anomaly detection (Section V-G). While encryption techniques do offer ways to aid with partial sharing of data, we will also discuss other ways of keeping data confidential.

*1) Data transport:* The MQTT protocol is widely used for data sharing between industrial systems, but by itself only supports user/password authentication, and provides no security measures on the network or application layer. This becomes problematic especially in the context of the IIoT. In order to remedy this, Lesjak et al. [38] propose using TLS as a secure layer upon which MQTT can function. While this provides all the security benefits of TLS, it does add considerable overhead to the edge devices that will now have to manage TLS contexts. In their work, the authors propose using a trusted hardware extension at the edge devices that can store keys and also manage the TLS context. While modern devices might have access to cheap trusted hardware, this is not always possible with legacy devices, therefore, other solutions will need to be investigated. Katsikeas et al. [34] also observe that TLS can be used to secure MQTT communication, but note that this will not work well in WSNs due to severe resource constraints. Therefore, they try to minimize the overhead by encrypting messages at the link layer. In a later work, Lesjak et al. [40] observe that an often-needed requirement is communication with other stakeholders, e.g. equipment manufacturers (for smart maintenance) or nearby links in a supply-chain. To enable authenticated, secure data communication between these, the authors propose a hybrid multi-stakeholder protocol on top of MQTT that allows end-to-end encryption of payloads that need to be transmitted to external parties.

Alternatively, more modern protocols such as The OPC Unified Architecture (OPC UA) [145] have authentication and encryption support [34], [143], and hardware acceleration for the cryptographic primitives used in these is starting to appear in lightweight products [251]. Adoption of the OPC UA could thus help in meeting some of these constraints. One recent experimental deployment combines this with trusted hardware to facilitate secure connections [142], but acknowledges that further research is needed. Finally, it is worth noting that regardless of the security protocol used, from an energy and efficiency standpoint, there is a case to be made for selectively encrypting only those messages that might harm the system if tampered with. In [144], the authors propose a symbolic analysis model that can identify such messages.

*2) External parties:* Data confidentiality when at rest or in transit, is often realized through cryptographic means. The challenges in finding suitable ciphers for the very diverse IIoT environment are described by Zhou et al. in [55]. Again, the main challenges appear to concern energy and other resource constraints. Irrespective of the cipher used, the authors also identify the key distribution and management problem, as previously discussed in Section V-B1. More generic challenges are described by Yu et al. [160]. They argue that Reliable storage, convenient usage, efficient search, and trustworthy data deletion are some of the major issues for Cloud and Fog scenarios.

As the Cloud promises a large amount of storage and computational resources, Cloud connectivity is often necessary for Industry 4.0 applications. With a suitable encryption scheme, data might be stored securely in the Cloud [98], but even then it is not possible to interact with it in any way other than retrieving it for decryption. Seeking to remedy this, there has been a recent increase in research efforts in modern cryptographic techniques such as homomorphic encryption, allowing for computation on encrypted data ( [65], [66]), and searchable encryption, enabling search operations on encrypted data ( [146], [161]). Specific to the IIoT data sharing scenario, Deng [147] proposes an anonymous aggregate encryption system that allows IIoT devices to encrypt data into one ciphertext that can be decrypted by multiple recipients with their individual keys, while retaining their relative anonymity.

Fu et al. [149] propose one way of ensuring confidentiality in the Cloud, while maintaining the ability to search through data sets, through a privacy-preserving encryption scheme.

Deployed IIoT devices transmit their data to special (on-site) servers which aggregate the data, remove redundant entries and prepare it for storage in a Cloud-backed database by indexing and encrypting it. Users can then search this database through trapdoor queries, meaning that the search process can be performed on the encrypted data. In order to obtain the searched data, users can download the encrypted results, and use their private keys to decrypt them. As a result, the Cloud environment will never have any access to the unencrypted data. Xu et al. [159] propose a similar solution, also relying on trapdoors to perform search queries on encrypted data sets in the Cloud. The difference is that in this solution, the used encryption techniques aim to be lightweight enough to allow for decryption by the IIoT devices (specifically sensor nodes) themselves, without requiring an intermediate server. This approach only targets data storage, search, and retrieval. Other use cases for Cloud environments, such as big data analysis in the Cloud itself, cannot be solved using this method. Miao et al. [155] also propose a Cloud-assisted method in the context of an e-health scenario, while attempting to minimize intensive tasks such as decryption and decryption at the Edge side, to computation requirements and power consumption

With the advent of blockchain technology, there has been an increase in interest in data sharing solutions based on decentralized ledgers. Sani et al. [157] propose a privacy preserving blockchain using mutually authenticated encryption for confidential data exchange, while others propose things such as energy trading [150] and big data markets [152]. Huang et al. [151] list three main challenges in blockchain technology: the trade-off between efficiency and security, coexistence of transparency and privacy, and conflicts between concurrency and throughput. These concerns are shared by Nikander et al. [156], who discuss throughput, latency, and resource requirements more in-depth. Further, they identify four models of operation for lightweight devices to participate in blockchains. Another proposed solution is to integrate devices with multiple ledgers, although the authors state that this is an active field of research. The aforementioned concerns are also identified in [148], where the authors further state that while blockchain promises enhanced data security and availability, for the IIoT domain there remain challenges regarding data privacy, integrity, and identify certification. They also list interoperability, standardization, and regulatory aspects as more general blockchain challenges. Other blockchain-based proposals in this domain are [153] and [154]. For a more thorough discussion on security requirements and challenges for blockchain in the IIoT, we refer the interested reader to [35], and for a discussion on risky characteristics common to blockchain technologies we point to [158].

*3) Data flow-control:* Through data flow control, data access policies can be enforced on a higher-level than encryption techniques, which provides a way to address security- and privacy requirements relating to the processing of data as it moves in a system.

Al-Ali et al. [162] describe a real-world use case for data flow monitoring, where certain data on machine error rates is shared within the company itself, and across organization boundaries based on a set of privacy policies. Some of these policies cannot be statically enforced because they depend on dynamically changing processes or coordinated interaction between different entities. They conclude that the ability to capture dynamic situations is a challenge that has yet to be overcome.

Identifying data security as a design requirement, Bloom et al. [163] investigated input-output patterns in existing IIoT applications in order to gain a better understanding of ways to secure information related to IIoT operations. Based on their observations, they propose some design patterns that can help protect data flow already in the design stage. Schütte and Brost [164] state that data flow enforcement is a requirement in certain contexts, and propose a policy-controlled data flow control framework capable of monitoring messages between entities both statically and at run-time. This allows users to not just specify access policies, but also to state how data elements are allowed to be processed by the system. Whether dynamic monitoring with this solution is possible in time-critical systems, is still a subject for further study.

*4) Data privacy:* Data privacy and ownership is an important topic for many companies and governments, and with the recent popularity of Cloud storage services, these issues require careful consideration [98]. With the amount of data that is generated by modern devices, it becomes possible to create detailed profiles of users, putting their privacy at risk [168]. In an attempt to mitigate this, an anonymous data collection framework is proposed in [169].

With recent legislation in the European Union (the General Data Protection Regulation (GDPR) [166]) effectively requiring privacy-by-design for all products, data privacy should be taken seriously by manufacturers as well. Preuveneers et al. [50] discuss the implications of the GDPR in Industry 4.0 and smart factory environments. For example, some requirements derived from this legislation are that (in general) customers of a service have the right to retrieve their personal information, the right to be forgotten, and the right to erasure of their personal information. This should be taken into account when designing systems that interact with humans and might collect such information. Acknowledging this need for integration, Conzon et al. [165] describe a model-based framework for IoT, the security and privacy principles of which are derived from the GDPR.

Privacy does not only concern data collection and Cloud storage, but also requires the obfuscation or omission of metadata and other properties that can be leveraged by adversaries. For example, in WSN networks, sensor nodes are often spread over a geographically wide area, and an adversary might attempt to locate the source node of specific traffic based on message flow. To remedy this, source location privacy schemes should be deployed such as the one proposed in [167].

### G. Security Monitoring

Dynamic monitoring of behavior in a system is an effective way to detect and respond to malicious activity, and systems that provide these capabilities are commonly known as Intrusion Detection Systems (IDSs). In the IIoT domain, two commonly identified security requirements are the ability to

TABLE X
SECURITY MONITORING REQUIREMENTS, SOURCES THAT IDENTIFY THESE, AND THEIR INTEREST LEVEL RELATIVE TO THE CATEGORY. THE RELATIVE INTEREST LEVEL IS BASED ON THE PERCENTAGE OF WORKS ADDRESSING THE SPECIFIC SECURITY REQUIREMENT COMPARED TO THE TOTAL NUMBER OF PAPERS FOR THAT CATEGORY.

| ID | Security requirement | Related sources | Relative interest | % within category |
|---|---|---|---|---|
| SM-01 | infrastructure monitoring | [55], [115], [170], [171], [173], [176], [178], [181], [184], [185], [189]–[195], [197]–[199], [201]–[204], [206]–[208] | Very High | 64% |
| SM-02 | threat response | [55], [115], [175], [184], [187], [193], [198], [204]–[206] | Medium | 24% |
| SM-03 | handle heterogeneous sources | [193] | Low | 2% |
| SM-04 | security policy enforcement | [191], [199], [204] | Low | 7% |

monitor infrastructure, and respond to known and unknown threats when necessary [55], [193], [198], [206]. The reason these are deemed particularly important for the IIoT comes from the fact that older, less secure devices are likely to be connected to the network as well [208]. These devices cannot always be patched to protect against known vulnerabilities, and therefore require continuous monitoring. An example of this is the IDS proposed by Kim and Kang [189], which specifically targets the Modbus protocol, a widely used industrial control protocol, and a good example of an existing protocol severely lacking in security mechanisms. Similarly, the MQTT protocol has been covered like this [178]. A second reason can be found in providing protection against Denial of Service (DoS) attacks [115] and improving congestion control in general [187].

Hasan and Mouftah [184] state that latency is one of the major challenges for security monitoring systems, due to the geographical distance between devices in certain Industry 4.0 networks, network latency can become too high for acceptable response times to intrusions, especially when using Cloud security services.

Another identified challenge for security monitoring in the IIoT is the imbalance of data sets. Due to the sheer amount of data generated by IIoT devices and the low attack frequency, obtained data sets that can be used for machine learning approaches to intrusion detection tend to be very imbalanced [206].

Many proposed IDS solutions exist that are designed to work in the general IT domain. However, it becomes harder to monitor threats when taking into account the extreme environments in which some IIoT appliances are deployed, resource constraints, and data privacy requirements. On the other hand, as IIoT system activity is largely the result of automated processes, the traffic patterns tend to be fairly static and periodic, making it easier to perform accurate anomaly detection [180], [208]. Additionally, this predictability introduces the possibility for utilizing these patterns against the system through stealthy injection attacks [196], or to establish covert communication channels, as demonstrated in [172], and should be monitored against. In [176], Bernieri et al. show that this predictability can be used against attackers by developing a honeypot for a water distribution system. It simulates physical processes, and is able to detect attacks that aim to modify the system's behavior. A machine learning based IDS capable of detecting these types of attacks proposed in [181]. However, Genge et al [183] note that when monitoring the output of physical processes, care has to be taken to take the gradual decay of processes (e.g. the wear on equipment) into account. They show that this can be done through statistical analysis. As the authors observed, there is very little work done in this area, and more research is needed to develop sophisticated measures that incorporate for process aging.

Settanni et al. [198] propose a self-adapting IDS that detects anomalies in the range of certain control values. Their solution requires the continuous collection of logs of all connected devices to a central control system, which is acceptable in environments with reasonably powerful machines, but not in WSNs or other sparse environments with lightweight nodes. The anomaly detection algorithm for physical quantities proposed by Zugasti et al. [208] similarly looks at observed quantities. However, in this work, no attention is given to the resource overhead of this approach, nor where it should be deployed in an IIoT system.

Very recently, there has been a surge in interest in machine learning techniques for anomaly detection in IIoT. For example, [200] and [203] provide a performance comparison of various machine learning algorithms for detecting anomalies in IIoT, in [171], Al-Hawawreh et al. propose a deep neural network approach for use in brownfield installations, in [170] the authors propose a similar system for ransomware detection, and in [197], the authors employ machine learning to detect time synchronization attacks. In [173], Alem et al. acknowledge the power of machine learning, but warn against high potentially false-positive rates. To mitigate this, they propose a hybrid system, that derives a semantic model from the ISA95 standard. Then, using a neural network for anomaly detection, they can filter out false positives and categorize anomalies based as being malicious or just dysfunction. Deep learning IDSs do not come without risk. In [186], the authors show that one can reliably create adversarial samples that defeat deep learning based systems. The findings in [188] agree with this, as also there the authors manage to bypass machine learning systems. Additionally, they show two methods of increasing resilience through retraining of the networks. Robustness against adversarial samples is something that needs to be taken into account when using machine learning for security monitoring. For a more thorough overview of the state machine learning for industrial IDSs, we refer the interested reader to [207].

Moustafa et al. [193] identify the requirement for IIoT monitoring services to handle a large amount of heterogeneous data sources. Their proposed solution uses Markov models and a central processing system (with parts running both in the

Cloud and Fog). The data collection itself happens through middleware, thereby minimizing the overhead on resource-constrained devices.

Threat response is a requirement identified by many works in this area, e.g. [55], [198], [205]. While this is usually in the form of notifying security personnel and mitigating the threat by stopping the service, Babiceanu et al. [175] use the flexibility provided by Software Defined Networkings (SDNs) to let the network operate in multiple modes, increasingly trading quality of service for security.

Whereas some approaches focus on intrusion detection in one layer in the Edge-Cloud spectrum, Yan et al. [115] propose a monitoring framework that contains systems operating in the Edge, Fog, and Cloud layers. This way, resource overhead for extremely lightweight Edge devices is kept to a minimum, while at the same time allowing localized management and response through the Fog layer. The Cloud layer uses data analysis approaches to intelligently detect attacks. This is similar to the DDoS mitigation approach proposed by Zhou et al. [205], where local virtual network functions, Fog, and Cloud work together to respond to DDoS attacks.

Another aspect of security monitoring concerns the continuous monitoring of network traffic ensuring that network security policies are not violated. This type of monitoring is to help maintain the integrity required of Industry 4.0 network infrastructure, and as such does not target devices themselves, but rather SDN controllers and routing devices. Melis et al. [191] propose a live monitoring solution of flow permission controls, as well as a proactive formal verification mechanism of the security policies in SDN systems.

That security monitoring can also be proactive, can be seen by looking at the fuzzing frameworks proposed by Flores et al. [182] and Niedermaier et al. [45]. The authors of the latter propose a fuzzing framework, that continuously tries to "attack" networked services with randomized data streams. It is lightweight, and is able to identify vulnerabilities due to common software bugs such as buffer overflows. However, with an approach such as this, care has to be taken that system performance is not affected, and that critical services remain available. As such, fuzzers might mainly be a tool for security researchers, and developers aiming to create a highly secure product. But when deployed carefully, production systems can also utilize them to detect configuration errors and vulnerabilities.

That IIoT environments can benefit from specialized monitoring approaches can also be seen when looking at drone scenarios. In their behavior and vulnerability assessment, Sharma et al. [199] identify a number of security requirements that are specific to this scenario, as well as several requirements that are more generally applicable. Specifically, they identify the need for: identification mechanisms; continuous monitoring; predictive and highly accurate vulnerability assessments; and the ability for anomalous drones to be marked by the monitoring service, so that this information can be shared with all drones in a swarm. Their solution utilizes Petri Nets to monitor behavior. Some other proposed monitoring solutions aimed at drone scenarios are based on behavior rule specifications [192] and recursive parameter estimation [177]. Another example of specialized security monitoring is provided by Deshpande et al. [179] propose a heartbeat protocol catering specifically to WSNs, ensuring that overhead on the sensor level is minimal.

### *H. Network Security*

Achieving adequate network security consists of many things, including authentication, secure transport, reliable and secure routing, and more. In previous sections we already discussed some of these, and will therefore focus on network infrastructure security.

With industrial networks becoming increasingly complex due to a large number of connected devices, we are faced with problems similar to those that occurred during the rapid expansion of the World Wide Web [187]. Because of this, many performance and scalability issues need to be addressed, such as bandwidth and latency contention. According to [212], many configuration, traffic control, and security systems rely on proprietary software which make integration in generic management frameworks impossible. At the same time, they state that network infrastructure is required to be flexible, to handle dynamic environments. To solve this challenge, two paradigms aimed at separating configuration and control from data transfer itself have been gaining traction: SDN and Network Function Virtualization (NFV). SDN concerns configuration and management, while NFV concerns virtual environments to run network and security functions on a layer that is abstracted the devices on which it runs. The authors propose an architecture using these paradigms to enforce security policies on switches with SDN and NFV capabilities, and move away from e.g. firewalls. They essentially attempt to address four security requirements through this approach: the ability to specify and enforce network security policies, to minimize management and configuration overhead, to allow for dynamic reconfiguration of the network and its security policies, and to minimize the overhead caused by enforcement of security policies. Other points where SDNs can improve system security are discussed in [217].

*1) Latency and timeliness:* Marchetto et al. [221] state that additionally connectivity and isolation between endpoints are network security requirements, although these can possibly be interpreted as security policies by themselves. While they identify these security requirements, their work addresses a slightly different matter: the Virtual Network Embedding problem, which concerns the placement of virtual network functions so that they are optimized and can be verified to correctly enforce the desired security policies. This can potentially be utilized by other works to keep overhead to a minimum and minimize network latency. Hu [218] also identifies latency as a challenge and states that the controllability and configurability of network architectures and applications are key elements in reducing latency. The implied requirement is thus that IIoT environments must be controllable and configurable at every level. This network latency issue is also relevant to security monitoring (previously discussed in Section V-G), as keeping latency to a minimum is a large issue in network monitoring services, and possible solutions include alteration of the network architecture [184].

TABLE XI
NETWORK SECURITY REQUIREMENTS, SOURCES THAT IDENTIFY THESE, AND THEIR INTEREST LEVEL RELATIVE TO THE CATEGORY. THE RELATIVE INTEREST LEVEL IS BASED ON THE PERCENTAGE OF WORKS ADDRESSING THE SPECIFIC SECURITY REQUIREMENT COMPARED TO THE TOTAL NUMBER OF PAPERS FOR THAT CATEGORY

| ID | Security requirement | Related sources | Relative interest | % within category |
|---|---|---|---|---|
| NS-01 | dynamicity of configuration | [212], [218], [219] | Medium | 10% |
| NS-02 | security policy enforcement | [221] | Low | 3% |
| NS-03 | management overhead minimization | [212], [224] | Low | 7% |
| NS-04 | network isolation | [129], [210], [211], [217], [220], [221], [224] | Medium | 24% |
| NS-05 | timeliness | [187], [213], [218]–[221] | Medium | 21% |
| NS-06 | availability (DoS, jamming, etc.) | [187], [214], [219], [221], [222], [226] | Medium | 21% |
| NS-07 | wireless transmission security | [32], [33], [209], [215], [216], [223]–[227] | High | 34% |

For time-critical applications, there exist specialized standards such as the Time Sensitive Networking (TSN) standards to provide deterministic and timely networking capabilities between systems. These applications often require remote access to sensors, actuators, and Programmable Logic Controller (PLC)s driving industrial devices. These connections must fulfill the same requirements as when those devices would be directly connected on the machine level [219]. For this, safety and security measures must be present in the network architecture to correctly prioritize such traffic.

With a gradual movement towards an IIoT enabled industrial process, it is expected that many legacy devices will remain operational for some time in parallel with new technologies, in a sunset phase. These legacy devices must thus be isolated from the internet, but care must be taken in the isolation technologies, as to not provide too much overhead in time-critical processes. To that end, Lackorzynski et al. [220] compare multiple readily available VPN solutions on metrics important to industrial appliances.

*2) Availability:* Latency is not the only issue. From a dependability perspective, single points of failure should be eliminated. However, with modern Cloud infrastructures, often the network virtualization solutions proposed by Cloud providers constrain customers to that one Cloud service provider [222]. To allow critical applications to utilize the Cloud for enhanced functionality, without sacrificing availability, the authors propose a platform to allow virtualized networks spanning multiple Cloud providers as well as private networks, while also solving the Virtual Network Embedding problem. This way, they are able to explore the flexibility of combining on-premises systems with Cloud systems, and satisfy privacy requirements by creating security policies limiting the mapping of sensitive NFV applications to specific classes of networks.

*3) Wireless:* Many smart devices make use of wireless technologies for data transmission. These wireless communication standards work on a lower level than the data transport technologies discussed in Section V-F1. However, the security requirements for wireless transmission that we found in the investigated literature largely overlap with those of data transport security. A common type of wireless communication technologies aimed at long-range low-power IoT devices are LPWAN technologies [229].

Chen et al. [223] list a number of security requirements in a review of the Narrow Band IoT (NB-IoT) standard. This standard was developed by the The 3rd Generation Partnership Project (3GPP) [252] and focuses on extremely low-power devices and indoor connections. The authors identify DoS as a much more apparent threat than in traditional networks, as low-power mobile devices will be easily drained from battery power. Another requirement is to prevent eavesdropping of transmissions, as information leakage can lead to devastating results. The authors also identify the need for devices to sign and encrypt their transmissions, in order to mitigate the potential impact of a compromised base station (they identify this as more likely than with traditional wireless technologies). Mutual authentication between devices and the base station is also mentioned, in order to prevent spoofing attacks. Recently, an exploratory investigation has shown that properties derived from the relative distance and direction between transmitters can help in identifying these types of attacks [32]. As the NB-IoT standard supports a large number of devices (100,000) being connected to one terminal, it is challenging to create sufficiently lightweight and efficient authentication and access control mechanisms for these.

Kail et al. [33] compare the security properties of several LPWAN technologies in the unlicensed bands. This comparison is done through the inspection of a number of capabilities that are to be expected of a secure standard, and therefore we consider them as sensible security requirements for wireless technologies: authentication, message integrity, confidentiality, Over-the-Air firmware upgrade capabilities, reliable communication, and key exchange capabilities. Note that these requirements are also already covered in other sections, so we do not list them in Table V-H. Additionally, they identify the need for protection against common attacks against wireless technologies, such as wide-band jamming, selective jamming, eavesdropping, traffic analysis, replay attacks, and wormhole attacks. Their conclusion is that further research on security and privacy-related features for low-power wireless communication standards is needed. Wang et al. [226] argue that in order to satisfy the confidentiality requirement, encryption techniques are not sufficient, and propose a friendly jamming scheme, making it harder for eavesdroppers to distinguish communication from noise.

6TiSCH [228] is a standardization effort by the Internet Engineering Task Force (IETF), aimed at low-power deterministic IPv6 communication for WSN technologies and industrial IIoT networks, by building on the IEEE 802.15.4 standard for low-rate Wireless Personal Area Network (WPAN)s, thus supporting a different category of devices than LPWAN technolo-

gies. Although timeliness is one of the major goals of 6TiSCH, it also aims to incorporate a variety of security properties. For example, the authors state that support for Datagram Transport Layer Security (DTLS) and TLS is taken into consideration. A further discussion of 6TiSCH security is given in [227]. Related to WPAN technology, Ulz et al. [225] propose a secure communication framework utilizing NFC, aimed at providing a reliable solution for mobile robots that need to communicate with machines. Due to the short-range nature, this naturally helps remedy eavesdropping and interference issues

The 5G standard also addresses IoT scenarios, and provides support for virtualization of network resources. This enables the creation of isolated network partitions with different demand profiles. Two key scenarios that 5G targets are massively deployed low-bandwidth IoT devices, and critical latency-sensitive applications. Both of these map very well to common IIoT and Industry 4.0 scenarios. Kurtz et al. [224] elaborate on network slicing, and how it can be realized through use of SDN and NFV technologies. The security requirements identified in their work concern strict isolation of network traffic, and the ability to provide hard service guarantees, such as on latency, data rate, and reliability. Additionally, they mention the need for manageability in this environment, as misconfiguration of systems can have a negative impact on the capabilities of the overall network. Their results show that 5G technologies can be used for real-time, critical applications.

### *I. Models and methodologies*

In this subsection, we discuss proposed security models and methodologies in the investigated literature. As the security issues that these address are relatively high-level, the security requirements are relatively abstract and encompass multiple aspects of IIoT systems. Therefore, the security requirements listed in table V-I are to be interpreted as recommendations and tools to improve the degree to which other security requirements can be satisfied, as well as easing the process of doing so.

Shaabany [51] states that software and hardware should be designed carefully, with security in mind, in order to reduce the attack surface as much as possible during design time. Among some less-security related requirements, they argue that specialized functions should be standardized for reuse as much as possible (across manufacturers as well), that all components should be uniquely identifiable and that this identifier should be used in communication with other components, and that security guarantees should be given on every hierarchical level. To aid in addressing these needs, the authors propose a security-by-design approach encompassing both hardware and software. It is thus clear that security should be considered at every step of the development lifecycle of a system, and in [234], Eckhart et al. propose 14 security activities spread across multiple phases in the development process that have shown to be effective for cyber-physical systems. Maksuti et al. [42] take a more flexible stance than Shaabany, observing that security and business process performance will always come at the cost of each other. They state that one possible solution is to create a self-adapting system that can flexibly provide end-to-end security. To this end, they propose the investigation of self-adapting models and describe a relevant meta-model. As an example, they suggest that TLS sessions can be re-used for intermittent communication in situations where the threat is deemed to be low, but the rate at which they should be renewed can be dynamically scaled up and down to accommodate for differences in threat levels. Another security-by-design approach recommends the usage of security control assignment matrices to determine the types of security controls that should be present in various parts of a system [132].

It is often easier and more effective to create more specific architectural frameworks rather than generic ones, and the investigated literature contains specialized models and methods for various scenarios. The security-by-design approach in [165] specializes in actuating and sensing scenarios, while in [231], the authors introduce an integrated model aimed specifically at mobile e-health applications. Their approach also considers security issues at design time and can be integrated into more generic architectures. Craggs et al. [233] target research scenarios, and describe a reference architecture for research testbeds, making the accurate observation that real IIoT scenarios are likely to have a mixture of legacy and new technologies and that security solutions should account for this. In [237], a method for arriving at a security capability-model for IIoT supply-chains is described, as the authors identified that businesses generally lack insight in their own supply chains, which is a security liability. In a comprehensive work, McGinthy and Michaels [242] describe secure architectural frameworks for IIoT and WSN sensor nodes, with security features grouped by energy class. They address many security requirements that should be satisfied for these classes, including data confidentiality, attestable boot procedures, and key management. Becue et al. [23] state that it is necessary to improve the prevention, detection, investigation, and response to adversarial machine learning attempts on AI-powered modules. At the same time, humans and machines should aid in the surveillance of each other; if a human behaves anomalously, machines should be able to detect and report this, and vice versa. They propose using a "cyber-range" approach where digital twins of physical devices are modeled by a team of engineers using feedback from the operators, as well as common design techniques such as risk assessments. These digital twins are then used to simulate more optimized usage scenarios, and red/blue teams perform attack and response scenarios, that help the digital twin learn about how to protect and respond to attacks by itself. Once a digital twin is deemed sufficiently secure it can be used in production settings. This approach requires decisions that steer towards such a model early on in the architectural design process. This is also necessary for the model described by Condry and Nielson [26]. In this model, the authors leverage capabilities of gateways between control systems and the internet to allow for direct communication between control systems and client devices. Kondeva et al [240] observe that the fields of safety and security engineering are closely related but have their own techniques and methods. They consider that safety and security requirements should not clash with each other and that these should be integrated more tightly. To this

TABLE XII
MODELS AND METHODOLOGIES SECURITY REQUIREMENTS, SOURCES THAT IDENTIFY THESE, AND THEIR INTEREST LEVEL RELATIVE TO THE CATEGORY. THE RELATIVE INTEREST LEVEL IS BASED ON THE PERCENTAGE OF WORKS ADDRESSING THE SPECIFIC SECURITY REQUIREMENT COMPARED TO THE TOTAL NUMBER OF PAPERS FOR THAT CATEGORY.

| ID | Security requirement | Related sources | Relative interest | % within category |
|---|---|---|---|---|
| MM-01 | adequate risk/threat assessment | [43], [44], [83], [100], [126], [232], [235], [236], [241], [244] | High | 33% |
| MM-02 | minimization of overall attack surface | [51] | Low | 3% |
| MM-03 | security by design | [23], [26], [41], [42], [51], [132], [230], [231], [234], [240], [242], [243], [245] | High | 43% |

end, they introduce a method to generate attack trees from fault tree analysis.

Risk assessment for the IIoT is another field that has seen activity in recent years. In [126] and [44] two risk assessment models for the IIoT are presented. The first is mainly focused on water sewage systems, but has aspects that can be generalized, while the second aims to be general, and utilizes use cases as its input. The authors of [44] state that it is not possible to protect against threats without a proper risk assessment. The reason that traditional risk assessment methods are not adequate due to the complexity of integrating all the aspects of an IIoT system, and due to the increased impact factor in IoT environments because of the increased amount of physical assets and ways it can affect human lives. To this end, they propose a 10-phase comprehensive risk assessment method, that is able to capture many relevant aspects. Mouratidis and Diamantopoulu [43] take things even further by proposing a more formal security analysis method for the IIoT. In their method they build on the Secure Tropos language to allow for precise modeling of industrial environments, their security constraints, and relevant threats. They then use graph analysis to trace possible attack paths and identify which devices should satisfy certain security requirements. A more manual approach is taken by Boyes et al. [232]. They propose a multidimensional categorization framework, that can help with a better analysis of threats, aside from being useful as a more general categorization framework. They envision that a proper categorization of devices will help with identifying similar threats across different aspects of the IIoT domain.

As resource constraints are often a bottleneck for IIoT systems, it is perhaps surprising that there has not been a lot of work on modeling the overhead these bring. The only such work that was found in the literature is by Ivkic et al. [238], and describes an onion layer model that enables one to sum all overhead introduced by security functions.

### *J. Summary and Discussion*

In our survey of the literature on security in the IIoT domain, we have extracted 49 security requirements covered by the investigated works, spread across 8 categories: Authentication, Access Control, Maintainability, Resilience, Data security and data sharing, Security Monitoring, Network Security, and Models and Methodologies. Additionally, we have made an effort to summarize the literature in our discussion.

In this subsection, we summarize the findings discussed in this section in two ways. Firstly, in Table XIII, we lay out

TABLE XIII
DISTRIBUTION OF THE INVESTIGATED PAPERS ACROSS THE CATEGORIES DISCUSSED IN THIS WORK.

| ID | Category | Papers (N°) | % |
|---|---|---|---|
| A | Authentication | 77 | 27% |
| AC | Access Control | 16 | 6% |
| M | Maintainability | 15 | 5% |
| R | Resilience | 16 | 6% |
| DSS | Data Security and Data Sharing | 58 | 20% |
| SM | Security Monitoring | 42 | 15% |
| NS | Network Security | 29 | 10% |
| MM | Models and Methodologies | 30 | 11% |

the number of works per category, providing a measure of the distribution of the papers across categories. As detailed in Section V, the number of papers addressing each category is taken from Figure 2 as the number of papers appearing in the corresponding level 1 (i.e., subsection) and all level 2 (i.e., subsubsections) nodes, but removing duplicates. Secondly, we summarize all the identified research requirements in Table XIV, listed in reverse order by their popularity based on the total number of investigated works. Note that the overall interest for this table is computed based on the total number of works covered in this survey, and is thus different from earlier tables in this section where it was computed based on the numbers within each category. Table XV lists these new thresholds.

A few observations can be made when looking at the popularity of the categories, which are laid out in Table XIII.

Firstly, research interest in Authentication, together with Data Security and Data Sharing appears significantly higher than the other categories. This is interesting because these intuitively also have the most in common with standard IoT scenarios. At the same time, the very IIoT-centered categories of Maintainability and Resilience are some of the least active. We believe that this exposes a promising area for new research.

Access Control has seemingly been of the least interest, perhaps because many of its security requirements and works are already implicitly treated in the Authentication section, and various works present frameworks that provide both, but are discussed in the Authentication category.

Security Monitoring is also fairly popular, with 41 works discussing it in various ways. What stands out about this category is that considering its popularity, there are relatively few (4) different requirements covered in the literature. This stands out even more when looking at Table XIV, where requirement SM-01 is the most popular of all. Further, both SM-01 and DSS-05 have seen significantly more interest

TABLE XIV
POPULARITY OF THE INDIVIDUAL REQUIREMENTS, TAKEN AS A PERCENTAGE OF THE TOTAL NUMBER OF UNIQUE WORKS COVERED IN THIS SURVEY.

| Overall interest | ID | Security Requirement | Category | Overall % |
|---|---|---|---|---|
| Very High | SM-01 | infrastructure monitoring | Security Monitoring | 9.5% |
| | DSS-05 | secure external data storage | Data Security and Data Sharing | 7.1% |
| | A-06 | mutual authentication | Authentication | 4.6% |
| | MM-03 | security by design | Models and Methodologies | 4.6% |
| High | DSS-02 | data confidentiality | Data Security and Data Sharing | 3.9% |
| | A-02 | key distribution | Authentication | 3.5% |
| | SM-02 | threat response | Security Monitoring | 3.5% |
| | NS-07 | wireless transmission security | Network Security | 3.5% |
| | MM-01 | adequate risk/threat assessment | Models and Methodologies | 3.5% |
| | A-08 | minimization of user interaction | Authentication | 2.8% |
| | AC-04 | decentralized AC | Access Control | 2.8% |
| Medium | A-01 | multi-factor authentication | Authentication | 2.5% |
| | NS-04 | network isolation | Network Security | 2.5% |
| | A-07 | privacy-preserving authentication | Authentication | 2.1% |
| | NS-05 | timeliness | Network Security | 2.1% |
| | NS-06 | availability (DoS, jamming, etc.) | Network Security | 2.1% |
| | A-03 | node addition, revocation, rekeying | Authentication | 1.8% |
| | A-04 | decentralized key management | Authentication | 1.8% |
| | AC-02 | fine-grained AC | Access Control | 1.8% |
| | R-01 | continuation of operation with compromised subsystems | Resilience | 1.8% |
| | R-03 | standards compliance | Resilience | 1.8% |
| | A-10 | attestation | Authentication | 1.4% |
| | AC-01 | handle dynamic changes | Access Control | 1.4% |
| | M-01 | software updateability | Maintainability | 1.4% |
| | M-08 | secure status transfer | Maintainability | 1.4% |
| | DSS-04 | secure data transport | Data Security and Data Sharing | 1.4% |
| | A-09 | non-repudation | Authentication | 1.1% |
| | AC-06 | transparency | Access Control | 1.1% |
| | M-02 | configuration updateability | Maintainability | 1.1% |
| | M-03 | disturbance-free updates | Maintainability | 1.1% |
| | DSS-06 | data flow control | Data Security and Data Sharing | 1.1% |
| | DSS-07 | data protection legislation compliance | Data Security and Data Sharing | 1.1% |
| | SM-04 | security policy enforcement | Security Monitoring | 1.1% |
| | NS-01 | dynamicity of configuration | Network Security | 1.1% |
| Low | AC-03 | centralized AC | Access Control | 0.7% |
| | AC-05 | privacy-preserving AC | Access Control | 0.7% |
| | M-05 | traceability | Maintainability | 0.7% |
| | M-06 | compatibility | Maintainability | 0.7% |
| | R-02 | operation with intermittent connectivity | Resilience | 0.7% |
| | NS-03 | management overhead minimization | Network Security | 0.7% |
| | A-05 | transitive authentication | Authentication | 0.3% |
| | AC-07 | compatibility | Access Control | 0.3% |
| | M-04 | usability of update process | Maintainability | 0.3% |
| | M-07 | transparency | Maintainability | 0.3% |
| | DSS-01 | data loss mitigation | Data Security and Data Sharing | 0.3% |
| | DSS-03 | standardization | Data Security and Data Sharing | 0.3% |
| | SM-03 | handle heterogeneous sources | Security Monitoring | 0.3% |
| | NS-02 | security policy enforcement | Network Security | 0.3% |
| | MM-02 | minimization of overall attack surface | Models and Methodologies | 0.3% |

than any other requirement. This is perhaps because these requirements are the most open-ended out of all identified requirements, thereby collecting a large variety of works that discuss them.

Finally, the observant eye might notice that in Table XIV the percentages sum up to 92.6%. This is because, throughout the study, roughly 7.4% of the investigated works identify some categories as requirements, meaning they have been included in this work, but do not identify any of the specific security requirements. Therefore, they are included in the category count, but not in the requirement count.

TABLE XV
INTEREST LEVELS ASSIGNED TO SECURITY REQUIREMENTS AND WEIGHTED ON THE COVERAGE OF EACH CATEGORY, APPLICABLE TO TABLE XIV

| Weighted interest | Range (x) |
|---|---|
| Low | 0% $\leq$ x $\leq$1.0% |
| Medium | 1.0% < x $\leq$2.5% |
| High | 2.5% < x $\leq$4.0% |
| Very High | 4.0% < x $\leq$100% |

## VI. QUANTITATIVE RESULTS

In this section, we provide a quantitative analysis of the set of studies resulting from the presented research.

In particular, we address research questions (RQ2)-(RQ4) by analyzing the number of publications related to IIoT security over the years, the geographical distribution of these studies, and the favorite publication venues.

### A. Spread of publications throughout the years (RQ2)

Figure 4 shows the number of publications between 2011 and 2019. Security research for the IIoT starts first appearing around 2011, being initially dormant but slowly growing from 2013 onward. In 2017, a drastic increase in activity can be seen. While it is tempting to attribute this growth to the fact that 2016 saw several serious IoT and industry related security incidents (such as Mirai [4] and Crashoverride/Industroyer [6]), which served to illustrate the importance of security on these devices, it should be noted that this is in line with the overall growth of IoT as a research area. In 2018 and 2019, the growth in activity continued, showing that the research community deems IIoT security to be of high importance.

### B. Geographical Distribution of IIoT Security Research (RQ3)

The geographical distribution of research activity is shown in Figure 5. The data for this was obtained by extracting the country of affiliation of the first author of the considered studies.

German-speaking countries are strongly represented, making for a total of 22% of contributions. One possible explanation is that one of our search terms, *Industry 4.0*, was originally coined by the German government [253], thus, it might have seen higher adoption in German-speaking countries. This raises the question of whether our search terms were successful in providing a good global sample of studies in this field. We believe they were, since the field we are considering is very narrow; we specifically searched for *Industrial* challenges in order to be able to extract security requirements unique to this field. Furthermore, we have conducted reverse snowball sampling to ensure a fair research scope.

China and the United States of America are the two other major contributors. This can be attributed to the size of their industries and thus the relevance of research in this area. However, interestingly, 54% of the studies originate from Europe, showing that this topic is also regarded as highly relevant in countries with smaller industries.

The 'others' group consists of the 23 countries that have 3 or fewer publications in this field: Algeria, Belgium, Brazil, Czech Republic, Finland, Greece, Hungary, Iran, Ireland, Japan, Malaysia, Morocco, New Zealand, Norway, Pakistan, Qatar, Romania, Russia, Saudi Arabia, Serbia, Taiwan, Turkey, Ukraine.

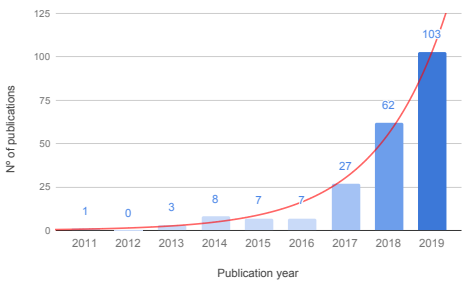



Fig. 4. Number of publications per year.

### C. Venue Types for Publication (RQ4)

We have grouped the studies based on the venue type of their publication, which is shown in Figure 6. As can be seen, conference proceedings are the most popular dissemination method, followed by journals. The 'others' category consists of venue types in which 4 or fewer publications were published: congresses, summits, and forums.

Looking at the specific venues of publication (Figure 7), it can be seen that the IEEE Transactions on Industrial Informatics journal is by far the most popular venue, with 25 publications. One noteworthy observation here is that, out of all considered studies, only 16 were published in venues focused on security. The vast majority of IIoT security-related works appears to be published in venues targeting industrial systems or IoT instead.

## VII. Opportunities Enabled by Fog Computing

In Section V, we have extracted security requirements for the IIoT from the investigated literature and discussed a number of challenges that stand in the way of the adoption of conventional solutions to address these requirements. In this section, we reflect on the challenges and discuss how Fog computing shows promise as a remedy to a number of those.

It is important to note that Fog computing is a relatively new paradigm the exact definition of which is still being debated in the scientific community and often intersects with similar paradigms, such as Edge computing, Mobile Edge computing, and Mobile Cloud computing. To maintain consistency with earlier work, we use the definition of Fog computing as used in [254]; a paradigm that extends the Cloud and integrates Edge and IoT, while providing a new, horizontally scalable highly virtualized layer that distributes computing, storage, control, and networking capabilities across the Cloud-To-Things spectrum [8]. For a more detailed treatise on the differences between Fog, Edge, and other paradigms we refer the interested reader to [254].

Also, we are aware that a comprehensive and thorough discussion on how Fog computing could tackle the IIoT security requirements would require a dedicated treatment that would result in an entire paper itself, which is out of the scope of this work (for instance, in [255] we focus on how Cloud requirements can impact IoT). Thus, the aim of this section is to provide food for thought on the topic and a source of inspiration for future research, rather than an exhaustive analysis.

In detail, we first give the definition of Fog computing assumed in this work. Then, we revisit the majority of topics covered in Section V and depicted in Figure 8: authentication, access control, maintainability, resilience, data security and data sharing, security monitoring, and network security. For each of these, we discuss how we envision what Fog-enabled

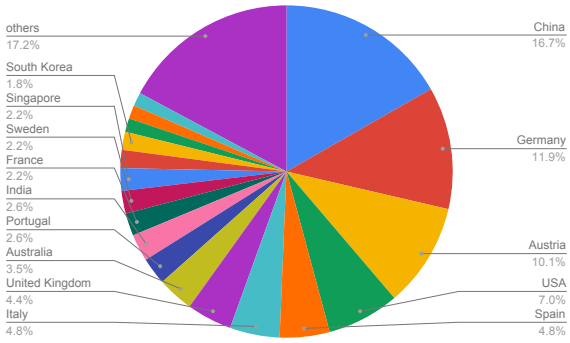



Fig. 5. Demographic: geographical distribution of research activity based on first author's country of affiliation.

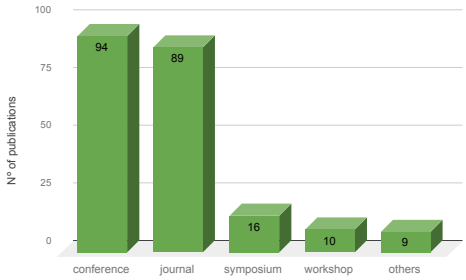



Fig. 6. Popularity of different venue types.

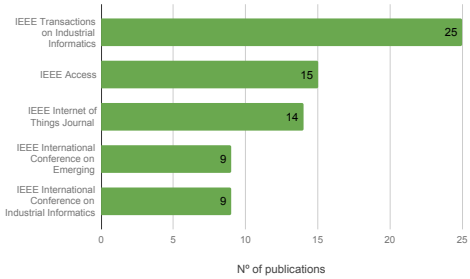



Fig. 7. Popularity of specific venues for publications.

solutions might look like and suggest potential research opportunities, but we leave confirmation of the validity of these ideas as a topic for further research. We close the section with a discussion on limitations and open challenges for Fog computing.

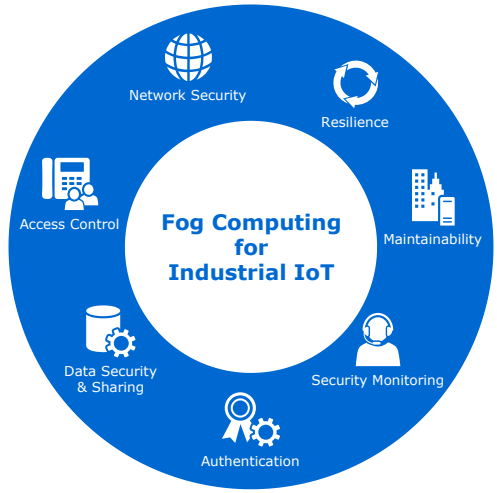



Fig. 8. Fog computing opportunities for IIoT security.

### A. Fog Computing

Fog computing is a relatively recent computing paradigm born from the necessity to provide the missing link in the Cloud-to-Thing continuum [8].

According to the IEEE standard 1934-2018 [256], Fog computing is "a horizontal, system-level architecture that distributes computing, storage, control, and networking functions closer to the users along a cloud-to-thing continuum". Thus, Fog computing can be considered as an extension of Cloud computing that distributes the benefits of the Cloud closer to the IIoT and across multiple layers of the network topology.

Any system that wants to be compliant with the aforementioned definition of Fog computing needs to present the fol-

lowing attributes, also referred to as pillars: security, scalability, openness, autonomy, reliability, availability, serviceability, agility, hierarchy, and programmability. A thorough discussion of these pillars can be found in [8], [256].

In this setting, the fog node is "the physical and logical network element that implements fog computing services" [8]. Since Fog nodes can be placed on-premises, they can be accessed by devices even when the connection to the outside world is failing. This helps us in identifying research opportunities for issues arising from intermittent connectivity. Note that this can be generalized: if there is a connection failure anywhere on the route from the (local) Fog node to the (remote) Cloud, then all Fog nodes that are positioned *before* the unreachable hop are still reachable and thus able to provide the local system with their services.

### B. Fog-enabled Authentication

When looking at the authentication challenges discussed in Section V-B, it can be observed that there are a number of points where a Fog node can be helpful in addressing them.

A first intuitive way of applying Fog computing to these challenges can be found by considering existing authentication solutions that require third-party servers in their setup or execution, such as [63], [86], [93]. A Fog node fits the requirements for these servers perfectly, as it is not severely restrained by computational or energy resources, is on-premises, and has very low response times. If Fog computing nodes are considered as part of the infrastructure, many of the issues with relying on a third-party server are thus addressed "for free".

Secondly, Fog nodes can serve to enhance traditional PKI infrastructures, where Fog nodes can act as "certificate authorities" for local devices or help establish a federated and robust key infrastructure through e.g. peer-to-peer networking capabilities with other Fog nodes. To our knowledge, no work investigating this currently exists.

In dynamic environments, Fog nodes can potentially help alleviate issues relating to node addition, removal, and rekeying as well. For example, it could serve as a trusted "gateway" to which Edge devices are paired, preventing them from communicating directly with any other system. This is not unlike how Bluetooth devices can be paired with smartphones and other devices. Node addition, removal, and rekeying can then be handled from the Fog node.

As we have seen in Section V-B2, some proposed solutions require biometric features ( [72], [77]), smart cards ( [72]) or NFC tags ( [36], [37], [67], [225]), in the authentication process. Also here there is potential for Fog nodes: not every lightweight system might be equipped with the necessary sensors for this. However, it might be possible to equip Fog nodes with sensors and use them as proxies for sensor readings. This would increase scalability, as a Fog node can be positioned so that it is more easily accessible than the Edge devices connected to it. Thus, if maintenance engineers would want to e.g. authenticate updates for the devices by using NFC keys or biometrics, they will only need to seek out the Fog node and present the relevant keys to it, as opposed to seeking out every relevant device separately.

Fog nodes might also enable the possibility of bringing TPM and/or TEE capabilities to Edge devices that do not contain these modules themselves. For that to be possible, the Edge devices need to set up a trusted channel between the Fog node's TPM/TEE module, which could be possible through some form of a key setup protocol that involves a one-time pairing step. Fog nodes could be equipped with multiple TPM or TEE modules to serve more than one Edge device (or itself) at the same time, such as the recently introduced Intel SGX cards [257]. Trusted hardware capabilities in Fog nodes can also be used for attestation purposes in various settings (against remote Fog nodes, against Edge devices, and so on). We expect that there are a lot of fruitful research directions for the combination of Fog nodes and trusted hardware.

### C. Fog-enabled Access Control

As with authentication, Fog nodes have the potential for enhancing AC challenges in industrial scenarios.

Firstly, some AC policies could be outsourced from extremely resource-constrained devices to a Fog node (e.g. accessing sensitive files from a central repository), or if the scenario is suited for it, AC can be managed completely by a Fog node. Another identified challenge for AC frameworks is that while managing policies centrally gives more flexibility, it introduces new risks due to the central server now being a single point of failure. Fog nodes could provide a "hybrid" middle ground where AC is federated between various Fog nodes on-premises, and that Edge devices can then query these Fog nodes, thus increasing the overall reliability and scalability. To the best of our knowledge, this is still an open research area.

As mentioned in Section V-C, compatibility with legacy devices is another issue in the IIoT. Fog nodes could act as a bridge between newer devices and legacy devices with poor security, keeping them sufficiently isolated from the wider network and providing security measures where necessary in exposed interfaces, possibly through a ZTN approach.

### D. Fog-enabled Maintainability

Fog computing can bring large benefits to the maintainability of industrial systems.

By their very nature, industrial systems are connected to the Internet, and thus enable the possibility of managing software and configuration updates for attached Edge devices. Fog nodes are perfectly situated to verify the validity of such updates and perform in-depth tests such as performing the updates in a sandboxed environment and then observing for anomalies before deploying them on real devices, while at the same time allowing for the application of updates with minimal disturbance to the services themselves. In practice, this would turn the solution proposed by [111] into a Fog application.

Fog nodes also provide an ideal target platform for an "industrial app marketplace" such as proposed in [113]. It is not difficult to envision a system where a Fog node would allow users to view software packages together with their version number and update information for all connected devices,

in an ordered and user-friendly way. Moreover, Fog nodes could go further and allow for management of configuration files for connected Edge devices as well. For example, one could think of an application where configuration files are retrieved from a Cloud service, verified by the Fog node and subsequently delivered to specified Edge devices, filling in sensitive information fields as necessary so as to prevent the Cloud from requiring access to this information.

As Fog nodes could provide an easily accessible location for the reading of NFC tags or other hardware authentication modules, one could easily extend maintenance processes with those extra authentication factors without requiring engineers to physically attend to each affected device individually.

The ideas described here are merely speculative, and there is plenty of room for research in any of these areas. We expect a variety of maintainability-enhancing applications of Fog computing will be identified and researched in the future.

### *E. Fog-enabled Resilience*

Fog nodes could act as reactive security agents, isolating or disabling connected devices when they appear compromised. This allows security personnel to then further investigate the issue, while the system itself can continue operations. This is also discussed in [258], where a number of Fog use-cases and research challenges are listed. The authors state that automatic fault detection and reconfiguration is essential, and identify the potential for Fog nodes to do this autonomously, but state that this is a challenging topic that requires addressing. However, a solution to this challenge would enable resiliency as it is defined by the ICS.

A second challenge that can be overcome through Fog computing, is maintaining normal operation through intermittent internet connectivity. To an extent, a Fog node can take over processes normally executed in the Cloud. Thus, when the connection to the Cloud fails, the operational capability of Edge devices is not affected. Related to this, some devices continuously or periodically need to transmit data to the cloud, where it can then be processed. If this were done directly, data loss is a risk in case of intermittent connectivity. As an alternative to introducing some data storage capabilities on the Edge devices themselves, a Fog node could collect data from the devices, and forward it to the Cloud. Then, when there is no connection to the internet, the Fog node can act as a buffer and send the buffered information upwards to the Cloud once the connection is restored. This way, Edge devices do not need to worry about failing internet connectivity at all.

Finally, Fog nodes and their application-independent software can be developed to satisfy resiliency-related indicator points, which in turn can aid in providing contractual service guarantees as is currently often seen in Cloud service agreements.

### *F. Fog-enabled data security and data sharing*

Whenever it is necessary for a device to access sensitive data that should be stored securely, this requires the device to firstly have the storage capacity, and secondly the means to secure this data at rest. For lightweight systems that do not have the capacity to store and secure data securely, Fog nodes can provide a solution; they are not tied to severe resource constraints and can be equipped with ample storage and computational capacity for common encryption methods. Additionally, Fog nodes can be deployed on the local network, meaning data will never have to leave the premises. Even for extremely large amounts of data, Fog nodes could act as middleware between external Cloud storage, and encrypt/decrypt data stored in the Cloud transparently, e.g. using the techniques described in [149], [159]. To the Edge devices, it can be presented as originating from the Fog node, and they do not need to be aware of the underlying storage and security mechanisms.

As Fog nodes can be positioned between Edge devices and external parties as gateways, this also unlocks the opportunity to secure and control data flow to these external parties. A Fog node can set up and maintain highly secure, authenticated channels with remote parties, potentially alleviating some of the challenges involved in designing lightweight Edge devices that need to interact with these parties, as they only need to concern themselves with secure communication with the Fog node. If the Fog node additionally has the ability to access the message content of traffic passing through it, it can enforce data flow policies, e.g. as described in [164], allowing fine-grained data security mechanisms on top of encryption techniques.

In Section V-F4 we stated that the protection of sensitive data is in many cases now a legal requirement in the European Union. Fog nodes present a very natural way of meeting these requirements, as they can store data locally, while at the same time allowing for fine-grained data sharing with third parties, should a user allow this. Moreover, it can become easier to manage user rights such as the right to be forgotten.

### *G. Fog-enabled Security Monitoring*

Because Fog nodes can take on central positions in Industrial networks, they provide a great platform for security monitoring solutions.

For example, a Fog node could run IDS software to detect anomalies or attack signatures. This also provides an opportunity for the Fog and Cloud to augment each other. Intrusion detection models could be trained in a Cloud environment, while executed on a Fog node, thereby addressing the latency issues normally apparent in Cloud solutions. Examples of this can be found in [115], [193]. Because Fog nodes stand in direct connection to sensor devices, they can also perform simple anomaly detection techniques such as ensuring that sensor values are within a certain value range, without adding overhead to the sensors themselves.

Another use of Fog nodes as a security monitoring tool could be the deployment of an anti-malware for IoT devices that is supported by the Fog infrastructure [259]. Indeed, De Donno et al. [260], [261] propose an anti-malware software for IoT and they discuss how the use of Fog computing helps to solve some of the challenges intrinsic in the deployment.

Fog nodes can also potentially take action based on incoming traffic patterns, enabling the mitigation of DoS attacks aimed at very specific devices, even when those devices are

not able to protect themselves against those attacks. This also presents the opportunity for dynamic traffic shaping, and other techniques that might help reduce battery consumption on lightweight IoT devices connected to the Fog node.

### H. Fog-enabled Network Security

Also in network infrastructure, Fog computing can potentially help in overcoming current challenges.

With the rise of SDN and NFV technologies, Fog nodes can possibly play a role as a platform for some of these. For example, they can create isolated network environments between themselves and each connected device.

Fog nodes could also be equipped to handle TSN standards when there is a need for deterministic and timely delivery of network traffic between two connected devices. By moving the management of these interfaces to a Fog node, opportunities are created for easier (remote) management and reconfiguration of time-critical systems, even going so far as to move entire control applications to Fog nodes. As an extreme manifestation of this vision, one could imagine "plug-and-play" industrial hardware that can be connected to a Fog node which will then autonomously configure and use it.

We also see opportunities for Fog nodes to improve the availability of critical services, in two ways. Firstly, Fog nodes could run critical applications in a federated fashion, allowing migration or load balancing of tasks between them. This way, the application only becomes unavailable when all participating Fog nodes fail. Secondly, a Fog node can act as a middleware for a critical service running in the Cloud. By deploying this service on multiple Cloud providers, Edge applications relying on it will not be affected by the outage of any one cloud provider; the Fog node can automatically route requests to the remaining available providers.

Finally, Fog nodes could potentially aid in securing wireless infrastructure, by incorporating wireless technologies in security monitoring solutions. This way, jamming attacks or other anomalies in the wireless spectrum can be detected.

### I. Challenges and Limitations

Fog computing is not a panacea capable of filling any Cloud-IIoT gap without much issue. The paradigm is very much in its early stages, and deployment so far has been extremely limited. Open challenges include practical federation frameworks, resource offloading, and resilience [262]. While we believe that solutions to these challenges are capable of satisfying the security requirements collected in this work, we acknowledge that every solution comes with its own trade-offs, and a thorough analysis of the benefits and drawbacks of Fog computing can only be done once enough Fog-based systems exist to investigate. Nevertheless, one can attempt to make an analysis based on the current state-of-the-art. Thus, in this section, we briefly discuss what we consider some of the biggest potential drawbacks.

Firstly, Fog systems add extra workload to maintenance personnel, and will likely require special training, making it more costly than the Cloud. Whereas Cloud infrastructure is maintained by a specialized team on the Cloud service provider's end, the Fog paradigm shifts this responsibility to users of the system. The spread of functionality across the Cloud-to-Things continuum potentially complicates this even more. If a security issue is found in a well-known piece of Fog infrastructural software, it is the responsibility of maintainers *at every point in the continuum* to update their software, as opposed to having to update just the Cloud infrastructure, which is managed by one entity. If one maintainer of a Fog node fails to do this within an appropriate time-window, this can put all entities making use of that node at risk.

Secondly, incident response might be hampered by the distributed nature of Fog systems. We believe this might manifest itself in multiple ways: necessary security expertise might not be available on-site, and specialized incident response teams will have to be called in from external parties. Further, complex incidents might require cooperation between multiple entities along the continuum for forensic analysis, which might not always be possible or add a lot of overhead.

Finally, compatibility between Fog nodes can potentially be a huge issue. If standards are not well-defined or not followed rigorously, it will be very hard to meet the harsh requirements set by industrial environments with nodes from different providers that cannot interoperate efficiently and accurately. This, in turn, can negatively impact the ability to federate and offload tasks to other nodes in the local network, as well as potentially violate security policies if some nodes in the system are unable to uphold the necessary requirements.

## VIII. Conclusion

In this work, we have performed a systematic literature review about security for the IIoT.

As in any mapping study, it is challenging to take all studies of the field into account, but it is more important to have a good representation of studies rather than a high number of studies [20]. To achieve a good representation, we have methodologically constructed the search queries and queried multiple literature repositories. After that, we utilized reverse snowball sampling to further increase the quality, and to mitigate any possible selection bias. Our initial search queries resulted in 356 possibly relevant papers, which we brought down to a selection of 218 papers through the use of a systematic approach comprised of several phases. These papers were fully read and analyzed for the purposes of this study.

At glance, the work has elaborated around four main research questions: (RQ1) what security requirements exist for the IIoT, (RQ2) how scientific publications about IIoT security are spread during the years, (RQ3) how IIoT security research activity is geographically distributed, and (RQ4) what publication venues are the most popular for IIoT security.

First, we have answered question RQ1 by extracting security requirements for the IIoT from the investigated works and exploring them, along with the related challenges that make these requirements hard to meet with existing solutions and a measure of their interest in the research community. Then, we have addressed questions (RQ2)-(RQ4) by providing a quantitative analysis of the investigated IIoT security research.

Finally, we provided a discussion on how Fog computing can play a role in meeting the requirements posed by industrial environments, by taking a Fog computing perspective and revisiting the requirements that were extracted during our investigation, as well as pointing out what limitation and challenges still need to be faced to achieve massive Fog computing deployment.

This work identifies an abundance of research opportunities in the IIoT security area and shows that Fog computing, as a rising computing paradigm, can become a powerful tool in securing a variety of connected industrial environments, once its limitations and challenges are overcome.

## ACKNOWLEDGMENT

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 764785, FORA – Fog Computing for Robotics and Industrial Automation.

## REFERENCES

[1] P. Daugherty and B. Berthon, "Winning with the industrial internet of things: How to accelerate the journey to productivity and growth," Dublín: Accenture, Tech. Rep., 2015.

[2] N. Dragoni, A. Giaretta, and M. Mazzara, "The Internet of Hackable Things," in *Proceedings of 5th International Conference in Software Engineering for Defence Applications*, P. Ciancarini, S. Litvinov, A. Messina, A. Sillitti, and G. Succi, Eds. Springer, 2017, pp. 129–140.

[3] M. De Donno, N. Dragoni, A. Giaretta, and A. Spognardi, "Analysis of DDoS-Sapable IoT Malwares," in *Federated Conference on Computer Science and Information Systems (FedCSIS)*. IEEE, 2017, pp. 807–816.

[4] ——, "DDoS-Capable IoT Malwares: Comparative Analysis and Mirai Investigation," *Security and Communication Networks*, vol. 2018, 2018.

[5] R. Langner, "Stuxnet: Dissecting a cyberwarfare weapon," *IEEE Security & Privacy*, vol. 9, no. 3, pp. 49–51, 2011.

[6] R. Lee, "CRASHOVERRIDE: Analysis of the threat to electric grid operations," Dragos Inc., Tech. Rep., 2017.

[7] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog Computing and Its Role in the Internet of Things," in *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, ser. MCC '12. ACM, 2012, pp. 13–16.

[8] OpenFog Consortium Architecture Working Group and others, "OpenFog Reference architecture for Fog Computing," OpenFog Consortium, Tech. Rep., February 2017. [Online]. Available: https://www.iiconsortium.org/pdf/OpenFog_Reference_Architecture_2_09_17.pdf

[9] "Good Practices for Security of Internet of Things in the Context of Smart Manufacturing," ENISA, Tech. Rep. TP-04-18-940-EN-N, 2018.

[10] X. Yu and H. Guo, "A Survey on IIoT Security," in *2019 IEEE VTS Asia Pacific Wireless Communications Symposium (APWCS)*, 2019, pp. 1–5.

[11] F. Meneghello, M. Calore, D. Zucchetto, M. Polese, and A. Zanella, "IoT: Internet of Threats? A Survey of Practical Security Vulnerabilities in Real IoT Devices," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8182–8201, 2019.

[12] N. Neshenko, E. Bou-Harb, J. Crichigno, G. Kaddoum, and N. Ghani, "Demystifying IoT Security: An Exhaustive Survey on IoT Vulnerabilities and a First Empirical Look on Internet-Scale IoT Exploitations," *IEEE Communications Surveys Tutorials*, vol. 21, no. 3, pp. 2702–2733, 2019.

[13] D. E. Kouicem, A. Bouabdallah, and H. Lakhlef, "Internet of things security: A top-down survey," *Computer Networks*, vol. 141, pp. 199 – 221, 2018.

[14] M. Lezzi, M. Lazoi, and A. Corallo, ""cybersecurity for industry 4.0 in the current literature: A reference framework"," *Computers in Industry*, vol. 103, pp. 97 – 110, 2018.

[15] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, "Industrial internet of things: Challenges, opportunities, and directions," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 11, pp. 4724–4734, 2018.

[16] F. Hofer, "Architecture, technologies and challenges for cyber-physical systems in industry 4.0: A systematic mapping study," in *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, ser. ESEM '18. ACM, 2018, pp. 1:1–1:10.

[17] A. Sadeghi, C. Wachsmann, and M. Waidner, "Security and privacy challenges in industrial internet of things," in *Proceedings of the 52Nd Annual Design Automation Conference*, ser. DAC '15. ACM, 2015, pp. 54:1–54:6.

[18] A. Sajid, H. Abbas, and K. Saleem, "Cloud-assisted IoT-based scada systems security: A review of the state of the art and future challenges," *IEEE Access*, vol. 4, pp. 1375–1384, 2016.

[19] G. Hansch, P. Schneider, K. Fischer, and K. BÃűttinger, "A Unified Architecture for Industrial IoT Security Requirements in Open Platform Communications," in *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2019, pp. 325–332.

[20] K. Petersen, S. Vakkalanka, and L. Kuzniarz, "Guidelines for conducting systematic mapping studies in software engineering: An update," *Information and Software Technology*, vol. 64, pp. 1–18, 2015.

[21] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," EBSE Technical Report, Tech. Rep. EBSE-2007-01, 2007.

[22] J. Beel and B. Gipp, "Google scholar's ranking algorithm: An introductory overview," in *Proceedings of the 12th International Conference on Scientometrics and Informetrics (ISSI'09*. Springer, 2009, pp. 439–446.

[23] A. Bécue, Y. Fourastier, I. Praça, A. Savarit, C. Baron, B. Gradussofs, E. Pouille, and C. Thomas, "Cyberfactory#1 — securing the industry 4.0 with cyber-ranges and digital twins," in *2018 14th IEEE International Workshop on Factory Communication Systems (WFCS)*. IEEE, 2018, pp. 1–4.

[24] M. Beltrán, M. Calvo, and S. González, "Federated system-to-service authentication and authorization combining pufs and tokens," in *2017 12th International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC)*. IEEE, 2017, pp. 1–8.

[25] A. Bicaku, S. Maksuti, S. Palkovits-Rauter, M. Tauber, R. Matischek, C. Schmittner, G. Mantas, M. Thron, and J. Delsing, "Towards trustworthy end-to-end communication in industry 4.0," in *2017 IEEE 15th International Conference on Industrial Informatics (INDIN)*. IEEE, 2017, pp. 889–896.

[26] M. W. Condry and C. B. Nelson, "Using smart edge IoT devices for safer, rapid response with industry IoT control operations," *Proceedings of the IEEE*, vol. 104, no. 5, pp. 938–946, 2016.

[27] J. Delsing, "Local cloud internet of things automation: Technology and business model features of distributed internet of things automation solutions," *IEEE Industrial Electronics Magazine*, vol. 11, no. 4, pp. 8–21, 2017.

[28] A. Esfahani, G. Mantas, R. Matischek, F. B. Saghezchi, J. Rodriguez, A. Bicaku, S. Maksuti, M. G. Tauber, C. Schmittner, and J. Bastos, "A Lightweight Authentication Mechanism for M2M Communications in Industrial IoT Environment," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 288–296, 2019.

[29] C. Esposito, A. Castiglione, B. Martini, and K. R. Choo, "Cloud manufacturing: Security, privacy, and forensic concerns," *IEEE Cloud Computing*, vol. 3, no. 4, pp. 16–22, 2016.

[30] C. Esposito, A. Castiglione, F. Palmieri, and A. D. Santis, "Integrity for an event notification within the industrial internet of things by using group signatures," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 8, pp. 3669–3678, 2018.

[31] M. A. Ferrag, L. A. Maglaras, H. Janicke, J. Jiang, and L. Shu, ""authentication protocols for internet of things: A comprehensive survey"," *Security and Communication Networks*, vol. 2017, 2017.

[32] X. Jiang, Z. Pang, M. Luvisotto, F. Pan, R. Candell, and C. Fischione, "Using a Large Data Set to Improve Industrial Wireless Communications: Latency, Reliability, and Security," *IEEE Industrial Electronics Magazine*, vol. 13, no. 1, pp. 6–12, 2019.

[33] E. Kail, A. Banati, E. László, and M. Kozlovszky, "Security survey of dedicated IoT networks in the unlicensed ism bands," in *2018 IEEE 12th International Symposium on Applied Computational Intelligence and Informatics (SACI)*. IEEE, 2018, pp. 000 449–000 454.

[34] S. Katsikeas, K. Fysarakis, A. Miaoudakis, A. V. Bemten, I. Askoxylakis, I. Papaefstathiou, and A. Plemenos, "Lightweight amp; secure

industrial IoT communications via the mq telemetry transport protocol," in *2017 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 2017, pp. 1193–1200.

[35] T. Kumar, A. Braeken, V. Ramani, I. Ahmad, E. Harjula, and M. Yliant-tila, "SEC-BlockEdge: Security Threats in Blockchain-Edge based Industrial IoT Networks," in *2019 11th International Workshop on Resilient Networks Design and Modeling (RNDM)*, 2019, pp. 1–7.

[36] C. Lesjak, T. Ruprechter, H. Bock, J. Haid, and E. Brenner, "Estado — enabling smart services for industrial equipment through a secured, transparent and ad-hoc data transmission online," in *The 9th International Conference for Internet Technology and Secured Transactions (ICITST-2014)*. IEEE, 2014, pp. 171–177.

[37] C. Lesjak, T. Ruprechter, J. Haid, H. Bock, and E. Brenner, "A secure hardware module and system concept for local and remote industrial embedded system identification," in *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*. IEEE, 2014, pp. 1–7.

[38] C. Lesjak, D. Hein, M. Hofmann, M. Maritsch, A. Aldrian, P. Priller, T. Ebner, T. Ruprechter, and G. Pregartner, "Securing smart maintenance services: Hardware-security and tls for mqtt," in *2015 IEEE 13th International Conference on Industrial Informatics (INDIN)*. IEEE, 2015, pp. 1243–1250.

[39] C. Lesjak, D. Hein, and J. Winter, "Hardware-security technologies for industrial IoT: Trustzone and security controller," in *IECON 2015 - 41st Annual Conference of the IEEE Industrial Electronics Society*. IEEE, 2015, pp. 002 589–002 595.

[40] C. Lesjak, H. Bock, D. Hein, and M. Maritsch, "Hardware-secured and transparent multi-stakeholder data exchange for industrial IoT," in *2016 IEEE 14th International Conference on Industrial Informatics (INDIN)*. IEEE, 2016, pp. 706–713.

[41] Z. Ma, A. Hudic, A. Shaaban, and S. Plosz, "Security viewpoint in a reference architecture model for cyber-physical production systems," in *2017 IEEE European Symposium on Security and Privacy Workshops (EuroS PW)*. IEEE, 2017, pp. 153–159.

[42] S. Maksuti, A. Bicaku, M. Tauber, S. Palkovits-Rauter, S. Haas, and J. Delsing, "Towards flexible and secure end-to-end communication in industry 4.0," in *2017 IEEE 15th International Conference on Industrial Informatics (INDIN)*. IEEE, 2017, pp. 883–888.

[43] H. Mouratidis and V. Diamantopoulou, "A security analysis method for industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 9, pp. 4093–4100, 2018.

[44] E. T. Nakamura and S. L. Ribeiro, "A privacy, security, safety, resilience and reliability focused risk assessment methodology for IIoT systems steps to build and use secure IIoT systems," in *2018 Global Internet of Things Summit (GIoTS)*. IEEE, 2018, pp. 1–6.

[45] M. Niedermaier, F. Fischer, and A. von Bodisco, "Propfuzz — an it-security fuzzing framework for proprietary ics protocols," in *2017 International Conference on Applied Electronics (AE)*. IEEE, 2017, pp. 1–4.

[46] Y. Nozaki and M. Yoshikawa, "Countermeasure of Lightweight Physical Unclonable Function Against Side-Channel Attack," in *2019 Cybersecurity and Cyberforensics Conference (CCC)*, 2019, pp. 30–34.

[47] OWASP, "2018 OWASP IoT Top 10," OWASP, Tech. Rep., December 2018. [Online]. Available: https://www.owasp.org/images/1/1c/OWASP-{IoT}-Top-10-2018-final.pdf

[48] M. S. Pardeshi and S. Yuan, "SMAP Fog/Edge: A Secure Mutual Authentication Protocol for Fog/Edge," *IEEE Access*, vol. 7, pp. 101 327–101 335, 2019.

[49] T. Pereira, L. Barreto, and A. Amaral, "Network and information security challenges within industry 4.0 paradigm," *Procedia Manufacturing*, vol. 13, pp. 1253–1260, 2017, manufacturing Engineering Society International Conference 2017, MESIC 2017, 28-30 June 2017, Vigo (Pontevedra), Spain.

[50] D. Preuveneers, W. Joosen, and E. Ilie-Zudor, "Data protection compliance regulations and implications for smart factories of the future," in *2016 12th International Conference on Intelligent Environments (IE)*. IEEE, 2016, pp. 40–47.

[51] G. Shaabany and R. Anderl, "Security by design as an approach to design a secure industry 4.0-capable machine enabling online-trading of technology data," in *2018 International Conference on System Science and Engineering (ICSSE)*. IEEE, 2018, pp. 1–5.

[52] T. Ulz, T. Pieber, C. Steger, S. Haas, and R. Matischek, "Secured remote configuration approach for industrial cyber-physical systems," in *2018 IEEE Industrial Cyber-Physical Systems (ICPS)*. IEEE, 2018, pp. 812–817.

[53] K. Wallis, F. Kemmer, E. Jastremskoj, and C. Reich, "Adaption of a privilege management infrastructure (pmi) approach to industry 4.0," in *2017 5th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*. IEEE, 2017, pp. 101–107.

[54] Z. Yang, J. He, Y. Tian, and J. Zhou, "Faster Authenticated Key Agreement with Perfect Forward Secrecy for Industrial Internet-of-Things," *IEEE Transactions on Industrial Informatics*, pp. 1–1, 2019.

[55] L. Zhou, K. Yeh, G. Hancke, Z. Liu, and C. Su, "Security and privacy for the industrial internet of things: An overview of approaches to safeguarding endpoints," *IEEE Signal Processing Magazine*, vol. 35, no. 5, pp. 76–87, 2018.

[56] O. Bergmann, S. Gerdes, and C. Bormann, "Simple keys for simple smart objects," in *Workshop on Smart Object Security*, 2012.

[57] U. Hunkeler, H. L. Truong, and A. Stanford-Clark, "MQTT-S – A publish/subscribe protocol for Wireless Sensor Networks," in *2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE'08)*. IEEE, 2008, pp. 791–798.

[58] S. Raza, D. Trabalza, and T. Voigt, "6LoWPAN compressed DTLS for CoAP," in *2012 IEEE 8th International Conference on Distributed Computing in Sensor Systems*. IEEE, 2012, pp. 287–289.

[59] D. Airehrour, J. Gutierrez, and S. K. Ray, "Securing rpl routing protocol from blackhole attacks using a trust-based mechanism," in *2016 26th International Telecommunication Networks and Applications Conference (ITNAC)*. IEEE, 2016, pp. 115–120.

[60] A. AlAbdullatif, K. AlAjaji, N. S. Al-Serhani, R. Zagrouba, and M. AlDossary, "Improving an Identity Authentication Management Protocol in IIoT," in *2019 2nd International Conference on Computer Applications Information Security (ICCAIS)*, 2019, pp. 1–6.

[61] P. C. Bartolomeu, E. Vieira, S. M. Hosseini, and J. Ferreira, "Self-Sovereign Identity: Use-cases, Technologies, and Challenges for Industrial IoT," in *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2019, pp. 1173–1180.

[62] S. Blanch-Torné, F. Cores, and R. M. Chiral, "Agent-based pki for distributed control system," in *2015 World Congress on Industrial Control Systems Security (WCICSS)*. IEEE, 2015, pp. 28–35.

[63] M. H. Eldefrawy, N. Pereira, and M. Gidlund, "Key distribution protocol for industrial internet of things without implicit certificates," *IEEE Internet of Things Journal*, 2018, (early access).

[64] K. Huang, X. Zhang, Y. Mu, X. Wang, G. Yang, X. Du, F. Rezaeibagha, Q. Xia, and M. Guizani, "Building Redactable Consortium Blockchain for Industrial Internet-of-Things," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3670–3679, 2019.

[65] D. W. McKee, S. J. Clement, J. Almutairi, and J. Xu, "Survey of advances and challenges in intelligent autonomy for distributed cyber-physical systems," *CAAI Transactions on Intelligence Technology*, vol. 3, no. 2, pp. 75–82, 2018.

[66] ——, "Massive-Scale Automation in Cyber-Physical Systems: Vision & Challenges," in *2017 IEEE 13th International Symposium on Autonomous Decentralized System (ISADS)*. IEEE, 2017, pp. 5–11.

[67] T. Ulz, T. Pieber, C. Steger, S. Haas, H. Bock, and R. Matischek, "Bring your own key for the industrial internet of things," in *2017 IEEE International Conference on Industrial Technology (ICIT)*. IEEE, 2017, pp. 1430–1435.

[68] A. Whitmore, A. Agarwal, and L. Da Xu, "The internet of thingsâĂŤa survey of topics and trends," *Information Systems Frontiers*, vol. 17, no. 2, pp. 261–274, 2015.

[69] L. Marchegiani and I. Posner, "Long-term driving behaviour modelling for driver identification," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 913–919.

[70] F. Al-Turjman and S. Alturjman, "Context-sensitive access in industrial internet of things (IIoT) healthcare applications," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 6, pp. 2736–2744, 2018.

[71] P. Autenrieth, C. Lörcher, C. Pfeiffer, T. Winkens, and L. Martin, "Current significance of it-infrastructure enabling industry 4.0 in large companies," in *2018 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC)*. IEEE, 2018, pp. 1–8.

[72] A. K. Das, M. Wazid, N. Kumar, A. V. Vasilakos, and J. J. P. C. Rodrigues, "Biometrics-based privacy-preserving user authentication scheme for cloud-based industrial internet of things deployment," *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 4900–4913, 2018.

[73] B. D. Deebak, F. Al-Turjman, M. Aloqaily, and O. Alfandi, "An Authentic-Based Privacy Preservation Protocol for Smart e-Healthcare Systems in IoT," *IEEE Access*, vol. 7, pp. 135 632–135 649, 2019.

[74] S. Garg, K. Kaur, G. Kaddoum, and K. R. Choo, "Towards Secure and Provable Authentication for Internet of Things: Realizing Industry 4.0," *IEEE Internet of Things Journal*, pp. 1–1, 2019.

[75] S. Hussain and S. A. Chaudhry, "Comments on "Biometrics-Based Privacy-Preserving User Authentication Scheme for Cloud-Based Industrial Internet of Things Deployment"," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 10 936–10 940, 2019.

[76] K. K. Kolluru, C. Paniagua, J. van Deventer, J. Eliasson, J. Delsing, and R. J. DeLong, "An AAA solution for securing industrial IoT devices using next generation access control," in *2018 IEEE Industrial Cyber-Physical Systems (ICPS)*. IEEE, 2018, pp. 737–742.

[77] X. Li, J. Niu, M. Z. A. Bhuiyan, F. Wu, M. Karuppiah, and S. Kumari, "A robust ecc-based provable secure authentication protocol with privacy preserving for industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 8, pp. 3599–3609, 2018.

[78] X. Li, J. Peng, J. Niu, F. Wu, J. Liao, and K. R. Choo, "A robust and energy efficient authentication protocol for industrial internet of things," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 1606–1615, 2018.

[79] M. Loske, L. Rothe, and D. G. Gertler, "Context-Aware Authentication: State-of-the-Art Evaluation and Adaption to the IIoT," in *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*, 2019, pp. 64–69.

[80] Z. Ma, Y. Yang, X. Liu, Y. Liu, S. Ma, K. Ren, and C. Yao, "EmIr-Auth: Eye-movement and Iris Based Portable Remote Authentication for Smart Grid," *IEEE Transactions on Industrial Informatics*, pp. 1–1, 2019.

[81] Q. Tian, Y. Lin, X. Guo, J. Wen, Y. Fang, J. Rodriguez, and S. Mumtaz, "New Security Mechanisms of High-Reliability IoT Communication Based on Radio Frequency Fingerprint," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 7980–7987, 2019.

[82] D. Wang and P. Wang, ""understanding security failures of two-factor authentication schemes for real-time applications in hierarchical wireless sensor networks"," *Ad Hoc Networks*, vol. 20, pp. 1–15, 2014.

[83] R. Ankele, S. Marksteiner, K. Nahrgang, and H. Vallant, "Requirements and Recommendations for IoT/IIoT Models to Automate Security Assurance through Threat Modelling, Security Analysis and Penetration Testing," in *Proceedings of the 14th International Conference on Availability, Reliability and Security*, ser. ARES âĂŹ19. New York, NY, USA: Association for Computing Machinery, 2019.

[84] F. Fraile, T. Tagawa, R. Poler, and A. Ortiz, "Trustworthy industrial IoT gateways for interoperability platforms and ecosystems," *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 4506–4514, 2018.

[85] A. Karati, S. H. Islam, and M. Karuppiah, "Provably secure and lightweight certificateless signature scheme for IIoT environments," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 8, pp. 3701–3711, 2018.

[86] F. Li, J. Hong, and A. A. Omala, "Efficient certificateless access control for industrial internet of things," *Future Generation Computer Systems*, vol. 76, pp. 285–292, 2017.

[87] C. Lin, D. He, X. Huang, K. R. Choo, and A. V. Vasilakos, "Bsein: A blockchain-based secure mutual authentication with fine-grained access control system for industry 4.0," *Journal of Network and Computer Applications*, vol. 116, pp. 42–52, 2018.

[88] F. Rezaeibagha, Y. Mu, X. Huang, W. Yang, and K. Huang, "Fully Secure Lightweight Certificateless Signature Scheme for IIoT," *IEEE Access*, vol. 7, pp. 144 433–144 443, 2019.

[89] V. Sklyar and V. Kharchenko, "Challenges in assurance case application for industrial IoT," in *2017 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, vol. 2. IEEE, 2017, pp. 736–739.

[90] T. Wu, C. Chen, K. Wang, and J. M. Wu, "Security Analysis and Enhancement of a Certificateless Searchable Public Key Encryption Scheme for IIoT Environments," *IEEE Access*, vol. 7, pp. 49 232–49 239, 2019.

[91] W. Yang, S. Wang, X. Huang, and Y. Mu, "On the Security of an Efficient and Robust Certificateless Signature Scheme for IIoT Environments," *IEEE Access*, vol. 7, pp. 91 074–91 079, 2019.

[92] Y. Zhang, R. H. Deng, D. Zheng, J. Li, P. Wu, and J. Cao, "Efficient and Robust Certificateless Signature for Data Crowdsensing in Cloud-Assisted Industrial IoT," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 9, pp. 5099–5108, 2019.

[93] H. Cui, R. H. Deng, J. K. Liu, X. Yi, and Y. Li, "Server-aided attribute-based signature with revocation for resource-constrained industrial-internet-of-things devices," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 8, pp. 3724–3732, 2018.

[94] S. Paliwal, "Hash-Based Conditional Privacy Preserving Authentication and Key Exchange Protocol Suitable for Industrial Internet of Things," *IEEE Access*, vol. 7, pp. 136 073–136 093, 2019.

[95] A. Hoeller and R. Toegl, "Trusted platform modules in cyber-physical systems: On the interference between security and dependability," in *2018 IEEE European Symposium on Security and Privacy Workshops (EuroS PW)*. IEEE, 2018, pp. 136–144.

[96] H. Laaki, Y. Miche, and K. Tammi, "Prototyping a Digital Twin for Real Time Remote Control Over Mobile Networks: Application of Remote Surgery," *IEEE Access*, vol. 7, pp. 20 325–20 336, 2019.

[97] E. Weippl and P. Kieseberg, "Security in cyber-physical production systems: A roadmap to improving it-security in the production system lifecycle," in *2017 AEIT International Annual Conference*. IEEE, 2017, pp. 1–6.

[98] Z. Bakhshi, A. Balador, and J. Mustafa, "Industrial IoT security threats and concerns by considering cisco and microsoft IoT reference models," in *2018 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*. IEEE, 2018, pp. 173–178.

[99] G. Chen and W. S. Ng, "An efficient authorization framework for securing industrial internet of things," in *TENCON 2017 - 2017 IEEE Region 10 Conference*. IEEE, 2017, pp. 1219–1224.

[100] G. Falco, C. Caldera, and H. Shrobe, "IIoT cybersecurity risk modeling for scada systems," *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 4486–4495, 2018.

[101] X. Feng, J. Wu, J. Li, and S. Wang, "Efficient secure access to ieee 21451 based wireless IIoT using optimized teds and mib," in *IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society*. IEEE, 2018, pp. 5221–5227.

[102] D. He, J. Bu, S. Zhu, S. Chan, and C. Chen, "Distributed Access Control with Privacy Support in Wireless Sensor Networks," *IEEE Transactions on Wireless Communications*, vol. 10, pp. 3472–3481, 2011.

[103] Y. Kim, Y. Lee, and J. Kim, "RIPPLE: Adaptive fine-grained access control in multi-hop LLNs," in *2018 International Conference on Information Networking (ICOIN)*. IEEE, 2018, pp. 863–868.

[104] A. Lahbib, K. Toumi, A. Laouiti, and S. Martin, "DRMF: A Distributed Resource Management Framework for Industry 4.0 Environments," in *2019 IEEE 18th International Symposium on Network Computing and Applications (NCA)*, 2019, pp. 1–9.

[105] M. Langfinger, M. Schneider, D. Stricker, and H. D. Schotten, "Addressing security challenges in industrial augmented reality systems," in *2017 IEEE 15th International Conference on Industrial Informatics (INDIN)*. IEEE, 2017, pp. 299–304.

[106] F. Martinelli, P. Mori, A. Saracino, and F. Di Cerbo, "Obligation Management in Usage Control Systems," in *2019 27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, 2019, pp. 356–364.

[107] D. Preuveneers, W. Joosen, and E. Ilie-Zudor, "Identity management for cyber-physical production workflows and individualized manufacturing in industry 4.0," in *Proceedings of the Symposium on Applied Computing*, ser. SAC '17. ACM, 2017, pp. 1452–1455.

[108] R. Vanickis, P. Jacob, S. Dehghanzadeh, and B. Lee, "Access control policy enforcement for zero-trust-networking," in *2018 29th Irish Signals and Systems Conference (ISSC)*. IEEE, 2018, pp. 1–6.

[109] X. Yao, H. Kong, H. Liu, T. Qiu, and H. Ning, "An Attribute Credential Based Public Key Scheme for Fog Computing in Digital Manufacturing," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 4, pp. 2297–2307, 2019.

[110] G. George and S. M. Thampi, "A graph-based security framework for securing industrial IoT networks from vulnerability exploitations," *IEEE Access*, vol. 6, pp. 43 586–43 601, 2018.

[111] I. Mugarza, J. Parra, and E. Jacob, "Cetratus: Towards a live patching supported runtime for mixed-criticality safe and secure systems," in *2018 IEEE 13th International Symposium on Industrial Embedded Systems (SIES)*. IEEE, 2018, pp. 1–8.

[112] I. Mugarza, A. Amurrio, E. Azketa, and E. Jacob, "Dynamic Software Updates to Enhance Security and Privacy in High Availability Energy Management Applications in Smart Cities," *IEEE Access*, vol. 7, pp. 42 269–42 279, 2019.

[113] A. Seitz, D. Henze, D. Miehle, B. Bruegge, J. Nickles, and M. Sauer, "Fog computing as enabler for blockchain-based IIoT app marketplaces - a case study," in *2018 Fifth International Conference on Internet of Things: Systems, Management and Security*. IEEE, 2018, pp. 182–188.

[114] G. Yadav and K. Paul, "PatchRank: Ordering updates for SCADA systems," in *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2019, pp. 110–117.

[115] Q. Yan, W. Huang, X. Luo, Q. Gong, and F. R. Yu, "A multi-level ddos mitigation framework for the industrial internet of things," *IEEE Communications Magazine*, vol. 56, no. 2, pp. 30–36, 2018.

[116] L. Zhou and H. Guo, "Anomaly detection methods for IIoT networks," in *2018 IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI)*. IEEE, 2018, pp. 214–219.

[117] P. Priller, A. Aldrian, and T. Ebner, "Case study: From legacy to connectivity migrating industrial devices into the world of smart services," in *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*, 2014, pp. 1–8.

[118] A. W. Atamli and A. Martin, "Threat-based security analysis for the internet of things," in *2014 International Workshop on Secure Internet of Things*, 2014, pp. 35–43.

[119] E. Bauer, O. Schluga, S. Maksuti, A. Bicaku, D. Hofbauer, I. Ivkic, M. G. Tauber, and A. Wöhrer, "Towards a security baseline for iaas-cloud back-ends in industry 4.0," in *2017 12th International Conference for Internet Technology and Secured Transactions (ICITST)*. IEEE, 2017, pp. 427–432.

[120] A. Bicaku, C. Schmittner, M. Tauber, and J. Delsing, "Monitoring industry 4.0 applications for security and safety standard compliance," in *2018 IEEE Industrial Cyber-Physical Systems (ICPS)*. IEEE, 2018, pp. 749–754.

[121] B. Dieber and B. Breiling, "Security Considerations in Modular Mobile Manipulation," in *2019 Third IEEE International Conference on Robotic Computing (IRC)*, 2019, pp. 70–77.

[122] M. Ehrlich, H. Trsek, L. Wisniewski, and J. Jasperneite, "Survey of Security Standards for an automated Industrie 4.0 compatible Manufacturing," in *IECON 2019 - 45th Annual Conference of the IEEE Industrial Electronics Society*, vol. 1, 2019, pp. 2849–2854.

[123] Industrial Internet Consortium, "Industrial Internet of Things Volume G4: Security Framework," Tech. Rep. IIC:PUB:g4:V1.0:PB:20160926, September 2016. [Online]. Available: https://www.iiconsortium.org/IISF.htm

[124] International Electrotechnical Commission, *IEC 62443 Security for Industrial Automation and Control Systems*, Std., 2009-2018.

[125] F. Januário, C. Carvalho, A. Cardoso, and P. Gil, "Security challenges in scada systems over wireless sensor and actuator networks," in *2016 8th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*. IEEE, 2016, pp. 363–368.

[126] A. Laszka, W. Abbas, Y. Vorobeychik, and X. Koutsoukos, "Synergistic security for the industrial internet of things: Integrating redundancy, diversity, and hardening," in *2018 IEEE International Conference on Industrial Internet (ICII)*. IEEE, 2018, pp. 153–158.

[127] B. Leander, A. Čaušević, and H. Hansson, "Applicability of the IEC 62443 Standard in Industry 4.0 / IIoT," in *Proceedings of the 14th International Conference on Availability, Reliability and Security*, ser. ARES âĂŹ19. New York, NY, USA: Association for Computing Machinery, 2019.

[128] V. Sklyar and V. Kharchenko, "ENISA Documents in Cybersecurity Assurance for Industry 4.0: IIoT Threats and Attacks Scenarios," in *2019 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, vol. 2, 2019, pp. 1046–1049.

[129] N. Benias and A. P. Markopoulos, "A review on the readiness level and cyber-security challenges in industry 4.0," in *2017 South Eastern European Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM)*. IEEE, 2017, pp. 1–5.

[130] S. R. Chhetri, N. Rashid, S. Faezi, and M. A. A. Faruque, "Security trends and advances in manufacturing systems in the era of industry 4.0," in *Proceedings of the 36th International Conference on Computer-Aided Design*, ser. ICCAD '17. IEEE Press, 2017, pp. 1039–1046.

[131] R. Chong and W. Lee, "Accelerating ElGamal Partial Homomorphic Encryption with GPU Platform for Industrial Internet of Things," in *2019 International Conference on Green and Human Information Technology (ICGHIT)*, 2019, pp. 108–112.

[132] A. Hassanzadeh, S. Modi, and S. Mulchandani, "Towards effective security control assignment in the industrial internet of things," in *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*. IEEE, 2015, pp. 795–800.

[133] N. Jazdi, "Cyber physical systems in the context of industry 4.0," in *2014 IEEE International Conference on Automation, Quality and Testing, Robotics*. IEEE, 2014, pp. 1–4.

[134] M. Kiss, G. Breda, and L. Muha, "Information security aspects of Industry 4.0," *Procedia Manufacturing*, vol. 32, pp. 848 – 855, 2019, 12th International Conference Interdisciplinarity in Engineering, INTER-ENG 2018, 4âĂŞ5 October 2018, Tirgu Mures, Romania.

[135] M. Ma, D. He, N. Kumar, K. R. Choo, and J. Chen, "Certificateless searchable public key encryption scheme for industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 2, pp. 759–767, 2018.

[136] J. Moyne, S. Mashiro, and D. Gross, "Determining a security roadmap for the microelectronics industry," in *2018 29th Annual SEMI Advanced Semiconductor Manufacturing Conference (ASMC)*. IEEE, 2018, pp. 291–294.

[137] K. Niemann, "IT security extensions for PROFINET," in *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, vol. 1, 2019, pp. 407–412.

[138] C. Yin, J. Xi, R. Sun, and J. Wang, "Location privacy protection based on differential privacy strategy for big data in industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 8, pp. 3628–3636, 2018.

[139] M. Zhang, B. Peng, and Y. Chen, "An Efficient Image Encryption Scheme for Industrial Internet-of-Things Devices," in *Proceedings of the 2nd International ACM Workshop on Security and Privacy for the Internet-of-Things*, ser. IoT S&P'19. New York, NY, USA: Association for Computing Machinery, 2019, pp. 38–43.

[140] Y. Zhang, H. Huang, L. Yang, Y. Xiang, and M. Li, "Serious Challenges and Potential Solutions for the Industrial Internet of Things with Edge Intelligence," *IEEE Network*, vol. 33, no. 5, pp. 41–45, 2019.

[141] Y. Zhao, L. T. Yang, and J. Sun, "Privacy-Preserving Tensor-Based Multiple Clusterings on Cloud for Industrial IoT," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 4, pp. 2372–2381, 2019.

[142] H. Klaus, F. Hetzelt, P. Hofmann, A. Blecker, and D. Schwaiger, "Challenges and Solutions for Industry-Grade Secure Connectivity," in *2019 International Conference on Networked Systems (NetSys)*, 2019, pp. 1–5.

[143] S. Marksteiner, "Reasoning on adopting opc ua for an IoT-enhanced smart energy system from a security perspective," in *2018 IEEE 20th Conference on Business Informatics (CBI)*, vol. 02. IEEE, 2018, pp. 140–143.

[144] V. Nigam and C. Talcott, "Formal Security Verification of Industry 4.0 Applications," in *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2019, pp. 1043–1050.

[145] OPC, "The OPC Unified Architecture," Tech. Rep., 2008. [Online]. Available: https://opcfoundation.org/about/opc-technologies/opc-ua/

[146] B. Chen, L. Wu, N. Kumar, K. R. Choo, and D. He, "Lightweight Searchable Public-key Encryption with Forward Privacy over IIoT Outsourced Data," *IEEE Transactions on Emerging Topics in Computing*, pp. 1–1, 2019.

[147] L. Deng, "Anonymous Aggregate Encryption Scheme for Industrial Internet of Things," *IEEE Systems Journal*, pp. 1–8, 2019.

[148] T. M. Fernandez-Carames and P. Fraga-Lamas, "A Review on the Application of Blockchain to the Next Generation of Cybersecure Industry 4.0 Smart Factories," *IEEE Access*, vol. 7, pp. 45 201–45 218, 2019.

[149] J. Fu, Y. Liu, H. Chao, B. K. Bhargava, and Z. Zhang, "Secure data storage and searching for industrial IoT by integrating fog computing and cloud computing," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4519–4528, 2018.

[150] Z. Guan, X. Lu, N. Wang, J. Wu, X. Du, and M. Guizani, "Towards secure and efficient energy trading in IIoT-enabled energy internet: A blockchain approach," *Future Generation Computer Systems*, 2019.

[151] J. Huang, L. Kong, G. Chen, M. Wu, X. Liu, and P. Zeng, "Towards Secure Industrial IoT: Blockchain System With Credit-Based Consensus Mechanism," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3680–3689, 2019.

[152] Y. Jiang, Y. Zhong, and X. Ge, "Smart Contract-Based Data Commodity Transactions for Industrial Internet of Things," *IEEE Access*, vol. 7, pp. 180 856–180 866, 2019.

[153] W. Liang, M. Tang, J. Long, X. Peng, J. Xu, and K. Li, "A Secure FaBric Blockchain-Based Data Transmission Technique for Industrial Internet-of-Things," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3582–3592, 2019.

[154] C. H. Liu, Q. Lin, and S. Wen, "Blockchain-Enabled Data Collection and Sharing for Industrial IoT With Deep Reinforcement Learning," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3516–3526, 2019.

[155] Y. Miao, Q. Tong, K. R. Choo, X. Liu, R. H. Deng, and H. Li, "Secure Online/Offline Data Sharing Framework for Cloud-Assisted Industrial Internet of Things," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8681–8691, 2019.

[156] P. Nikander, J. Autiosalo, and S. Paavolainen, "Interledger for the Industrial Internet of Things," in *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, vol. 1, 2019, pp. 908–915.

[157] A. S. Sani, D. Yuan, W. Bao, P. L. Yeoh, Z. Y. Dong, B. Vucetic, and E. Bertino, "Xyreum: A High-Performance and Scalable Blockchain for IIoT Security and Privacy," in *2019 IEEE 39th International*

*Conference on Distributed Computing Systems (ICDCS)*, 2019, pp. 1920–1930.

[158] N. Stifter, M. Eckhart, B. Brenner, and E. Weippl, "Avoiding Risky Designs When Using Blockchain Technologies in Cyber-Physical Systems," in *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2019, pp. 1623–1626.

[159] P. Xu, S. He, W. Wang, W. Susilo, and H. Jin, "Lightweight searchable public-key encryption for cloud-assisted wireless sensor networks," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 8, pp. 3712–3723, 2018.

[160] Y. Yu, R. Chen, H. Li, Y. Li, and A. Tian, "Toward Data Security in Edge Intelligent IIoT," *IEEE Network*, vol. 33, no. 5, pp. 20–26, 2019.

[161] X. Zhang, C. Xu, H. Wang, Y. Zhang, and S. Wang, "FS-PEKS: Lattice-based Forward Secure Public-key Encryption with Keyword Search for Cloud-assisted Industrial Internet of Things," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–1, 2019.

[162] R. Al-Ali, R. Heinrich, P. Hnetynka, A. Juan-Verdejo, S. Seifermann, and M. Walter, "Modeling of dynamic trust contracts for industry 4.0 systems," in *Proceedings of the 12th European Conference on Software Architecture: Companion Proceedings*, ser. ECSA '18. ACM, 2018, pp. 45:1–45:4.

[163] G. Bloom, B. Alsulami, E. Nwafor, and I. C. Bertolotti, "Design patterns for the industrial internet of things," in *2018 14th IEEE International Workshop on Factory Communication Systems (WFCS)*. IEEE, 2018, pp. 1–10.

[164] J. Schuette and G. S. Brost, "Lucon: Data flow control for message-based IoT systems," in *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*. IEEE, 2018, pp. 289–299.

[165] D. Conzon, M. R. A. Rashid, X. Tao, A. Soriano, R. Nicholson, and E. Ferrera, "BRAIN-IoT: Model-Based Framework for Dependable Sensing and Actuation in Intelligent Decentralized IoT Systems," in *2019 4th International Conference on Computing, Communications and Security (ICCCS)*, 2019, pp. 1–8.

[166] The European Parliament and the council of the European union, "REGULATION (EU) 2016/679 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL (GDPR)," European Union, Tech. Rep., April 2016. [Online]. Available: http://data.europa.eu/eli/reg/2016/679/oj

[167] G. Han, X. Miao, H. Wang, L. Liu, J. Jiang, and Y. Peng, "A Dynamic Multipath Scheme for Protecting Source-Location Privacy Using Multiple Sinks in WSNs Intended for IIoT," *IEEE Transactions on Industrial Informatics*, pp. 1–1, 2019.

[168] Z. A. Solangi, Y. A. Solangi, S. Chandio, M. bt. S. Abd. Aziz, M. S. bin Hamzah, and A. Shah, "The future of data privacy and security concerns in internet of things," in *2018 IEEE International Conference on Innovative Research and Development (ICIRD)*. IEEE, 2018, pp. 1–4.

[169] M. Usman, M. A. Jan, A. Jolfaei, M. Xu, X. He, and J. Chen, "DaaC: A Distributed and Anonymous Data Collection Framework based on Multi-Level Edge Computing Architecture," *IEEE Transactions on Industrial Informatics*, pp. 1–1, 2019.

[170] M. Al-Hawawreh and E. Sitnikova, "Industrial Internet of Things Based Ransomware Detection Using Stacked Variational Neural Network," in *Proceedings of the 3rd International Conference on Big Data and Internet of Things*, ser. BDIOT 2019. New York, NY, USA: Association for Computing Machinery, 2019, p. 126âĂŞ130.

[171] M. Al-Hawawreh, E. Sitnikova, and F. den Hartog, "An Efficient Intrusion Detection Model for Edge System in Brownfield Industrial Internet of Things," in *Proceedings of the 3rd International Conference on Big Data and Internet of Things*, ser. BDIOT 2019. New York, NY, USA: Association for Computing Machinery, 2019, p. 83âĂŞ87.

[172] C. Alcaraz, G. Bernieri, F. Pascucci, J. Lopez, and R. Setola, "Covert Channels-Based Stealth Attacks in Industry 4.0," *IEEE Systems Journal*, vol. 13, no. 4, pp. 3980–3988, 2019.

[173] S. Alem, D. Espes, E. Martin, L. Nana, and F. De Lamotte, "A Hybrid Intrusion Detection System in Industry 4.0 Based on ISA95 Standard," in *2019 IEEE/ACS 16th International Conference on Computer Systems and Applications (AICCSA)*, 2019, pp. 1–8.

[174] R. Antrobus, B. Green, S. Frey, and A. Rashid, "The forgotten I in IIoT: A vulnerability scanner for industrial Internet of Things," in *Living in the Internet of Things (IoT 2019)*, 2019, pp. 1–8.

[175] R. F. Babiceanu and R. Seker, "Cyber resilience protection for industrial internet of things: A software-defined networking approach," *Computers in Industry*, vol. 104, pp. 47 – 58, 2019.

[176] G. Bernieri, M. Conti, and F. Pascucci, "MimePot: a Model-based Honeypot for Industrial Control Networks," in *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, 2019, pp. 433–438.

[177] Z. Birnbaum, A. Dolgikh, V. Skormin, E. O'Brien, and D. Muller, "Unmanned aerial vehicle security using recursive parameter estimation," in *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2014, pp. 692–702.

[178] R. Colelli, S. Panzieri, and F. Pascucci, "Securing connection between IT and OT: the Fog Intrusion Detection System prospective," in *2019 II Workshop on Metrology for Industry 4.0 and IoT (MetroInd4.0 IoT)*, 2019, pp. 444–448.

[179] V. Deshpande, L. George, and H. Badis, "PulSec: Secure Element based framework for sensors anomaly detection in Industry 4.0," *IFAC-PapersOnLine*, vol. 52, no. 13, pp. 1204 – 1209, 2019, 9th IFAC Conference on Manufacturing Modelling, Management and Control MIM 2019.

[180] S. D. Duque Anton, M. Strufe, and H. D. Schotten, "Modern Problems Require Modern Solutions: Hybrid Concepts for Industrial Intrusion Detection," in *Mobile Communication - Technologies and Applications; 24. ITG-Symposium*, 2019, pp. 1–5.

[181] C. Enăchescu, H. Sándor, and B. Genge, "A Multi-Model-based Approach to Detect Cyber Stealth Attacks in Industrial Internet of Things," in *2019 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, 2019, pp. 1–6.

[182] J. L. Flores and I. Mugarza, "Runtime vulnerability discovery as a service on industrial internet of things (IIoT) systems," in *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1. IEEE, 2018, pp. 948–955.

[183] B. Genge, P. Haller, and C. Enachescu, "Anomaly Detection in Aging Industrial Internet of Things," *IEEE Access*, vol. 7, pp. 74 217–74 230, 2019.

[184] M. M. Hasan and H. T. Mouftah, "Cloud-centric collaborative security service placement for advanced metering infrastructures," *IEEE Transactions on Smart Grid*, vol. 10, no. 2, pp. 1339–1348, 2017.

[185] Y. Hu, D. Zhang, G. Cao, and Q. Pan, "Network Data Analysis and Anomaly Detection Using CNN Technique for Industrial Control Systems Security," in *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, 2019, pp. 593–597.

[186] O. Ibitoye, O. Shafiq, and A. Matrawy, "Analyzing Adversarial Attacks against Deep Learning for Intrusion Detection in IoT Networks," in *2019 IEEE Global Communications Conference (GLOBECOM)*, 2019, pp. 1–6.

[187] P. Kadera and P. Novák, "Performance modeling extension of directory facilitator for enhancing communication in fipa-compliant multiagent systems," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 2, pp. 688–695, 2017.

[188] M. E. Khoda, T. Imam, J. Kamruzzaman, I. Gondal, and A. Rahman, "Robust Malware Defense in Industrial IoT Applications using Machine Learning with Selective Adversarial Samples," *IEEE Transactions on Industry Applications*, pp. 1–1, 2019.

[189] B. Kim and Y. Kang, "Abnormal traffic detection mechanism for protecting IIoT environments," in *2018 International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE, 2018, pp. 943–945.

[190] V. Krundyshev and M. Kalinin, "Prevention of false data injections in smart infrastructures," in *2019 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*, 2019, pp. 1–5.

[191] A. Melis, D. Berardi, C. Contoli, F. Callegati, F. Esposito, and M. Prandini, "A policy checker approach for secure industrial sdn," in *2018 2nd Cyber Security in Networking Conference (CSNet)*. IEEE, 2018, pp. 1–7.

[192] R. Mitchell and I. Chen, "Adaptive intrusion detection of malicious unmanned air vehicles using behavior rule specifications," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 44, no. 5, pp. 593–604, 2014.

[193] N. Moustafa, E. Adi, B. Turnbull, and J. Hu, "A new threat intelligence scheme for safeguarding industry 4.0 systems," *IEEE Access*, vol. 6, pp. 32 910–32 924, 2018.

[194] D. M. Nedeljkovic, Z. B. Jakovljevic, Z. D. Miljkovic, and M. Pajic, "Detection of cyber-attacks in electro-pneumatic positioning system with distributed control," in *2019 27th Telecommunications Forum (TELFOR)*, 2019, pp. 1–4.

[195] M. Niedermaier, F. Fischer, D. Merli, and G. Sigl, "Network Scanning and Mapping for IIoT Edge Node Device Security," in *2019 International Conference on Applied Electronics (AE)*, 2019, pp. 1–6.

[196] S. Potluri, C. Diedrich, S. R. Roy Nanduru, and K. Vasamshetty, "Development of Injection Attacks Toolbox in MATLAB/Simulink for Attacks Simulation in Industrial Control System Applications," in *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, vol. 1, 2019, pp. 1192–1198.

[197] M. Smache, A. Olivereau, T. Franco-Rondisson, and A. Tria, "Autonomous Detection of Synchronization Attacks in the Industrial Internet Of Things," in *2019 IEEE 38th International Performance Computing and Communications Conference (IPCCC)*, 2019, pp. 1–9.

[198] G. Settanni, F. Skopik, A. Karaj, M. Wurzenberger, and R. Fiedler, "Protecting cyber physical production systems using anomaly detection to enable self-adaptation," in *2018 IEEE Industrial Cyber-Physical Systems (ICPS)*. IEEE, 2018, pp. 173–180.

[199] V. Sharma, G. Choudhary, Y. Ko, and I. You, "Behavior and vulnerability assessment of drones-enabled industrial internet of things (IIoT)," *IEEE Access*, vol. 6, pp. 43 368–43 383, 2018.

[200] S. Tamy, H. Belhadaoui, M. A. Rabbah, N. Rabbah, and M. Rifi, "An Evaluation of Machine Learning Algorithms To Detect Attacks in Scada Network," in *2019 7th Mediterranean Congress of Telecommunications (CMT)*, 2019, pp. 1–5.

[201] A. Wadsworth, M. I. Thanoon, C. McCurry, and S. Z. Sabatto, "Development of IIoT Monitoring and Control Security Scheme for Cyber Physical Systems," in *2019 SoutheastCon*, 2019, pp. 1–5.

[202] T. Wang, P. Wang, S. Cai, Y. Ma, A. Liu, and M. Xie, "A Unified Trustworthy Environment Establishment based on Edge Computing in Industrial IoT," *IEEE Transactions on Industrial Informatics*, pp. 1–1, 2019.

[203] H. Yao, P. Gao, P. Zhang, J. Wang, C. Jiang, and L. Lu, "Hybrid Intrusion Detection System for Edge-Based IIoT Relying on Machine-Learning-Aided Detection," *IEEE Network*, vol. 33, no. 5, pp. 75–81, 2019.

[204] T. Yu, V. Sekar, S. Seshan, Y. Agarwal, and C. Xu, "Handling a trillion (unfixable) flaws on a billion devices: Rethinking network security for the Internet-of-Things," in *Proceedings of HotNets*, Philadelphia, PA, 2015, pp. 1–7.

[205] L. Zhou, H. Guo, and G. Deng, "A fog computing based approach to DDoS mitigation in IIoT systems," *Computers & Security*, vol. 85, pp. 51 – 62, 2019.

[206] M. Zolanvari, M. A. Teixeira, and R. Jain, "Effect of imbalanced datasets on security of industrial IoT using machine learning," in *2018 IEEE International Conference on Intelligence and Security Informatics (ISI)*. IEEE, 2018, pp. 112–117.

[207] M. Zolanvari, M. A. Teixeira, L. Gupta, K. M. Khan, and R. Jain, "Machine Learning-Based Network Vulnerability Analysis of Industrial Internet of Things," *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 6822–6834, 2019.

[208] E. Zugasti, M. Iturbe, I. Garitano, and U. Zurutuza, "Null is not always empty: Monitoring the null space for field-level anomaly detection in industrial IoT environments," in *2018 Global Internet of Things Summit (GIoTS)*. IEEE, 2018, pp. 1–6.

[209] Y. Ai, M. Cheffena, T. Ohtsuki, and H. Zhuang, "Secrecy Performance Analysis of Wireless Sensor Networks," *IEEE Sensors Letters*, vol. 3, no. 5, pp. 1–4, 2019.

[210] C. Alcaraz, R. Roman, P. Najera, and J. Lopez, "Security of industrial sensor network-based remote substations in the context of the Internet of Things," *Ad Hoc Networks*, vol. 11, pp. 1091–1104, 2013.

[211] A. Bluschke, W. Bueschel, M. Hohmuth, F. Jehring, R. Kaminski, K. Klamka, S. Koepsell, A. Lackorzynski, T. Lackorzynski, M. Matthews, P. Rietzsch, A. Senier, P. Sieber, V. Ulrich, R. Wiggers, and J. Wolter, "fastvpn - secure and flexible networking for industry 4.0," in *Broadband Coverage in Germany; 12th ITG-Symposium*. VDE, 2018, pp. 1–8. [Online]. Available: https://imld.de/en/research/research-projects/fastvpn/

[212] M. Cheminod, L. Durante, L. Seno, F. Valenza, A. Valenzano, and C. Zunino, "Leveraging sdn to improve security in industrial networks," in *2017 IEEE 13th International Workshop on Factory Communication Systems (WFCS)*. IEEE, 2017, pp. 1–7.

[213] 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)*. IEEE, 2013, pp. 676–681.

[214] S. Jeong, W. Na, J. Kim, and S. Cho, "Internet of things for smart manufacturing system: Trust issues in resource allocation," *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 4418–4427, 2018.

[215] C. Lipps, M. Strufe, S. B. Mallikarjun, and H. D. Schotten, "Physical Layer Security for IIoT and CPPS: A Cellular-Network Security Approach," in *Mobile Communication - Technologies and Applications; 24. ITG-Symposium*, 2019, pp. 1–5.

[216] C. Lipps, D. Krummacker, and H. D. Schotten, "Securing Industrial Wireless Networks: Enhancing SDN with PhySec," in *2019 Conference on Next Generation Computing Applications (NextComp)*, 2019, pp. 1–7.

[217] J. O'Raw, D. Laverty, and D. J. Morrow, "Securing the Industrial Internet of Things for Critical Infrastructure (IIoT-CI)," in *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*, 2019, pp. 70–75.

[218] P. Hu, "A system architecture for software-defined industrial internet of things," in *2015 IEEE International Conference on Ubiquitous Wireless Broadband (ICUWB)*. IEEE, 2015, pp. 1–5.

[219] T. Kobzan, S. Schriegel, S. Althoff, A. Boschmann, J. Otto, and J. Jasperneite, "Secure and time-sensitive communication for remote process control and monitoring," in *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1. IEEE, 2018, pp. 1105–1108.

[220] 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)*, 2019, pp. 1–8.

[221] G. Marchetto, R. Sisto, J. Yusupov, and A. Ksentinit, "Formally verified latency-aware vnf placement in industrial internet of things," in *2018 14th IEEE International Workshop on Factory Communication Systems (WFCS)*. IEEE, 2018, pp. 1–9.

[222] M. Alaluna, L. Ferrolho, J. R. Figueira, N. Neves, and F. M. V. Ramos, "Secure Multi-Cloud Virtual Network Embedding," *arXiv e-prints*, p. arXiv:1703.01313, 2017.

[223] M. Chen, Y. Miao, Y. Hao, and K. Hwang, "Narrow band internet of things," *IEEE Access*, vol. 5, pp. 20 557–20 577, 2017.

[224] F. Kurtz, C. Bektas, N. Dorsch, and C. Wietfeld, "Network slicing for critical communications in shared 5g infrastructures - an empirical evaluation," in *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*. IEEE, 2018, pp. 393–399.

[225] T. Ulz, T. Pieber, C. Steger, S. Haas, and R. Matischek, "Sneakernet on wheels: Trustworthy nfc-based robot to machine communication," in *2017 IEEE International Conference on RFID Technology Application (RFID-TA)*. IEEE, 2017, pp. 260–265.

[226] Q. Wang, H. Dai, H. Wang, G. Xu, and A. K. Sangaiah, "UAV-enabled friendly jamming scheme to secure industrial Internet of Things," *Journal of Communications and Networks*, vol. 21, no. 5, pp. 481–490, 2019.

[227] N. Accettura and G. Piro, "Optimal and secure protocols in the ietf 6tisch communication stack," in *2014 IEEE 23rd International Symposium on Industrial Electronics (ISIE)*. IEEE, 2014, pp. 1469–1474.

[228] D. Dujovne, T. Watteyne, X. Vilajosana, and P. Thubert, "6TiSCH: deterministic IP-enabled industrial internet (of things)," *IEEE Communications Magazine*, vol. 52, no. 12, pp. 36–41, 2014.

[229] R. S. Sinha, Y. Wei, and S.-H. Hwang, "A survey on lpwa technology: Lora and nb-IoT," *Ict Express*, vol. 3, no. 1, pp. 14–21, 2017.

[230] H. C. Pöhls, V. Angelakis, S. Suppan, K. Fischer, G. Oikonomou, E. Z. Tragos, Rodrigo Diaz Rodriguez, and T. Mouroutis, "Rerum: Building a reliable IoT upon privacy- and security- enabled smart objects," in *2014 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, 2014, pp. 122–127.

[231] H. Aranha, M. Masi, T. Pavleska, and G. P. Sellitto, "Securing Mobile e-Health Environments by Design: A Holistic Architectural Approach," in *2019 International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, 2019, pp. 1–6.

[232] H. Boyes, B. Hallaq, J. Cunningham, and T. Watson, "The industrial internet of things (IIoT): An analysis framework," *Computers in Industry*, vol. 101, pp. 1–12, 2018.

[233] B. Craggs, A. Rashid, C. Hankin, R. Antrobus, O. Serban, and N. Thapen, "A reference architecture for IIoT and industrial control systems testbeds," in *Living in the Internet of Things (IoT 2019)*, 2019, pp. 1–8.

[234] M. Eckhart, A. Ekelhart, A. LÃijder, S. Biffl, and E. Weippl, "Security Development Lifecycle for Cyber-Physical Production Systems," in *IECON 2019 - 45th Annual Conference of the IEEE Industrial Electronics Society*, vol. 1, 2019, pp. 3004–3011.

[235] H. Flatt, S. Schriegel, J. Jasperneite, H. Trsek, and H. Adamczyk, "Analysis of the cyber-security of industry 4.0 technologies based on rami 4.0 and identification of requirements," in *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 2016, pp. 1–4.

[236] Y. Huang, W. Sun, and Y. Tang, "3aRAM: A 3-Layer AHP-Based Risk Assessment Model and its Implementation for an Industrial IoT Cloud," in *2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, 2019, pp. 450–457.

[237] R. A. Isbell, C. Maple, B. Hallaq, and H. Boyes, "Development of a capability maturity model for cyber security in IIoT enabled supply chains," in *Living in the Internet of Things (IoT 2019)*, 2019, pp. 1–8.

[238] I. Ivkic, A. Mauthe, and M. Tauber, "Towards a Security Cost Model for Cyber-Physical Systems," in *2019 16th IEEE Annual Consumer Communications Networking Conference (CCNC)*, 2019, pp. 1–7.

[239] V. Kharchenko, S. Dotsenko, O. Illiashenko, and S. Kamenskyi, "Integrated Cyber Safety Security Management System: Industry 4.0 Issue," in *2019 10th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, 2019, pp. 197–201.

[240] A. Kondeva, V. Nigam, H. Ruess, and C. Carlan, "On Computer-Aided Techniques for Supporting Safety and Security Co-Engineering," in *2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, 2019, pp. 346–353.

[241] L. Liang, Y. Liu, Y. Yao, T. Yang, Y. Hu, and C. Ling, "Security challenges and risk evaluation framework for industrial wireless sensor networks," in *2017 4th International Conference on Control, Decision and Information Technologies (CoDIT)*. IEEE, 2017, pp. 0904–0907.

[242] J. M. Mcginthy and A. J. Michaels, "Secure Industrial Internet of Things Critical Infrastructure Node Design," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8021–8037, 2019.

[243] N. Mohamed and J. Al-Jaroodi, "Applying Blockchain in Industry 4.0 Applications," in *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*, 2019, pp. 0852–0858.

[244] S. Pasandideh, L. Gomes, and P. Malïf£¡, "Improving Attack Trees Analysis using Petri Net modeling of Cyber-Attacks," in *2019 IEEE 28th International Symposium on Industrial Electronics (ISIE)*, 2019, pp. 1644–1649.

[245] R. Sharpe, K. van Lopik, A. Neal, P. Goodall, P. P. Conway, and A. A. West, "An industrial evaluation of an Industry 4.0 reference architecture demonstrating the need for the inclusion of security and human components," *Computers in Industry*, vol. 108, pp. 37 – 44, 2019.

[246] L. Shu, M. Mukherjee, M. Pecht, N. Crespi, and S. N. Han, "Challenges and research issues of data management in IoT for large-scale petrochemical plants," *IEEE Systems Journal*, vol. 12, no. 3, pp. 2509–2523, 2018.

[247] "TrustZone," [Accessed 2 Jul. 2019]. [Online]. Available: https://developer.arm.com/ip-products/security-ip/trustzone

[248] "Intel® Software Guard Extensions," [Accessed 2 Jul. 2019]. [Online]. Available: https://software.intel.com/en-us/sgx

[249] I. O. for Standardization/International Electrotechnical Commission *et al.*, "Information technology-Trusted Platform Module–Part 1: Overview," *International Standard, ISO/IEC*, pp. 11 889–1.

[250] "Axiomtek's Fanless Embedded System with TPM 1.2 and Flexible Expansions," [Accessed 2 Jul. 2019]. [Online]. Available: https://www.axiomtek.com/Default.aspx?MenuId=News&FunctionId=NewsView&ItemId=12845

[251] "CC2652R SimpleLink Multiprotocol 2.4-GHz Wireless MCU," 2019. [Online]. Available: http://www.ti.com/lit/ds/symlink/cc2652r.pdf

[252] 3GPP, "The 3rd Generation Partnership Project Website," Tech. Rep. [Online]. Available: https://www.3gpp.org

[253] H. Kagermann, W. Wahlster, and J. Helbig, "Recommendations for Implementing the Strategic Initiative INDUSTRIE 4.0 – Securing the Future of German Manufacturing Industry," acatech – National Academy of Science and Engineering, München, Final Report of the Industrie 4.0 Working Group, apr 2013. [Online]. Available: http://forschungsunion.de/pdf/industrie_4_0_final_report.pdf

[254] M. De Donno, K. Tange, and N. Dragoni, "Foundations and evolution of modern computing paradigms: Cloud, IoT, edge, and fog," *IEEE Access*, vol. 7, pp. 150 936–150 948, 2019.

[255] M. De Donno, A. Giaretta, N. Dragoni, A. Bucchiarone, and M. Mazzara, "Cyber-storms come from clouds: Security of cloud computing in the IoT era," *Future Internet*, vol. 11, no. 6, p. 127, 2019.

[256] I. S. Association *et al.*, "IEEE 1934-2018-IEEE standard for adoption of OpenFog reference architecture for Fog Computing," 2018.

[257] R. Skillern, "Intel® SGX Data Protections Now Available for Mainstream Cloud Platforms," Tech. Rep., 2019. [Online]. Available: https://itpeernetwork.intel.com/sgx-data-protection-cloud-platforms/

[258] M. Aazam, S. Zeadally, and K. A. Harras, "Deploying fog computing in industrial internet of things and industry 4.0," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4674–4682, 2018.

[259] M. De Donno, J. M. D. Felipe, and N. Dragoni, "AntibIoTic 2.0: A Fog-based Anti-Malware for Internet of Things," in *Proceedings of the European Workshop on Security and Privacy in Edge Computing (EuroSPEC 2019), located at IEEE Conference on Security & Privacy (EuroS&P*, 2019.

[260] M. De Donno and N. Dragoni, "Combining AntibIoTic with Fog Computing: AntibIoTic 2.0," in *Proceedings of the 3rd International Conference on Fog and Edge Computing (ICFEC)*. IEEE, 2019, pp. 1–6.

[261] M. De Donno, N. Dragoni, A. Giaretta, and M. Mazzara, "AntibIoTic: protecting IoT devices against DDoS attacks," in *International Conference in Software Engineering for Defence Applications*. Springer, 2016, pp. 59–72.

[262] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niakanlahiji, J. Kong, and J. P. Jue, "All one needs to know about fog computing and related edge computing paradigms: A complete survey," *Journal of Systems Architecture*, vol. 98, pp. 289 – 330, 2019.

**Koen Tange** is a PhD student at DTU Compute, Technical University of Denmark (Denmark), under the supervision of Prof. Nicola Dragoni. He received a B.Sc in Software Science at Eindhoven University of Technology (TU/e), Eindhoven, the Netherlands, in 2016. Afterwards, in 2018, he received a joint M.Sc. degree in Engineering, Security, and Mobile Computing from the Technical University of Denmark, Lyngby, Denmark, and Aalto University, Espoo, Finland, as part of the Nordic Masters Programme in Security and Mobile Computing. His research interests include information security, Fog computing, trusted hardware, and distributed systems.

**Michele De Donno** is a PhD Student at DTU Compute, Technical University of Denmark (Denmark), under the supervision of Prof. Nicola Dragoni. He got a M.Sc. Degree in Computer Engineering from Politecnico di Torino, Turin, Italy, in 2017. His main research interests include cyber-security, networking, distributed systems, Internet-of-Things, and Fog computing.

**Xenofon Fafoutis** (S'09-M'14-SM'20) received a PhD degree in Embedded Systems Engineering from the Technical University of Denmark in 2014; an MSc degree in Computer Science from the University of Crete (Greece) in 2010; and a BSc in Informatics and Telecommunications from the University of Athens (Greece) in 2007. From 2014 to 2018, he held various researcher positions at the University of Bristol (UK), and he was a core member of SPHERE: UK's flagship Interdisciplinary Research Collaboration on Healthcare Technology. He is currently an Associate Professor with the Embedded Systems Engineering (ESE) section of the Department of Applied Mathematics and Computer Science of the Technical University of Denmark (DTU Compute). His research interests primarily lie in Wireless Embedded Systems as an enabling technology for Digital Health, Smart Cities, and the (Industrial) Internet of Things (IoT).

**Nicola Dragoni** is Professor in Secure Pervasive Computing at DTU Compute, Technical University of Denmark, where he also serves as Deputy Head of the PhD School. He is also part-time Professor in Computer Engineering at Centre for Applied Autonomous Sensor Systems, Örebro University, Sweden, and he is affiliated with the Copenhagen Center for Health Technology (CACHET) and the Nordic IoT Hub. Nicola Dragoni received the M.Sc. (cum laude) and Ph.D. degrees in computer science from the University of Bologna, Italy. His main research interests include pervasive computing and cybersecurity, with current focus on Internet-of-Things, Fog computing and mobile systems. He has co-authored 110+ peer-reviewed articles and he has edited 3 journal special issues and 1 book. He is active in a number of national and international projects.

Paper C

# Fault-tolerant Clock Synchronization using Precise Time Protocol Multi-Domain Aggregation

E. Kyriakakis, K. Tange, N. Reusch, E. O. Zaballa, X. Fafoutis, M. Schoeberl, and N Dragoni. "Fault-tolerant Clock Synchronization using Precise Time Protocol Multi-Domain Aggregation." In: *2021 IEEE 24th International Symposium on Real-Time Distributed Computing (ISORC)*. IEEE, 2021. DOI: 10.1109/ISORC52013.2021.00025

2021 IEEE 24th International Symposium on Real-Time Distributed Computing (ISORC)

# Fault-tolerant Clock Synchronization using Precise Time Protocol Multi-Domain Aggregation

Eleftherios Kyriakakis*§, Koen Tange *§, Niklas Reusch*, Eder Ollora Zaballa†,
Xenofon Fafoutis*, Martin Schoeberl*, and Nicola Dragoni*
{elky, kpta, nikre, eoza, xefa, masca, ndra}@dtu.dk
*DTU Compute, †DTU Fotonik,
Technical University of Denmark

2021 IEEE 24th International Symposium on Real-Time Distributed Computing (ISORC) | 978-1-6654-0414-3/21/$31.00 ©2021 IEEE | DOI: 10.1109/ISORC52013.2021.00025

***Abstract*—Distributed real-time systems often rely on time-triggered communication and task execution to guarantee end-to-end latency and time-predictable computation. Such systems require a reliable synchronized network time to be shared among end-systems. The IEEE 1588 Precision Time Protocol (PTP) enables such clock synchronization throughout an Ethernet-based network. While security was not addressed in previous versions of the IEEE 1588 standard, in its most recent iteration (IEEE 1588-2019), several security mechanisms and recommendations were included describing different measures that can be taken to improve system security and safety. One proposal to improve security and reliability is to add redundancy to the network through modifications in the topology. However, this recommendation omits implementation details and leaves the question open of how it affects synchronization quality.**

**This work investigates the quality impact and security properties of redundant PTP deployment and proposes an observation window-based multi-domain, PTP end-system, design to increase fault-tolerance and security. We implement the proposed design inside a discrete-event network simulator and evaluate its clock synchronization quality using two test-case network topologies with simulated faults.**

***Index Terms*—time-sensitive networking, precise time protocol, clock synchronization, fault tolerance, availability, safety**

## I. Introduction

Modern Cyber-Physical Systems (CPS) are becoming increasingly connected to the Internet through the advancements of Fog Computing and Industrial Internet of Things. Thus nowadays, security becomes an essential factor in the design of such systems, in addition to the traditional safety and reliability requirements [1], [2].

Time-triggered communication is often used in distributed CPS that require strict guarantees on the timing of messages. Such systems need a high precision global notion of time to be shared among the nodes in the network to achieve synchronous scheduled communication and computation [3], [4]. Time-Sensitive Networking (TSN) [5] is a newly developed standard that aims to enable deterministic real-time communication for mixed-criticality traffic while preserving the high-bandwidth capabilities of Ethernet. It is developed by the TSN Task Group as an extension to the 802.1 Ethernet standard and consists of many sub-standards for different components. TSN uses a profile (802.1 AS-Rev [6]) of the IEEE 1588 Precision Time Protocol (PTP) standard [7] to enable accurate clock synchronization. Although PTP has been in use for decades, recent research indicates that this protocol's security and safety aspects have been overlooked, leaving it vulnerable to time synchronization attacks [8]. An attack on an automated factory's network time would disrupt the communication and computation schedule leading to missed deadlines and messages. This could have catastrophic consequences both in the production line and operating machinery, as well as possibly endanger human lives or the environment.

To address some of these issues, the IEEE Precise Networked Clock Synchronization Working Group has included various security measures in the updated IEEE 1588-2019 standard [9]. This updated standard proposes several measures involving redundancy to mitigate security and safety issues due to unavailable links. To support the proposed redundancy, the standard recommends using a voting algorithm to derive a converged clock offset from the multiple domains. However, no further information is given, leaving the algorithm's choice and its implementation to the user.

While distributed consensus and voting algorithms are extensively studied [10], to our knowledge, no such work exists in the context of highly time-sensitive PTP networks. We explore the concept of fault-tolerant clock synchronization within TSN and propose a multi-domain synchronization scheme that uses redundant paths combined with frame aggregation and a time-based observation window to achieve secure and fault-tolerant operation. We evaluate the proposed approach by simulating three test-case network topologies in a discrete-event network simulation tool OMNeT++ [11]. The achieved clock synchronization is compared against standard PTP end-systems and evaluated regarding two metrics, accuracy as average mean and jitter as the standard deviation of the clock offset. The proposed multi-domain design is able to preserve microsecond precision despite the existence of network failures. The contributions of this paper are:

- A fault-tolerant PTP end-system design that supports multiple synchronization domains.
- A timed observation window mechanism that aims to increase security by filtering received frames.
- A comparative analysis of clock synchronization quality in different test-case scenarios with faults.

The remainder of this paper is structured in 6 sections:

§These authors contributed equally to this work.

2375-5261/21/$31.00 ©2021 IEEE
DOI 10.1109/ISORC52013.2021.00025

Section II presents the fundamental concepts of PTP and fault-tolerant synchronization and introduces the problem statement. Section III discusses the related work in PTP security and fault tolerance. Section IV presents the proposed multi-domain end-system architecture and discusses the required network topology. Section V evaluates the proposed multi-domain design and compares its performance against the standard PTP mechanisms by simulating different test-cases with synthetic scenarios. Section VI provides a discussion on the safety and security implications of the proposed multi-domain aggregation mechanism. Section VII presents the planned future extensions of this work. Section VIII summarizes the presented work and concludes the paper.

## II. Background

### A. Fault-Tolerant Clock Synchronization

Precise and fault-tolerant time synchronization is an operational requirement of distributed safety-critical real-time systems, such as those found in aerospace and automotive industry. Redundancy is the key to tolerate Byzantine faults in these systems, as any master clock can exhibit arbitrary behaviour and provide false readings of its local clock to connected systems. Consequently, slave clocks can misinterpret this information either because accurate convergence algorithms have not been implemented or simply because the in-place redundancy is not sufficient. It can lead to drift in the relative clock offset of the network-wide time base.

This effect has been described in research [12], [13], where the authors have analyzed the need for $3f+1$ nodes available in a distributed system that can tolerate $f$ faults and have provided static bounds for different convergence algorithms. The most predominant algorithm of these is the Fault-Tolerant Average (FTA), which was first introduced in [14] and is incorporated in the fault-tolerant clock synchronization of TTEthernet [15] that is now part of the aerospace standard AS6802 [16]. In this work, we try to incorporate FTA principles in PTP and evaluate its performance in TSN networks as a means to provide fault-tolerant multi-domain clock synchronization.

### B. IEEE 1588-2019 Precise Time Protocol

PTP is a hierarchical clock synchronization protocol based on a periodic exchange of Ethernet frames that estimates the clock offset between end-system ports configured as slaves and masters [17]. Typically, a PTP stack is assigned to each PTP port and is responsible for executing the protocol. A mechanism, called a clock servo, is responsible for correcting the device clock using a proportional-integral filter [18]. The PTP stack on a slave port calculates the time difference from a master by collecting four timestamps using four respective frames:

1) SYNC, from master to slave
2) FOLLOW_UP, from master to slave
3) DELAY_REQ, from slave to master
4) DELAY_REPLY, from master to slave

Moreover, precise time-stamping of the received/sent frames is a crucial part of the protocol, as it directly influences the precision of the estimated clock offset. Select hardware units can be used for this purpose [19] . In the rest of this work, we assume that such time-stamping units are available in all end-systems.

PTP allows for multiple masters to exist, but only one master's synchronization frames are used to calibrate an end system's internal clock at each given synchronization cycle. This selection is made using the best master clock algorithm (BMCA). The BMCA works by comparing an arbitary value, which represents the remote clock quality, connected network end-systems advertised that in dedicated periodic frames called ANNOUNCE frames. From this information it derives the best clock and then it compares that to the quality of its local clock to determine its role as a master or a slave.

The IEEE-1588-2019 standard adds several security features to PTP [9]. Most notably, it adds support for multiple types of authenticated encryption, addressing many of the security concerns that were present in its predecessor, IEEE-1588-2008 [7]. However, none of the introduced features protects against delay attacks, nor do they consider faulty master nodes (e.g., compromised by a malicious party). A malicious master node might try to influence the system time by announcing high accuracy during the BMCA, subsequently moving the time window once it has been elected. Delay attacks assume that an attacker can control a link, and might delay messages for an indefinite amount of time. To mitigate the impact of such attacks, the IEEE-1588-2019 standard only includes two recommendations: to deploy redundant master clocks; and to deploy redundant network topologies. The first recommendation works without any alterations to the protocol. As the PTP protocol is a distributed algorithm, it will eventually select one of the redundant master clocks if the primary one fails; however, this can introduce significant time overhead that leads to jitter. The second recommendation stands out: a PTP system distils a logical minimum spanning tree topology with the elected master clock as a root, and all slaves (i.e., consumers of the synchronization signal) as leaves. A minimum spanning tree does not allow multiple paths between any two nodes to exist by its very definition. The solution to this is to run multiple PTP domains in parallel, ensuring that they choose different physical network paths for their tree topology. A PTP domain is a numerical identifier included in every protocol message. It allows multiple PTP systems to operate on one network without interfering with other PTP systems.

A multi-domain setup combines neatly with the first recommendation of using multiple master clocks. With multiple parallel domains, every slave system needs to execute a deterministic voting algorithm to arrive at the same approximate time. However, the IEEE-1588-2019 standard does not recommend any voting algorithms. Additionally, it is left unclear what the performance impact and effectiveness of these measures will be. Therefore, we attempt to fill this knowledge gap by analyzing two voting algorithms' performance in a simulated PTP system. Further, we explore the impact of link failures on timing accuracy during the execution of a PTP system both with and without redundancy in place.

## III. Related Work

In the broad spectrum of network attacks related to PTP, disrupting the synchronization is the primary goal. Lack of message authentication is one of the main attack vectors to break master-slave synchronization. The authors of [8] analyze the security risks associated with PTP by building a testbed that shows synchronization disruption between PTP devices. The tests conducted include master spoof attacks (spoofing ANNOUNCE and SYNC packets), ANNOUNCE DoS attacks (spamming target slave) and master clock takeover attacks. Similarly, Lisova [20] presents a threat model that shows an attack classification that lists several PTP clock synchronization attacks (e.g. replay and delay attacks, flooding/DoS) that target availability among other factors. Lisova proposes a distributed monitoring strategy to detect if an attacker is affecting clock synchronization. While both studies [8] [20] point out existing threats to availability, the current work provides a fault-tolerant design to guarantee availability.

To tackle some of the attacks mentioned earlier, IPSec and MACSec have already been analyzed for time synchronization [21] to provide authentication, encryption, and confidentiality. However, neither provide any availability guarantees or fault tolerance against a compromised endpoint or delay attacks. We consider IPSec and MACsec complementary to the fault tolerance algorithms and mechanism discussed in this work.

In 2014, Mizrahi published an informational Request For Comments (RFC) with the requirements to secure time protocols in packet-switched networks [22]. The document presents a threat model and threat analysis that lists several attack types such as packet manipulation, spoofing or replay attacks. It focuses on listing minimum security requirements such as authentication, authorization, confidentiality. While these requirements could create a security basis for next versions of time synchronization protocols, they do not guarantee availability. Additionally, the document briefly references a few mechanisms to protect against delay attacks or attacks that degrade clock accuracy, such as using of multiple paths [23]. This RFC also proposes that outliers in received time values should be considered erroneous and be ignored. The current study aims to fill the gap of fault tolerance, resilience and availability that the RFC does not cover. Specifically, it presents an implementation and evaluates its resilience to faults.

Mizrahi presents the concept of *slave diversity* [23] to obtain high clock accuracy and reduce time error using multiple paths. Similarly, Shipiner et al. present a multi-path approach [24] that evaluates path diversity. While both studies demonstrate the applicability of multiple path time synchronization, there are significant differences with this work. First, Mizrahi [23] does not tackle the master redundancy and availability features into fault tolerance and Shipiner et al. [24] do not provide simulation and performance results. In contrast, this work, uses different PTP domains with multiple masters to guarantee availability and evaluates the fault-tolerance in simulation.

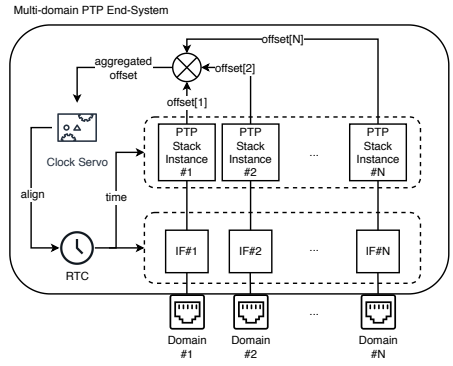



Fig. 1: Extended PTP end-system architecture to support multi-domain aggregation. Each domain uses a separate network interface and PTP stack. The calculated offsets are fed into an aggregation function, which corrects the clock.

## IV. Multi-domain Node and Algorithm Design

Our proposed approach consists of multiple design elements and considerations spread over multiple layers. Firstly, we introduce a redundant variant of a typical PTP node. A node's ability to interact with singular (i.e., non-redundant) nodes is preserved, leaving room for hybrid PTP systems. Secondly, we discuss network topology requirements that should be taken into account when designing redundant PTP systems. Finally, we describe the implemented convergence algorithms.

The design proposed in this section aims to mitigate link failures and protect against Byzantine actors on the network, but it does not guarantee the communicated messages' integrity or authenticity. It is intended to complement existing resilience and security features proposed by the IEEE-1588-2019 standard, which provide these properties.

### A. Node Architecture

A redundant PTP node has to support running PTP on $n$ domains at once. To this end, we design a node architecture that maintains $n$ parallel PTP stacks and aggregates their computed offsets. As is usual for Byzantine fault-tolerant systems, to protect against $f$ faults, $n$ should be picked as $n = 3f + 1$. Figure 1 presents the design of the proposed node. Each PTP stack is assigned an individual network interface port and executes isolated from the others. Using only one network interface is possible, but it would turn this into a bottleneck and the weakest link for each node. If the node is a slave, each stack periodically receives PTP messages that have to be aggregated somehow. Each stack distils an offset from the incoming messages. The calculated offset is combined with the latest PTP frame ingress timestamps as a tuple and fed

into a convergence algorithm. If the node is a master node, it simply has to transmit PTP messages on every domain.

The convergence algorithm aggregates the most recently received offsets for each domain within an observation window, and produces a single aggregated offset correction for the real-time clock (RTC). This convergence algorithm is transparent to the PTP stacks, the clock servo, and any applications depending on the synchronized time of the RTC.

### B. Network Topology

To effectively mitigate link/node failures and malicious PTP actor nodes, network paths for each domain should be entirely disjoint. Therefore, one can specify the main goal for the network topology is to introduce redundancy where possible. The observation window should be tuned according to the maximum expected latency of all the redundant domain paths. Thus, to minimize the observation window span , a design using redundant network paths should strive to preserve a symmetric topology with the same number of hops between slaves and master nodes. Further optimization on the asymmetry of links has been investigated by [25], [26]. Note that while a fully symmetric topology describes an ideal situation, it is not an explicit requirement. A symmetric topology allows for balanced network delays with equal worst-case end-to-end latency (WCEL), and thus it is hypothesized to lead to better convergence algorithm performance. In the remainder of this work, we thus assume a fully symmetric topology to explore the ideal case.

### C. Convergence Algorithms

The convergence algorithm is run on each PTP slave node individually and takes as inputs a collection of latest observed offsets from each domain PTP stack. We implement two different convergence algorithms for evaluation. The first offset aggregation algorithm is a simple averaging function (AVG) over the available offsets. The second algorithm implements a Byzantine Fault Tolerant approach (FTA) for clock synchronization.

*1) Observation Window Filtering:* Figure 2 illustrates how individual PTP frames from the different PTP stacks are converged by initiating separate observation windows. Each received SYNC, or FOLLOW_UP frame initiates a new observation window based on the ingress timestamp over which the convergence algorithm operates. Only frames within an observation window time are taken into account to calculate the converged clock offset for that specific point in time. The duration of the observation window, controls the accepted time difference threshold of the received master frame timestamps from the last received PTP frame timestamp. This parameter should be tuned proportionally to the WCEL that the PTP master frames can experience, i.e. the longest path delay between a redundant master and the receiving slave.

The windowed decision algorithm is listed in Algorithm 1. This algorithm takes a new (incoming) offset $o$ from its local clock as input, together with its ingress time $i$ and PTP domain $d$. The algorithm's output is an approximate offset and ingress

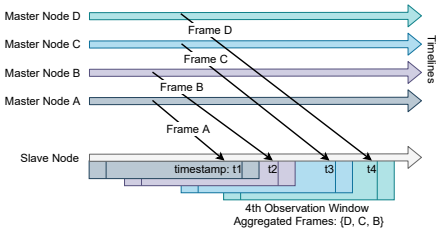



Fig. 2: Observation windows are generated by new SYNC/-FOLLOW_UP frames. Received frames that are within the time window are used in the aggregated offset calculation.

**Algorithm 1** Windowed Decision Algorithm

1: **procedure** WINDOWEDDECISION($o, i, d$) ▷ Executes a windowed decision algorithm using the latest received timestamps
**State:** $S$ ▷ A table $d \rightarrow (o_d, i_d)$ mapping all domains $d \in D$ to (offset, ingress) tuples
2: $S[d] \leftarrow (o, i)$
3: $S' \leftarrow \{x \rightarrow (o_x, i_x) \in S \text{ where } |i - i_x| \leq WINDOW\}$
4: $i_a \leftarrow \begin{cases} 0 \text{ if } |S'| = 0 \\ \frac{\Sigma_{x \in S'} i_x}{|S'|} \text{ otherwise} \end{cases}$
5: $o_a \leftarrow$ FTA($S'$) or AVG($S'$)
6: **return** $(o_a, i_a)$
7: **end procedure**

time, which can be used by the clock servo to correct the RTC. First, it stores the tuple $(o, i)$ in a table structure using the domain as an index, ensuring that only one offset per domain is considered. After this, the table $S$ is filtered to $S'$, excluding offsets that were not received within a given delta `WINDOW` from the new ingress timestamp. Then, the ingress of all offsets in $S'$ are averaged to $i_a$, and an approximate offset $o_a$ is calculated using either the FTA or the AVG algorithm.

*2) Averaging Algorithm (AVG):* The AVG consists of a simple averaging function that extracts all offsets from the given map and returns the average of these, or 0 if there are no offsets.

*3) Fault Tolerant averaging Algorithm (FTA):* The FTA [14] is an algorithm that provides bounded clock synchronization even in the presence of faulty and possibly malicious master clocks (see also Section II). Algorithm 2 describes the implemented FTA algorithm.

In the general case where $k$ faults should be tolerated, this algorithm drops the earliest and last $k$ offsets and averages the remaining offsets. First, usable offsets are extracted from the given map structure $S'$, and special assignments are made for the $2k$ most extreme offsets. Then, it distinguishes 3 cases: firstly, if there is only one offset, we return that offset; secondly, if there are only two offsets, their average is returned, and finally, if there are three or more offsets, it drops the extremes

**Algorithm 2** Fault Tolerant Algorithm

**procedure** FTA($S$) ▷ Executes a fault-tolerant convergence algorithm over a set of offsets
  **if** $|S| = 0$ **then**
    **return** 0
  **end if**
  $O = \{o_x | x \in S\}$
  $o_{min} \leftarrow k$ earliest offsets in $O$
  $o_{max} \leftarrow k$ latest offsets in $O$
  **if** $|O| = 1$ **then**
    **return** $o_{min}$
  **else if** $|O| = 2$ **then**
    **return** $\frac{o_{min}+o_{max}}{2}$
  **else if** $|O| \geq 3$ **then**
    $O' \leftarrow O \setminus \{o_{min}, o_{max}\}$
    **return** $\frac{\sum_{x \in O'} x}{|O'|}$
  **end if**
**end procedure**

and returns the average of the remaining offsets.

The first two cases will usually only trigger if there are remote failures, and the system does not receive enough offsets. In this case, the failures are regarded as faulty nodes, thereby exceeding the number of tolerated faults, and the most we can do is a best-effort execution of the algorithm. The third case covers the standard execution of the algorithm. By dropping the $2k$ outer offsets, adversaries are forced to operate within a limited time offset range. By taking the average of the remaining offsets, adversaries would have to control more master nodes than our model tolerates to have a considerable effect on the aggregated offset. For a formal proof, we refer the interested reader to [14], [12].

## V. Evaluation

To demonstrate the fault-tolerance of the proposed redundant PTP scheme and evaluate the synchronization quality, we generate two test-case network topologies [1]. These topologies are simulated witin the OMNeT++-4.6 [11] discrete-event network simulator using our extended version [2] of a PTP simulation library named LibPTP [27]. LibPTP [28] is a complete simulation framework for OMNeT++ that allows the simulation of standard PTP devices. To the RTC oscillator noise and yield more realistic clock drift results, we utilize a Power-law noise library (LibPLN [29]) as described in the LibPTP documentation [27]. All experiments are done on a 64-bit i7-7700HQ CPU system running at 2.8 GHz with 32GB RAM.

### A. Simulation parameters

The presented experiments are based on the following assumptions. Firstly, we assume that every node has multiple network interfaces, one for each domain, which is in line with the standard's recommendations, where it is advised that each domain operates over a separate network interface. Secondly, to optimize the simulation time and isolate the PTP evaluation, we assume that the network is used exclusively by PTP, so no other network traffic is simulated in the experiments. Empirically, we assume that every link has a bit-rate of 1 Gbps and is 1 meter long. Finally, every PTP stack uses the recommended gPTP profile for TSN [30] as shown in Table I and a peer-to-peer (P2P) delay mechanism.

[1] https://github.com/dtu-ese/ptp_multidomain
[2] https://github.com/dtu-ese/libPTP

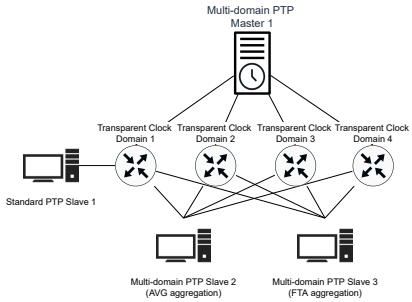



Fig. 3: First test-case network topology of single multi-domain PTP master on four isolated redundant domain paths. The domains are isolated using four different switches.

TABLE I: PTP port profile options. Values correspond to the interval of the respective messages in seconds and are represented as powers of two.

| Parameter | Value |
|---|---|
| logAnnounceInterval | 1 |
| announceReceiptTimeout | 3 |
| logSyncInterval | -3 |
| logMinDelayReqInterval | -3 |
| logMinDelayReqInterval | -3 |

### B. Test-case 1: Single PTP master on four redundant domains

This experiment aims to evaluate the stability of the proposed multi-domain aggregation scheme using the custom design of Figure 1 for both master and slave nodes. We generate a synthetic topology with three nodes and four switches as shown in Figure 3. A single multi-domain PTP master is connected to four redundant transparent clock nodes over four different domains. We integrate three different types of PTP slaves in the network: (A) a standard PTP slave connected only to the first transparent clock switch (domain), (B) a multi-domain PTP slave that uses the AVG algorithm and is connected to all domains and (C) a multi-domain PTP slave that uses the proposed FTA algorithm and connects to all domains.

We evaluate the synchronization quality in terms of average mean clock offset and standard deviation using a synthetic

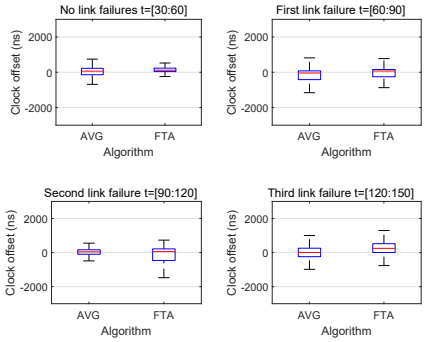


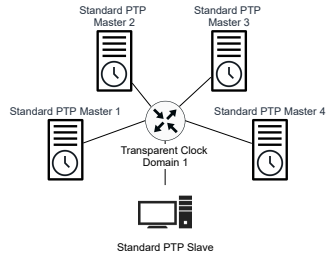

Fig. 4: Comparison of the mean clock offset and std. deviation measurements through the link failures of the experimental test-case 1 (see Section V-B) with topology from Figure 3.

scenario. We simulate a simple scenario of consecutive link failures where at 60 seconds the first link between Master 1 and Transparent Clock 1 is disconnected. The rest of the links between Master 1 and the transparent clocks are disconnected/fail similarly in intervals of 30 seconds. We simulate the scenario for a total run-time of 180 seconds.

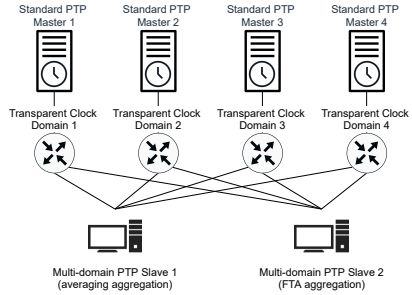
Figure 4 compares the measured mean clock offset and jitter of the two clock servo aggregation methods (AVG and FTA). Although the mean of the AVG and FTA aggregation methods are similar when no failures occur, we measure significantly less jitter using FTA throughout the experiment's run-time, resulting in more predictable clock synchronization. This is likely due to the nature of the FTA: outliers are discarded , ensuring that the system will take the average of the most consistent master clocks. If there are some master clocks that drift at different rates, or if these clock oscillators are very noisy, then it is likely that they are often discarded for the aggregated timestamp.

### *C. Test-case 2: Four PTP masters on four redundant domains*

For the second test-case, we generate and simulate two network topologies comparing the standard BMCA against the proposed multi-domain scheme. The first topology (see Figure 5a) has four PTP master capable standard nodes and a standard node that is configured as a PTP slave. All nodes operate over the same domain and are connected through a transparent clock switch in a star topology. The second topology (see Figure 5b) has four standard PTP masters connected and two redundant PTP slave nodes. The PTP masters operate over four different domains and are respectively connected to four different transparent clock switches. For simplicity, we assume that individual PTP master node clocks are synchronized to each other in order for the observation window to use all available domains. This requirement is further discussed in Section VI.




(a) Connect one standard PTP slave to four PTP masters operating on the same domain. Clock selection based on BMCA.




(b) Two multi-domain PTP slaves connected to four PTP masters operating on separate domains. Clock offset calculation uses multi-domain aggregation.

Fig. 5: Second test-case parallel network topologies evaluation.

PTP slave nodes 1 and 2 use respectively, the multi-domain aggregation methods described in Section IV. We evaluate the performance of the synchronization by simulating two synthetic scenarios.

*1) Link/node failure scenario:* In the first scenario, each of the PTP masters fails in sequence every 30 seconds after the first minute of stable operation. This scenario covers a variety of real-life failures such as device failures, cable failures or denial-of-service attacks. We simulate the experiment for a total run-time of 180 seconds.

Figure 6 presents the mean time difference of the three PTP slave nodes and compares the upper/lower bounds of the three PTP slave nodes. We observe that in contrast to Test-case 1, the standard PTP slave node can stay synchronized to the master through the consecutive link failures as it can now select a new master, from each operating domain, after each link failure. However, BMCA suffers from significant synchronization drift of more than 2 $\mu$s. The FTA and the AVG aggregation manage to achieve better clock synchronization accuracy with tighter bounds than the standard BCMA during

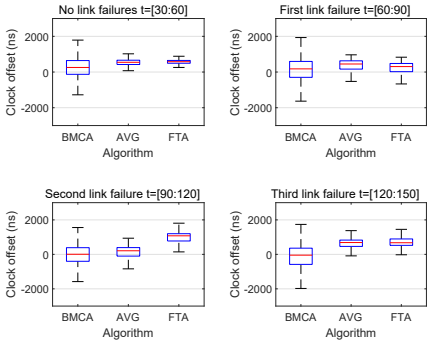



Fig. 6: Comparison of the mean clock offset and std. deviation measurements through the link failures of the experimental test-case 2 (Section V-C1) with topology from Figure 5.

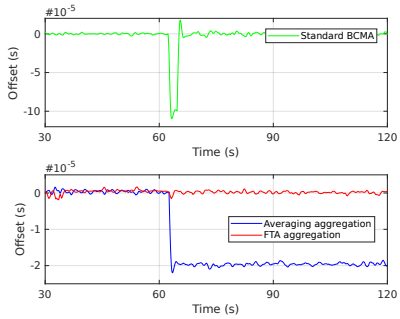



Fig. 7: Measured PTP-Slave clock offset relative to Master 1 in the test-case scenario of a new malicious PTP master node connection at t=60s (Section V-C2).

the first two link failures. As more links fail this difference between the methodologies is normalized because fewer nodes are available to aggregate.

*2) Malicious PTP master scenario:* In this scenario, we investigate the effects of a malicious PTP master clock that tries to offset the synchronized network time. The malicious end-system is connected to the network at a specific point in time and advertises that it has a higher clock quality that the existing master clocks. We emulate this scenario by simulating the instantaneous connection of a new PTP master with higher quality clock attributes after one minute of run-time at the first switch. The malicious master has its local clock offset by 100 $\mu$s than the existing masters. Due the implemented observation window's properties, a malicious master must be carefully implemented so that its local clock offset is within the observation window's bounds.

We measure this attack's effects on the clock synchronization precision of the topology's three PTP slaves relative to node Master 1. Figure 7 presents the measured results of the time-difference for the three PTP slaves. The top plot corresponds to the measurements taken from the Standard BMCA slave shown in Figure 5a. In comparison, the bottom plot presents the measurements from the multi-domain slaves shown in Figure 5b. We run the experiment for 120 seconds of simulation time.

In the standard PTP topology 5a, the newly connected malicious master is quickly elected as the best clock by the BMCA. We note a significant initial drift of the PTP slave relative to Master 1 after which the network is synchronized to the time of the malicious master clock. In the redundant PTP topology 5b, the connection of the malicious master cannot influence the independent masters as they operate in different domains. The simple approach of averaging the aggregated multi-domain master clocks is not sufficient as it is easily disturbed by the malicious clock's offset. In this scenario, the FTA proves to be the most resilient as the malicious master's relative clock offset is discarded according to Algorithm 2.

## VI. Discussion

In the evaluated test-cases, we experimentally showed that a multi-domain approach could guarantee synchronized network time availability despite network failures and malicious actions.

The platform designer has to guarantee that the PTP stack processes are isolated and cannot affect each other if the security of one PTP stack is compromised. This can be achieved using specialized hardware or sandboxing techniques such as virtualization. Considering the capabilities of modern industrial computing systems [31], the software cost for running the redundant PTP stacks in-parallel is minimal, especially if the proposed design is implemented completely in software. Preliminary results show that the CPU overhead generated by the PTP stack is less than 1% of the available computing resources. Nevertheless, the system designer should consider the additional cost for the redundant network topology based on the safety requirements of the application, as there is a significant cost increase in the number of links and switches.

The results showed that the FTA convergence algorithm could mitigate against link or node failures, as well as a compromised master node broadcasting incorrect timestamps. This work illustrates the importance of a fault-tolerant method of converging the calculated offset from the multiple PTP domains. It is worth noting that although the averaging aggregation performed as well as the FTA method, it was easily influenced by a malicious node and failed to provide secure synchronization. While our design does not enforce authentication and integrity of PTP messages by itself, the FTA algorithm leaves very little room for tampered messages, as it discards everything outside of a margin known to have a majority of correct offsets. What this approach does inherently provide is protection against various forms of DoS, timing,

and delay attacks where the number of affected links/nodes is less than $k$. As already noted in Section IV, this can be combined with the security measures proposed in IEEE-1588 (2019) to further harden the security by providing authenticity, confidentiality and integrity of messages. Thus, the combined application of the measures proposed in this work and the standardized security measures results in a secure PTP system that in addition to the standardized measures is difficult to disrupt with DoS and timing attacks.

Finally, although the proposed multi-domain PTP end-system scheme was tested with both master and slave roles, its functionality is based on the assumption that the redundant master clocks of each separate domains are synchronized to each other. This assumption is easily achievable using the proposed multi-domain PTP end-system design (see Figure 1), however standard PTP master clocks on separate devices require an external fault-tolerant mechanism of clock synchronization. One possible solution to this would be to use dual roles for master nodes, were on specific domains they would act as slaves to each other and other domains as masters in an interleaved scheme. It is hypothesized that the standard PTP boundary clock component can support this dual role functionality, but its implementation in a multi-domain network topology requires further investigation.

## VII. Future Work

As future work, we plan to explore the implementation and characterization of boundary clocks as a mechanism to enable standard PTP master clock synchronization in redundant domains. Additionally, we plan to extend the evaluated scenarios and investigate different types of attacks on PTP, such as frame spoofing. This will allow us to characterize further the proposed multi-domain design performance and identify its tuning parameters.

Moreover, one can think of a scenario where only a limited subset of all nodes are connected to multiple domains. This raises questions such as how many multi-domain nodes are necessary to meet a certain required timing accuracy? For this, we plan to explore the integration of the proposed design in boundary clocks that are connected to multiple domains, each maintaining slave clocks connected to only one of these domains.

## VIII. Conclusion

The presented work investigated the requirements for fault-tolerance in TSN clock synchronization and proposed a PTP end-system design that supports multi-domain aggregation. The proposed design implements isolated PTP stacks that use an FTA-based aggregation mechanism to correct the clock servo. This is combined with a time-based observation window for additional security. The multi-domain PTP end-system was evaluated and compared against standard PTP nodes in two scenarios with emulated link failures and possible malicious PTP masters. Overall, this work illustrated empirically the necessity for fault-tolerance in PTP and multi-domain aggregation design that manages to overcome network faults.

## Acknowledgment

This is work was part of the Fog Computing for Robotics and Industrial Automation (FORA) European Training Network (ETN) funded by the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 764785.

## References

[1] T. Pereira, L. Barreto, and A. Amaral, "Network and information security challenges within Industry 4.0 paradigm," *Procedia Manufacturing*, vol. 13, 2017.

[2] I. Studnia, V. Nicomette, E. Alata, Y. Deswarte, M. Kaaniche, and Y. Laarouchi, "Survey on security threats and protection mechanisms in embedded automotive networks," *Proc. DSN*, 2013.

[3] S. S. Craciunas and R. S. Oliver, "An overview of scheduling mechanisms for time-sensitive networks," *Proceedings of the Real-time summer school LÉcole dÉté Temps Réel (ETR)*, pp. 1551–3203, 2017.

[4] E. Kyriakakis, J. Sparsø, P. Puschner, and M. Schoeberl, "Synchronizing real-time tasks in time-aware networks: Work-in-progress," in *2020 International Conference on Embedded Software (EMSOFT)*. IEEE, 2020, pp. 15–17.

[5] *Official Website of the 802.1 Time-Sensitive Networking Task Group*, `http://www.ieee802.org/1/pages/tsn.html`, IEEE Std., 2016, accessed: 17.12.2020.

[6] *802.1AS-Rev - Timing and Synchronization for Time-Sensitive Applications*, `http://www.ieee802.org/1/pages/802.1AS-rev.html`, IEEE Std., 2016, accessed: 17.12.2020.

[7] *IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*, IEEE Std., 2008.

[8] C. DeCusatis, R. M. Lynch, W. Kluge, J. Houston, P. Wojciak, and S. Guendert, "Impact of cyberattacks on precision time protocol," *IEEE Transactions on Instrumentation and Measurement*, 2019.

[9] *IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*, IEEE Std., 2020.

[10] S. Bogomolov, C. Herrera, and W. Steiner, "Verification of fault-tolerant clock synchronization algorithms." in *ARCH@ CPSWeek*, 2016, pp. 36–41.

[11] A. Varga and R. Hornig, "An Overview of the OMNeT++ Simulation Environment," in *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*, ser. Simutools '08. Brussels, BEL: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008.

[12] H. Kopetz, A. Damm, C. Koza, M. Mulazzani, W. Schwabl, C. Senft, and R. Zainlinger, "Distributed fault-tolerant real-time systems: The mars approach," *IEEE Micro*, vol. 9, no. 1, pp. 25–40, 1989.

[13] P. Ramanathan, K. G. Shin, and R. W. Butler, "Fault-tolerant clock synchronization in distributed systems," *Computer*, vol. 23, no. 10, pp. 33–42, 1990.

[14] D. Dolev, N. A. Lynch, S. S. Pinter, E. W. Stark, and W. E. Weihl, "Reaching approximate agreement in the presence of faults," *Journal of the ACM (JACM)*, vol. 33, no. 3, pp. 499–516, 1986.

[15] W. Steiner and B. Dutertre, "The TTEthernet synchronisation protocols and their formal verification," *International Journal of Critical Computer-Based Systems 17*, vol. 4, no. 3, pp. 280–300, 2013.

[16] TTTech, *AS6802: Time-Triggered Ethernet*, SAE International Std., 2011.

[17] J. C. Eidson, *Measurement, control, and communication using IEEE 1588*. Springer Science & Business Media, 2006.

[18] G. Giorgi and C. Narduzzi, "Modeling and simulation analysis of PTP clock servo," in *2007 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication*, 2007.

[19] E. Kyriakakis, J. Sparsø, and M. Schoeberl, "Hardware Assisted Clock Synchronization with the IEEE 1588-2008 Precision Time Protocol," in *Proceedings of the 26th International Conference on Real-Time Networks and Systems*, ser. RTNS '18. New York, NY, USA: ACM, 2018, pp. 51–60. [Online]. Available: http://doi.acm.org.proxy.findit.dtu.dk/10.1145/3273905.3273920

[20] E. Lisova, "Monitoring for securing clock synchronization," Ph.D. dissertation, Mälardalen University, 2018.

[21] T. Mizrahi, "Time synchronization security using IPsec and MACsec," in *2011 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication*, 2011, pp. 38–43.

[22] T. Mizrahi, *Security Requirements of Time Protocols in Packet Switched Networks*, RFC 7384, Std. 7384, Oct. 2014. [Online]. Available: https://rfc-editor.org/rfc/rfc7384.txt

[23] T. Mizrahi, "Slave diversity: Using multiple paths to improve the accuracy of clock synchronization protocols," in *2012 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication Proceedings*, 2012, pp. 1–6.

[24] A. Shpiner, Y. Revah, and T. Mizrahi, "Multi-path time protocols," in *2013 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication (ISPCS) Proceedings*, 2013, pp. 1–6.

[25] O. Gurewitz and M. Sidi, "Estimating one-way delays from cyclic-path delay measurements," in *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No. 01CH37213)*, vol. 2. IEEE, 2001, pp. 1038–1044.

[26] S. Lee, "An enhanced IEEE 1588 time synchronization algorithm for asymmetric communication link using block burst transmission," *IEEE communications letters*, vol. 12, no. 9, pp. 687–689, 2008.

[27] W. Wallner, "Simulation of time-synchronized networks using ieee 1588-2008," Ph.D. dissertation, Wien, 2016.

[28] W. Wallner. (2016) LibPTP: A Library for PTP Simulation. https://github.com/ptp-sim/libPTP.

[29] W. Wallner. (2016) LibPLN: A Library for Efficient Powerlaw Noise Generation. https://github.com/ptp-sim/libPLN.

[30] K. Sridharan, K. Goossens, N. Concer, and H. B. Vermeulen, "Investigation of time-synchronization over ethernet in-vehicle networks for automotive applications," Master's thesis, Eindhoven: Eindhoven University of Technology, 2015.

[31] *Intel's Fog Reference Design Overview*, Intel, April 2018. [Online]. Available: https://www.intel.com/content/www/us/en/internet-of-things/fog-reference-design-overview.html

Paper D

# Fogification of Electric Drives: an Industrial Use Case

M. Barzegaran, N. Desai, J. Qian, K. Tange, B. Zarrin, P. Pop, and J. Kuusela. "Fogification of electric drives: An industrial use case." In: *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 2020. DOI: 10.1109/ETFA46521.2020.9212010

# Fogification of electric drives: An industrial use case

Mohammadreza Barzegaran[1], Nitin Desai[2], Jia Qian[1], Koen Tange[1], Bahram Zarrin[1], Paul Pop[1], and Juha Kuusela[3]

[1]DTU Compute, Technical University of Denmark, Kongens Lyngby, Denmark
[2]School of Innovation, Design and Engineering, Mälardalen University, Västerås, Sweden
[3]Danfoss Power Electronics A/S, Gråsten, Denmark

***Abstract*—Electric drives are used to control electric motors, which are pervasive in industrial applications. In this paper we propose enhancing the electric drives to fulfil the role of fog nodes within a Fog Computing Platform (FCP). Fog Computing is envisioned as a realization of future distributed architectures in Industry 4.0. We identify the system-level requirements of such an FCP, including requirements that are extracted from the current architecture of drives, which we consider as a baseline. These requirements are then used to design a system-level architecture, which we model using the Architecture Analysis & Design Language (AADL). We identify the "technology bricks" (components such as hardware, software, middleware, services, methods and tools) needed to implement the FCP. The proposed fog-based architecture is then used to implement a Conveyor Belt industrial use case. We evaluate the resulting use case on several aspects, demonstrating the usefulness of the proposed fog-based approach. By developing the electric drives as fog nodes, that we call fogification, new offerings like programmability, analytics and connectivity to customer Clouds are expected to increase the added value. Increased flexibility allows drives to assume a larger role in industrial and domestic control systems, instrumenting thus also legacy systems by using drives as the data source.**

## I. Introduction

Digitalization will affect all industries and sectors, with a potential cumulative value to society and industry of more than $100tn by 2025 [1]. This paper focuses on the industrial area, where the worldwide cumulative net value will be $1.7tn by 2020 worldwide [2]. We are at the beginning of a new industrial revolution (we will use the term Industry 4.0), which will bring increased productivity and flexibility, mass customization, reduced time-to-market, improved product quality, innovations and new business models [3].

The infrastructure of the information society is underpinned by Information Technologies (IT) such as Cloud Computing, Artificial Intelligence (AI), and Big Data. However, these technologies are not directly applicable to industrial applications [4]. The industrial area uses Operational Technologies (OT) consisting of cyber-physical systems (CPS) that monitor and control physical processes that manage, e.g., automated manufacturing, critical infrastructures, smart buildings and smart cities. These application areas are typically safety-critical and real-time, requiring guaranteed extra-functional properties, such as real-time behavior, reliability, availability, conformance to industry-specific safety standards, and security.

Industry 4.0 will only become a reality through the convergence of OT and IT [5], which are currently separated and use different computation and communication technologies. Currently, the industrial domains use OT, which is costly, completely separated from IT, and cannot support Industry 4.0 [5]. IT such as Cloud Computing has gained significant popularity and we use Cloud-based services on a daily basis as a commodity. However, Cloud Computing cannot provide dependability or quality-of-service guarantees, so it cannot be used for industrial applications. Additionally, technology paradigms such as AI and Big Data are resource-demanding and may compromise the performance of industrial applications under intensive workloads [4]. Thus, they cannot be used for these applications. Instead, a new paradigm called Fog Computing (FC), is needed as an architectural means to realize the IT/OT convergence [6]. Fog Computing is a "*system-level architecture that distributes resources and services of computing, storage, control and networking anywhere along the continuum from Cloud to Things*" [7].

This paper proposes the use of a Fog Computing Platform (FCP) for the implementation of industrial applications. We focus on the application areas where *electric drives* are present, since they are pervasive in industrial installations and are used in many domains, such as automotive, food and beverage, marine and offshore, hydraulics, refrigeration and air conditioning, etc. Electric drives are used to control the speed, torque and position of electrical motors (see Fig. 1), with real-time software that resides on heterogeneous safety-critical embedded systems controlling the power electronic circuits. Electric drives sit in the "control level" (see Fig. 2) and typically operate on the factory floor, working cooperatively with other devices to automate machinery. Also, data produced by each electric drive is a very critical asset because it carries vital information about the machinery it controls, so the factory owners are reluctant to expose this type of data over the Internet. In addition, data can be massive, often repetitive and not sensitive to delay. Sending it over the control network would eat capability, hence there is a need for each drive to be capable of data analytics locally. Our approach is to use a fog computing architecture (which we call *fogification*) in the implementation of electric drives, and show how such an architecture can be successfully used for the development of an industrial use case.

By developing the electric drives as fog nodes, new offerings like programmability, analytics and connectivity to customer Clouds are expected to increase the added value. Increased flexibility allows drives to assume a larger role in industrial and domestic control systems by benefiting from the ability to instrument as the data source which can help in bootstrapping the data economy. The main direct business benefit comes from the ability to instrument also legacy systems by using drives as the data source. Edge analytics will help in off-loading the network and extend the Internet-of-Things (IoT) solutions market. Digital services allow efficient service provisioning, improved uptime, and decreased overall costs. Correctly configured products and processes decrease energy consumption and improve quality. Open data ecosystems will allow anyone to innovate new value-added services and will create long term benefits for all ecosystem participants.

In the remainder of this paper, we take the current architecture of electric drives, tailor an FCP-based architecture

978-1-7281-8956-7/20/$31.00 ©2020 IEEE

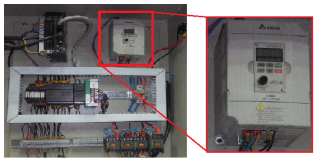


Fig. 1. An electric drive (in red) shown in an industrial setting.

for it and model them in Sect. II. We next identify the challenges related to the fogification of electric drives and collect the requirements that drive both architectures and also the "technology bricks" needed to implement the fogified electric drives in Sect. III. The proposed fog-based architecture is then used to model an industrial use case. We evaluate the resulting use case on several aspects, demonstrating the usefulness of the proposed fog-based platform in Sect. IV. We study the related work in Sect. V and conclude the paper in Sect. VI.

## II. Architecture

In this section, we first introduce electric drives and how they work in Sect. II-A. Then, we describe the current architecture of the drives in Sect. II-B and take it as baseline. We have derived a set of high-level requirements (presented in detail in Sect. III-A) that will drive the definition of the fog-based drives architecture in Sect. II-C. We compare the baseline and fogified architectures in Sect. II-D. We use the Architecture Analysis & Design Language (AADL) [8], which has the ability to model large-scale architectures from many aspects in a single analyzable model via its strong syntactic and semantic support for the description of both hardware and software systems. Hence, we use AADL to model the baseline and fogified architectures.

### A. Electric Drives

An electric motor is an electro-mechanical machine which converts electricity into mechanical rotary movement of its shaft. The mechanical rotary movement of the shaft is generated through the interaction of a magnetic field and an electric current which impacts the movement, i.e. rotation speed, rotation torque, and position of the shaft. Electric drives, alternatively called *drives*, are used to alter characteristics of the electric current such as frequency and voltage to control the motor speed, torque and position [9].

To automate and control machinery and industrial equipments, which comprise the first level of the automation pyramid (see Fig. 2) and sit on the "Machine level", a preliminary control device is used to determine the required output of the actuators: electric motors in our case. The electric drives, as secondary control equipment which are placed close to the actuators and industrial devices on the factory level, are connected to the preliminary control equipment, and control the actuators to generate the required output.

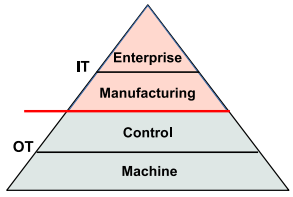



Fig. 2. Automation Pyramid

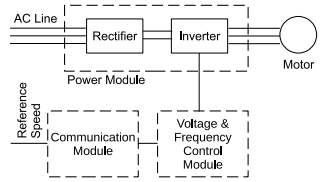



Fig. 3. Internals of an electric drive

The internals of an electric drive have a communication module, a control module, and a power module. The communication module receives the required motor output as a reference from the preliminary control equipment via industrial network. The control module runs a control application to drive the output to the given reference value via the power module which alters the frequency or voltage, based on the type of the drive, to generate the electric current that leads to the desired motor output. A drive is an embedded cyber-physical system that requires real-time response and reliability in order to meet degrees of quality in its output to be able to interact with other devices.

Electric drives are designed for either general purposes, i.e. to control certain power range motors, or specific purposes, i.e. to control a specific electric motor with specific requirements. The drives come in various types, indicated by their power modules, which use different electric current characterizations. Moreover, safety features such as a motor brake are embedded in the control module. Internals of an electric drive are depicted in Fig. 3.

### B. Baseline Architecture

In this section, we consider a VLT drive from Danfoss Power Electronics [10], and describe its current architecture, which we call the *baseline*. The hardware platform of the baseline has four modules: the communication module, the operation module, the control module and the power module. Each of the first three modules has a dedicated single-core processor and memory unit. The software stack for each module is dedicated. We model the baseline with AADL and show it in Fig. 4.

As depicted in the figure, the communication module has a network switch to connect through the Fieldbus interface with the ProfiNet/RT [11] standard. Its software stack has a real-time operating system which runs a time-triggered application with a limited cycle time that handles the network protocol. We assume that the drive communicates with a Programmable Logic Controller (PLC) as the preliminary controller to get the desired output of the motor, and a Human Machine Interface (HMI) to set the drive parameters such as communication and motor control configurations. The control module runs a

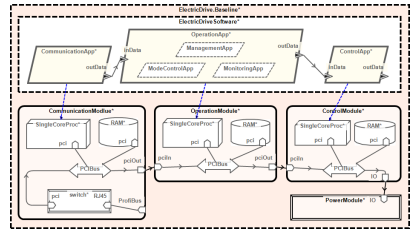



Fig. 4. AADL diagram of the baseline architecture.

feedback control application on the same real-time operating system, implemented according to the IEC61131-3 standard function blocks, once it is engaged. The control module has I/O links with the power module to get the feedback and set the control signal.

The software stack of the operation module has the same real-time operating system and runs applications which have priorities commensurate with their execution rates. The applications are: a mode application which engages and disengages the motor controller, a monitoring application which pre-analyzes the drive data, and a management application which configures the communication and controller parameters. The monitoring application collects data such as voltage and temperature and does local machine learning to predict the drive maintenance. The operation module shares separate dedicated buses with the communication module and the control module for data exchange.

### C. Fogified Architecture

We propose a fogified architecture for drives, based on extending the initial designs proposed in the Fog Computing for Robotics and Industrial Automation (FORA) European Training Network [12]. The proposed fogified drives architecture can be used both within the traditional hierarchical model of industrial automation (Fig 2) and in future distributed cyber-physical systems architectures (Fig. 5) that are envisioned to be used in Industry 4.0. In such a distributed architecture, the integration of computational and storage resources into the communication devices is realized in the fog node. In many applications, including industrial automation and robotics, several layers of fog nodes with differing computation, communication and storage capabilities will evolve, from powerful high-end fog nodes to low-end fog nodes with limited resources. Researchers have started to propose solutions for the implementation of fog nodes [13] and fog node solutions have started to be developed by companies [13]. Fog nodes could be connected to each other and to the machines through a deterministic communication solution, such as IEEE 802.1 Time-Sensitive Networking (TSN) [14], see Fig. 5. In TSN, time sensitive traffic is transmitted using schedule tables called "Gate Control Lists" (GCLs). Such an FCP-based architecture allows to increase the spatial distance between the physical process and the fog nodes that controls it, allowing the control functions to be executed remotely on the fog nodes. Several initiatives are currently working towards realizing this vision [12], [13].

We model the fogified architecture with AADL and show its schematic architecture in Fig. 6. The model consists of a hardware platform which has a dual-core processor, a switch and a power module, and a software stack which has a hypervisor, a middleware, an OS and an application layer.

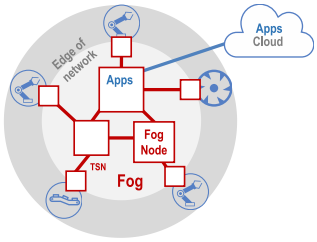



Fig. 5. Fog Computing platform. Boxes represent fog nodes, connected with each other and to the Cloud; the thick lines are the network. Applications (Apps) run on the fog and Cloud.

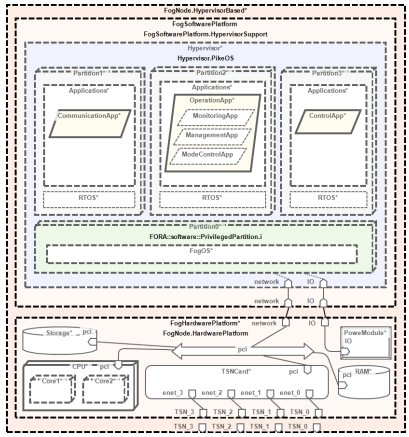



Fig. 6. AADL diagram of the fogified architecture

The mixed-criticality applications running on the dual-core processor are using temporal isolation enforced by the hypervisor, to prevent them from interference. The hardware platform has a TSN-enabled network switch and a power module which generates electric current according to the given control signal to drive motors, all connected through a shared bus. As depicted in the figure, the module has I/O with the processor which enables getting feedback and setting control signals.

We assume using PikeOS [15] as the hypervisor which implements temporal partitioning to isolate the applications. In our model, we have three partitions, indicated by the applications assigned to them: control, communication and operation. The hypervisor schedules the execution of partitions. We also consider using a middleware on top of the hypervisor to enable data exchange between the partitions and provide features like runtime updates.

The control partition has a soft-PLC OS, and the control application is implemented using the IEC61131-3 standard function blocks. The control application is configurable via the middleware. The communication partition has a real-time OS which runs applications for controlling the network traffic, applying security mechanisms, handling the applications traffic, and setting the message schedule tables (GCLs), all via the middleware. We assume that the operation partition has an OS to run different type of applications including a machine learning application for predictive maintenance (same as the predictive maintenance application in the baseline).

### D. Comparison

The most significant difference between the baseline and the fogified architecture, is the change in isolation mechanism from spatial to temporal. The multicore processor and the shared resources such as bus and memory increase the unpredictable delays and overheads in the execution of tasks, as well as the data exchange of applications. Although the fogified architecture brings more interference and unpredictability, it provides a programmable platform for monitoring purposes.

In the fogified architecture, we proposed using a hypervisor to enforce temporal isolation with the added cost of over-

TABLE I
SYSTEM LEVEL REQUIREMENTS

| # | Requirement | Realization in the baseline architecture | Realization in the fogified architecture |
|---|---|---|---|
| 1 | Drives shall be designed according to the industrial standards | IEC61800-based design | IEC61800-based design |
| 2 | Drives shall have time–constraint interface | 1 ms time–constraint ProfiNet interface | Jitter-free TSN interface |
| 3 | Drives shall be able to monitor and process data for predictive maintenance purpose | Machine Learning framework with appropriate Safety Integrity Level | Machine Learning framework with appropriate Safety Integrity Level |
| 4 | Drives shall run mixed-criticality applications according to the industrial standards | Spatial separation according to IEC61508 | Temporal separation according to IEC61508 |
| 5 | Drives shall control the electric motor accurately | Motor control with response time of 30 ms and good quality-of-control | Motor control with response time of 20 ms and good quality-of-control |
| 6 | Drives shall be configurable according to the industrial standards | Configurable according to IEC61508 | Configurable according to IEC61131 |
| 7 | Drives shall have secure access to the Cloud | Cloud connection provided by external devices | Cloud connection provided by TSN interface with security mechanisms |

head to the computation. The best effort partitions provide a programmable platform to collect data from different sensors, perform sensor fusion already in the drive as edge node, run simple machine learning algorithms in the device, and finally stream data over the interface.

The other significant difference between the two architectures lies in the use of TSN instead of ProfiNet/RT which helps in Cloud connectivity and IT/OT convergence. We also consider applying selected security mechanisms to protect against possible cyber-attacks. We see that the technology bottlenecks are in the hypervisor and TSN where applications and traffic should be scheduled, isolated and protected concerning industrial grade standards.

## III. CHALLENGES

In this section, we identify the system level requirements that drive the baseline and fogified architectures in Sect. III-A. We discuss the challenges inherent in the design and development of fogified drives and propose the necessary technology bricks in Sect. III-B.

### A. Requirements

We identify the system level requirements and the relevant realization in Tab. I, where the requirements are shown in column 2, the baseline realization in column 3 and the fogified realization in column 4. Unlike special-purpose drives which have to satisfy certain requirements that are needed for their specific purpose, the general-purpose drives which are considered in this paper have to satisfy more generic requirements that aim to make the drive compatible with a wide-range of electric motors.

For each of the requirements, we sort the relevant architecture components by the order of flexibility, take the least-flexible one as a pivot point, apply the requirement on the component to achieve the best possible performance, refine the requirement concerning the achieved performance, and apply the refined requirement to the next component. The system level requirements are qualitative at the beginning, and become quantitative after several iterations of applying them to the components, since each requirement aims to achieve the best performance. In industry, the strongest requirement is generally the financial aspect which mostly covers the costs, and prevents achieving the best performance.

Requirement 1 implies that the drive design should comply with the industrial standards, which is met by both architectures in the same way via IEC61800 that states control strategies and performance requirements of the motor controller and converter in different operation conditions. Industrial time constraint communication is considered in requirement 2, which is realized by ProfiNet in the baseline, and TSN in the fogified architecture. We consider a machine learning approach for drive maintenance prediction as a service in requirement 3. The service is implemented as an application with lower criticality in both architectures and addressed with different isolation approaches with respect to IEC61508, as imposed by requirement 4. Requirement 5 imposes accuracy (control performance) constraints on the control application along with performance constraints (imposed by requirement 1) and integrity constraints (imposed by requirement 4 where the control application has the highest priority and highest Safety Integrity Level). The configurability of the applications for performance and operation is addressed in requirement 6, which needs communication between applications that is realized in different ways in the baseline and fogified architectures. Secure access to the Cloud (imposed by requirement 7) is applied by using external equipment such as a gateway that has cloud connection in the baseline architecture, and by using security mechanisms on the TSN interface in the fogified architecture.

### B. Technology Bricks

The fogified architecture proposed in Sect. II-C has been driven by and meets the requirements from Sect. III-A. To implement such an architecture, we identify in this section the needed "technology bricks", which can be methods, models, hardware, software, tools, mechanisms, etc. We have identified the following technology bricks:

The **FCP Configuration** provides a configuration for temporal separation and scheduling of mixed-criticality tasks. The configuration also considers the extra functional requirements of the tasks such as the quality-of-control (QoC) for control applications (see [16] for more details). The configuration consists of partition tables, the task schedule tables inside each partition and GCLs of TSN switches; which addresses requirements 2, 4, 5 and 6. A **Machine Learning Framework** brings the capability to predict when the drive maintenance is needed, by accessing the drive data via middleware. We propose a decentralized framework concerning the TSN capability of the fogified architecture which leads to more accurate prediction. We also propose a fault detection, isolation, and recovery (**FDIR**) method to be applied on TSN communication and tasks execution. The method is implemented on the middleware and monitors tasks execution and network communication traffic. It applies detection and identification techniques, and provides recovery mechanisms in case of faults. Furthermore, we propose to deploy various **Security Mechanisms** which protect TSN from cyber-attacks and unauthorized access to the node. The mechanisms are implemented on the hypervisor where low-level access to hardware is possible.

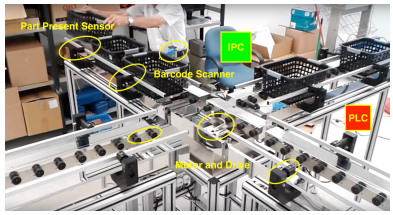



Fig. 7. Conveyor Belt Use Case

## IV. Evaluation

We have used the proposed fogified architecture to model a **Conveyor Belt** Use Case (UC): a conveyor belt is used to distribute packages from an inventory to different destinations based on the package. The conveyor belt is well-known and widely used in inventories for the automatic distribution of packages, and is realized using electric motors and drives. In the UC we consider a typical machine as depicted in Fig. 7. The machine is fed with packages from one side and reads the tag of the received package. It gets the destination of the package by accessing a database with the read tag; and drives the package towards the destination from one of the other sides of the machine. The UC has been realized using the proposed fogified architecture from Sect. II-C and the technology bricks from Sect. III-B. In this section we evaluate the resulting implementation on several aspects to address the requirements (showing the suitability of the implementation for industrial applications) and exhibit the new offerings such as programmability, analytics and connectivity to customer Clouds (showing the added value of the implementation for industrial applications) as shown in the subsections.

### A. Network configuration for QoC

Since the fogified architecture shares the same communication medium for hard and soft real-time, non-critical and best effort communication, we propose the scheduling of the traffic on TSN to guarantee timing requirements of the streams. We assume that the communication between drives and other industrial equipment is achieved via TSN and also the control applications are a set of streams which have hard real-time requirements. All messages are scheduled using the schedule-based time-sensitive traffic type in TSN, which, as mentioned, uses schedule tables in the switches (GCLs) to schedule the transmission of messages. The streams are prioritized in accordance with their criticality and scheduled with respect to their requirements. Recent works that address scheduling of TSN traffic can be found in [17].

We employ the constraint programming-based schedule synthesis strategy aiming at maximizing the QoC and satisfying the deadlines of real-time messages, proposed in [18] to schedule the traffic.

TABLE II
Streams in the Conveyor Belt Use Case

| # | Size (bytes) | Period ($\mu$s) | Routing | Max. delay ($\mu$s) |
|---|---|---|---|---|
| 1 | 500 | 3000 | $C_1 \rightarrow N_4 \rightarrow A_1$ | 4 |
| 2 | 100 | 1000 | $C_2 \rightarrow N_5 \rightarrow N_2 \rightarrow N_4 \rightarrow A_1$ | 8 |
| 3 | 150 | 3000 | $C_1 \rightarrow N_4 \rightarrow N_2 \rightarrow A_2$ | 4 |
| 4 | 250 | 4000 | $C_2 \rightarrow N_5 \rightarrow A_3$ | 2 |
| 5 | 1200 | 10000 | $A_2 \rightarrow N_2 \rightarrow N_5 \rightarrow C_2$ | 57 |
| 6 | 300 | 4000 | $A_1 \rightarrow N_8 \rightarrow N_6 \rightarrow C_1$ | 3 |
| 7 | 400 | 3000 | $S_1 \rightarrow N_1 \rightarrow N_3 \rightarrow C_1$ | 20 |
| 8 | 400 | 6000 | $S_2 \rightarrow N_1 \rightarrow N_3 \rightarrow C_1$ | 20 |
| 9 | 1500 | 15000 | $S_3 \rightarrow N_2 \rightarrow N_5 \rightarrow C_2$ | 59 |

TABLE III
Tasks in the Conveyor Belt Use Case

| Applications | Tasks | WCET ($\mu$s) | P (ms) |
|---|---|---|---|
| $\gamma_1$ | $\tau_1$ | 500 | 10 |
| | $\tau_2$ | 2500 | 10 |
| | $\tau_3$ | 2500 | 10 |
| $\gamma_2$ | $\tau_4$ | 1500 | 12 |
| | $\tau_5$ | 3000 | 12 |
| $\gamma_3$ | $\tau_6$ | 15000 | 50 |
| $\gamma_4$ | $\tau_7$ | 2000 | 20 |
| | $\tau_8$ | 3000 | 20 |
| | $\tau_9$ | 1500 | 20 |
| | $\tau_{10}$ | 1500 | 20 |

We have evaluated our proposed network configuration strategy on our Conveyor Belt UC. In the given example, the five switches (denoted with $N$) connect three sensors (denoted with $S$) to two controllers (denoted with $C$), and transmit the messages from controllers to four actuators (denoted with $A$). The details of the streams are shown in Tab. II where the stream name, size, period and routing are shown in columns 1 to 4 respectively. We assume that one of the controllers runs a motor control application for speed control of electric motors. The motor speed controller running on the node $C_1$ receives message 7 which is the sensor data $S_1$, and sends message 1 to our proposed fogified drive $A_1$, which controls an electric motor. We assume that all links have 1 Gbps bandwidth.

The proposed strategy has successfully scheduled all streams, i.e., none of the deadlines are missed, and minimized delay and jitter of streams, resulting in an optimized control performance. The results show that all streams have zero jitter, which improves control. The column 5 in Tab. II shows the maximum delay of streams. We used JitterTime [19] to simulate the behavior of the control application which reports a value of 0.008 for the QoC (see [18] for the exact cost function), i.e., a good control performance.

### B. Configuration of hypervisor partitions and task schedules

Since the real-time applications are virtualized and implemented as tasks, the FCP configuration (e.g., task scheduling) has impact on the performance of control applications. We assume the use of deterministic hypervisors for virtualization of applications on the fog node similar to [20], where hypervisors provide a deterministic access to shared resources via a static configuration table and provide spatial and temporal isolation of mixed-criticality applications via "partitioning". We propose a metaheuristic solution to optimize the hypervisor partition tables, map the tasks to the processing cores of the multi-core processors of fog nodes, assign the tasks to partitions and schedule the tasks inside the partition tables.

Our proposed solution provides temporal separation of tasks similar to [21], [22] and assignment of the tasks to the cores, and scheduling of the tasks inside the partition slices, similar to the optimization strategy presented in [16] where the static scheduling of tasks considers the QoC of control applications. We have evaluated our solution, while ignoring the temporal isolation of tasks, on the UC in which four applications, including a control application denoted with $\gamma_1$, are running on a fog node that has two cores. Each application has number of tasks and each task has a worst-case execution time (WCET) and a period (P). The control application is the drive's controller for controlling electric motors. The details of the applications are shown in Tab. III.

Our proposed optimization strategy has successfully scheduled all the tasks and decided the task mapping to the cores. The results show that none of the tasks has missed its deadline. Furthermore, the control application has a good

control performance which is evaluated with JitterTime [19] that calculates a value of 0.011 for the QoC (cf. the cost function from [16]).

*C. Addressing security mechanisms in TSN*

In order to adequately protect the system against adversaries, security mechanisms are required. A compromised system may lead to safety requirements being violated, meaning that security services must run with at least the same priority as critical tasks. We briefly discuss security solutions that are enabled by fogification of the drive in our UC. These should all be deployed in parallel, as an instance of a defense-in-depth approach. The various mechanisms proposed here are summarized in Table IV.

Firstly, the drive will communicate sensitive data over the Internet, such as usage statistics for predictive maintenance. To ensure confidentiality and integrity of the data, as well as authenticity of the remote party, secure communication standards such as TLS should be used to provide confidential authenticated communication channels. The FCP should block any attempt at communication to endpoints it cannot authenticate. To further limit the attack surface, a firewall should be active on the hypervisor level, ideally making use of the predictable nature of machine-to-machine communication by using whitelists for known addresses and services. These measures protect against attacks known as the Man-in-the-Middle (MitM) attacks, where an adversary sits between the legitimate sender and receiver, capable of snooping on and/or modifying data in transit.

Services that communicate with the Internet (including Cloud service) form a major attack vector of Internet-connected devices, and should be placed in separate partitions by the FCP, isolating them from other services. This makes it more difficult for attackers to pivot to other parts of the system, should they break into an Internet-facing service, thus limiting the impact of an intrusion.

Additionally, a security monitoring service should run in a separate highly-privileged partition, capable of detecting anomalous behavior in the system, while also improving forensic possibilities. These are important factors in a fast detection of system intrusion, impact analysis, and attacker attribution.

Because of the time-critical nature of TSN, the protocol itself provides only minimal security, it is necessary to isolate this as much as possible from the rest of the system. In this UC, the architecture is perfectly positioned to isolate all TSN traffic from the rest of the physical network using Software Defined Networks (SDN), or similar techniques. For further protection within TSN, per-stream filtering as described in the IEEE 802.10qw [14] standard can be applied as a light-weight monitoring technique. In order to mitigate Denial of Service (DoS) attacks as much as possible, careful consideration to the network topology should be given during the design phase, so that if a single link in the network were to fail, traffic can be routed over different paths.

Finally, within industrial networks, physical access to machines is a relatively common attack vector, therefore, configuration changes of the electric drive should not be possible without some form of authentication of the operator, such as a secure hardware element.

*D. Distributed predictive maintenance in the fog*

Here we propose a distributed Machine Learning framework where the distributed drives and the centralized server jointly (collaboratively) train one global model. Typically,

TABLE IV
THREATS AND THEIR MITIGATIONS

| Threat | Mitigations |
|---|---|
| MitM, impersonation | Confidential, authenticated com. channels |
| Attack impact | Service isolation (e.g. partitions) |
| Remote attacks | Firewalls, endpoint whitelisting |
| DoS | Redundant network topologies |
| TSN security | Isolation of TSN protocol, per-stream filtering |
| Physical attacks | Hardware token for configuration changes |
| Detection | Security monitoring service |

the distributed drives placed in different locations, generate data that captures the local information instead of global information and they train their local models based on the partial knowledge. The aggregation step at the server-side enables the information sharing between drives and server to obtain one model with overall knowledge. Consecutively, the server sends the aggregated model back to drives. The whole procedure may iterate several times. In a nutshell, as depicted in Fig. 8, it is divided into four steps: (i) local model training, (ii) model (or gradient) transmission, (iii) aggregation and (iv) sending aggregated model back to decentralized devices.

We applied this method combing with active learning [23] in the work [24], which experimentally proves its effectiveness. This collaborative learning scheme mainly has two virtues. Firstly, it may save the cost of bandwidth by avoiding the transmission of the massive training dataset (transmit model parameter or gradient instead). Secondly, it may preserve user privacy by keeping data in the generation place, and at the same time train a model that has comprehensive knowledge. Yet, recently some researchers argue the gradient may breach the privacy by reverse engineering work, but it can be defended by plugging noises to the gradient before the transmission, e.g., noises generated from Laplace distribution. As we mentioned before, data produced by electric drives is a very critical asset because it carries precious information about the machinery it controls, and we believe it can be addressed by the proposed collaborative model.

We experimented on a public simulated engine run-to-failure events dataset [25] to demonstrate our method, since a public drive failure dataset was not available. We assume four edge devices and one fog node in the experiment. The dataset is composed of 24 features and the binary labels where zero represents failure within one preset period (30 days), one otherwise. For the sake of a more elaborated result, we can chunk time-to-failure into more periods to convert it to a multiple-classification problem. For instance, label zero indicates time-to-failure less than two weeks, label one indicates the period between 2 weeks and one month, etc. We employ Logistic Regression [26] as the model to carry out the binary classification task. The one-shot binary classifier result is depicted in Fig. 9, where the accuracy of the devices and the aggregated model is shown. The aggregation step improves the overall performance and the accuracy of the devices. Note

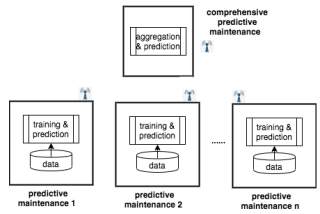



Fig. 8. Distributed ML Diagram

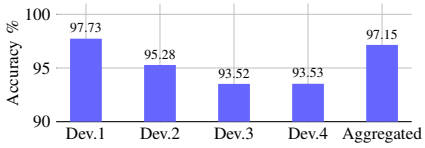



Fig. 9. One-shot Predictive Maintenance Performance

that here we only demonstrate the one-shot result, but it can be repeated multiple times according to the requirement.

*E. Fault detection, identification and recovery for the UC*

In this section, first, we list the safety requirements of the IEC61508 standard, and then we propose a way to provide safety assurance for the safety functions in the proposed UC. The primary safety requirement for electric drives, whether fogified or not, is to shutdown and stop the motor in case of emergency situations such as voltage/current surges, which can result in uncontrolled motor speeds, short-circuits and possible danger to human life. Additionally, we present below a set of safety requirements that needs to be included in the design of the fogified electric drives. The identification of failure modes for fogified electric drives shall be documented at design time. A safe timer is a common practice since functions related to drives require safe timers. The safety of the drive is directly relying on the fact that the timer is correct. Deadlock prevention among safety and non-safety functions in the drives is one of the critical requirements since a deadlock brings non-deterministic behaviors and is the major cause of deadline misses for safety tasks. All emergency and process shutdown functions (ESD, PSD) need to be executed regardless of concurrent processes running in the FCP. Their criticality must always be the highest. A failure event should have a persistence parameter which specifies the duration of the failure detection before a failure is declared. The parameter should specify either a time duration or a repeated detection threshold. The persistence time may be zero so that any detection is immediately treated as a failure. Adequate redundancy measures for safety functions are a standard practice as well as temporal and spatial isolation of safety and non-safety functions. The system shall have a defined behavior on detection of a fault or a failure event. This may be either a safe state or a well-defined consequence or behavior. Last but not least, any run-time changes (such as over-the-air firmware updates) in the fogified drives should only be done to the non-critical parts or should undergo a validation for safety.

To provide safety function guarantees, we have turned the safety instructions into drive operational states by introducing a marginal behavior. Therefore, the values of the operational state of the drive are decided based on the satisfactory behavior as a tool. We propose to deploy the tool on the middleware of the architecture, which has access to all the partitions that have applications with different criticality assigned to them.

We take the Conveyor Belt UC as an example to define several operational states for the drive and also proportional safety actions to take. An unsafe safety state is one wherein the safety state variables breach their threshold values. The objective is that we have no overlapping conditions and the drive always falls into one of the conditions which has a safety action to take. For example, as shown in Tab. V, a safety critical operational state such as an emergency brake failure will trigger a safety action like an emergency shutdown that is executed in a separate system partition.

TABLE V
OPERATIONAL STATES AND SAFETY ACTIONS

| **Operational state** | **Safety action** |
|---|---|
| Switch failure | Re-route stream on alternate switch |
| Part presence sensor failure | Stop conveyor belt if item is fragile |
| Emergency brake failure | Trigger emergency shutdown function |
| Motor over-heating | Inject coolant |
| Controller malfunction | Switch to safety controller (redundant) |

## V. Related work

Several research projects have addressed mixed-criticality applications on that share multicore-based distributed architectures. The EMC2 European project[1], aims to provide efficient handling of mixed criticality applications under real-time conditions, scalability and utmost flexibility, full scale deployment and management of integrated tool chains, through the entire life cycle. Research on Fog computing platform architectures has made progress in recent years [13], [27]. For example, the European projects FORA[2] and mF2C[3] focus on creating open source, standards-compliant fog platforms using COTS hardware to execute hard real-time industrial control applications such as the electric drives discussed in this paper. Companies such as TTTech Computertechnik AG and Nebbiolo Technologies, Inc. are pioneers in the field of commercializing the Fog computing paradigm with market ready products for industrial automation. While design paradigms for the fog are still in their early stages, there are certain generic guidelines that are followed to ensure isolation of tasks of varying criticality. In [28] the authors describe an execution framework wherein applications are isolated temporally on many-core processors.

Safety certification as proof of guarantees for the proper execution of safety functions is needed for the FCP. Classical safety controller design such as the simplex architecture [29], [30] provide a switching mechanism between a high performance but non-safety certified controller and a simple certified controller for safety functions. However, for complex systems such as the FCP, the simplex design is non-optimal due to the switching latencies. Selicean et al. [21] propose a method in which different Safety-Integrity Levels (SILs) are assigned to the applications. In this method applications with the same SIL are mapped to a single partition. Virtualization of control applications can be realized through separation and scheduling the control tasks inside the partitions similar to [22]. The modification of hypervisors provides different degrees of separation. Modification of the Xen hypervisor to guarantee timing constraints are proposed by Masrur et al. [31]. The authors modify the hypervisor with a new scheduler based on a fixed-priority policy and a control loop to control timing constraints of virtual machines. [32] addresses safety critical applications running in the Fog and how the FCP must cater to these specific requirements.

One of the major research themes is resource management in the Fog. In [33], the authors identify and classify the architectures, infrastructure, and underlying algorithms for managing resources in fog/edge computing. [34] proposes a list scheduling-based heuristic to solve this problem. The authors demonstrate the feasibility of reconfiguring the scheduled network at runtime for industrial applications within the fog. [35] introduces a vulnerability-based method to quantify

[1]www.artemis-emc2.eu
[2]www.fora-etn.eu
[3]https://www.mf2c-project.eu

the security performance of communications on distributed systems. Fault tolerant aspects are discussed in [36] where the design problem is to minimize the schedule length and security vulnerability of the application, subject to given fault-tolerant constraints. A multi-objective optimization method to find the best solutions is then proposed. [37] discuss potentially contradicting design constraints: real-time capability versus scalability. This paper suggests a design methodology and architecture as a step towards perfectly scalable real-time systems, i.e. systems with deterministic timing behavior and run-time reconfiguration.

## VI. Conclusions

In this paper, we have addressed a novel fog-based architecture that is a key enabling technology for Industry 4.0. We have proposed to re-engineer electric drives and turn them into fog nodes. We take the current drives architecture as baseline, apply our proposed fog computing architecture to it, and compare the two architectures. The proposed architecture is driven by the stringent safety and performance requirements of industrial applications. In addition, we have identified fog-specific requirements and challenges.

We have modeled our proposed architecture with AADL and studied the interaction of the components, identified the needed "technology bricks" and bottlenecks, and mapped the proposed architecture to a computing platform for the realization of an industrial use case, a Conveyor Belt application. We have evaluated the use case in relation to the proposed technology bricks. As the evaluation shows, a fog-based implementation of industrial applications is a promising approach to realize the vision of Industry 4.0.

## Acknowledgements

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 764785, FORA—Fog Computing for Robotics and Industrial Automation.

## References

[1] World Economic Forum, "Digital Transformation of Industries," http://reports.weforum.org/digital-transformation/wp-content/blogs.dir/94/mp/files/pages/files/wef-digital-transformation-2016-exec-summary.pdf, 2016 (accessed March 15, 2020).

[2] D. Floyer, "Defining and sizing the industrial internet," http://wikibon.org/wiki/v/Defining_and_Sizing_the_Industrial_Internet, 2013 (accessed March 15, 2020).

[3] H. Bauer, C. Baur, D. Mohr, A. Tschiesner, T. Weskamp, K. Alicke, and D. Wee, "Industry 4.0 after the initial hype–where manufacturers are finding value and how they can best capture it," *McKinsey Digital*, 2016.

[4] M. García-Valls, T. Cucinotta, and C. Lu, "Challenges in real-time virtualization and predictable cloud computing," *Journal of Systems Architecture*, vol. 60, no. 9, pp. 726–740, 2014.

[5] D. R. Harp and B. Gregory-Brown, "IT/OT convergence bridging the divide," *NEX DEFENSE*, 2014.

[6] W. Steiner and S. Poledna, "Fog computing as enabler for the Industrial Internet of Things," *e & i Elektrotechnik und Informationstechnik*, vol. 133, no. 7, pp. 310–314, 2016.

[7] OpenFog Consortium, "OpenFog reference architecture for fog computing," https://www.iiconsortium.org/pdf/OpenFog_Reference_Architecture_2_09_17.pdf, 2017 (accessed January 5, 2020).

[8] P. H. Feiler, D. P. Gluch, and J. J. Hudak, "The architecture analysis & design language (AADL): An introduction," Carnegie-Mellon Univ Pittsburgh PA Software Engineering Inst, Tech. Rep. CMU/SEI-2006-TN-011, 2006.

[9] I. Boldea and S. A. Nasar, *Electric drives*. CRC press, 2016.

[10] Danfoss, "Danfoss Electric Drives," https://www.danfoss.com/en/products/ac-drives/?sort=default_sort, 2020 (accessed March 15, 2020).

[11] Siemens Simatic, "Profinet system description–system manual," *Issue A5E00298288-04*, vol. 6, 2008.

[12] Fog Computing for Robotics and Industrial Automation (FORA), "Fog Computing Platform: requirements and initial designs," https://drive.google.com/file/d/1QwBfcqij72ZdeMWmhwAwm_MdSHePElUy/view, 2019 (accessed March 25, 2020).

[13] C. Puliafito, E. Mingozzi, F. Longo, A. Puliafito, and O. Rana, "Fog computing for the Internet of Things: A Survey," *ACM Transactions on Internet Technology (TOIT)*, vol. 19, no. 2, pp. 1–41, 2019.

[14] IEEE, "Official Website of the 802.1 Time-Sensitive Networking Task Group," http://www.ieee802.org/1/pages/tsn.html, 2016 (accessed March 5, 2020).

[15] R. Kaiser and S. Wagner, "The PikeOS concept: History and design," *SysGO AG White Paper. Available: http://www.sysgo.com*, 2007.

[16] M. Barzegaran, A. Cervin, and P. Pop, "Towards Quality-of-Control-Aware Scheduling of Industrial Applications on Fog Computing Platforms," in *Proceedings of the Workshop on Fog Computing and the IoT*. ACM, 2019, pp. 1–5.

[17] S. S. Craciunas, R. S. Oliver, M. Chmelík, and W. Steiner, "Scheduling real-time communication in IEEE 802.1 Qbv time sensitive networks," in *Proc. of the International Conference on Real-Time Networks and Systems*, 2016, pp. 183–192.

[18] M. Barzegaran, B. Zarrin, and P. Pop, "Quality-Of-Control-Aware Scheduling of Communication in TSN-Based Fog Computing Platforms Using Constraint Programming," in *2nd Workshop on Fog Computing and the IoT*, vol. 80. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020, pp. 3:1–3:9.

[19] A. Cervin, P. Pazzaglia, M. Barzegaran, and R. Mahfouzi, "Using JitterTime to Analyze Transient Performance in Adaptive and Reconfigurable Control Systems," in *Proc. of IEEE International Conference on Emerging Technologies and Factory Automation*. IEEE, 2019, pp. 1025–1032.

[20] J. Ruh and W. Steiner, "The need for deterministic virtualization in the Industrial Internet of Things," in *Proc. of the Workshop on Fog Computing and the IoT*. ACM, 2019, pp. 26–30.

[21] D. Tamas-Selicean and P. Pop, "Design optimization of mixed-criticality real-time systems," *ACM Transaction on Embedded Computing*, vol. 14, no. 3, pp. 50–78, May 2015.

[22] M. Barzegaran, A. Cervin, and P. Pop, "Performance Optimization of Control Applications on Fog Computing Platforms Using Scheduling and Isolation," *IEEE Access*, vol. 8, pp. 104 085–104 098, 2020.

[23] Y. Gal, R. Islam, and Z. Ghahramani, "Deep bayesian active learning with image data," in *Proc. of the International Conference on Machine Learning*. JMLR. org, 2017, pp. 1183–1192.

[24] J. Qian, S. Sengupta, and L. K. Hansen, "Active learning solution on distributed edge computing," *arXiv preprint arXiv:1906.10718*, 2019.

[25] A. Saxena and K. Goebel, "Turbofan engine degradation simulation data set. NASA Ames Prognostics Data repository, NASA Ames Research Center, Moffett Field," 2008.

[26] D. G. Kleinbaum, K. Dietz, M. Gail, M. Klein, and M. Klein, *Logistic regression*. Springer, 2002.

[27] S. Yi, Z. Hao, Z. Qin, and Q. Li, "Fog Computing: Platform and Applications," in *Proc. of IEEE Workshop on Hot Topics in Web Systems and Technologies*, 2015, pp. 73–78.

[28] Q. Perret, P. Maurere, E. Noulard, C. Pagetti, P. Sainrat, and B. Triquet, "Temporal Isolation of Hard Real-Time Applications on Many-Core Processors," in *Proc. of IEEE Real-Time and Embedded Technology and Applications Symposium*, 2016, pp. 1–11.

[29] S. Bak, D. K. Chivukula, O. Adekunle, M. Sun, M. Caccamo, and L. Sha, "The System-Level Simplex Architecture for Improved Real-Time Embedded System Safety," in *Proc. of IEEE Real-Time and Embedded Technology and Applications Symposium*, 2009, pp. 99–107.

[30] Lui Sha, "Using simplicity to control complexity," *IEEE Software*, vol. 18, no. 4, pp. 20–28, 2001.

[31] A. Masrur, S. Drossler, T. Pfeuffer, and S. Chakraborty, "VM-Based Real-Time Services for Automotive Control Applications," in *IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, Aug 2010, pp. 218–223.

[32] N. Desai and S. Punnekkat, "Safety of Fog-Based Industrial Automation Systems," in *Proc. of the Workshop on Fog Computing and the IoT*. ACM, 2019, p. 6–10.

[33] C.-H. Hong and B. Varghese, "Resource Management in Fog/Edge Computing: A Survey on Architectures, Infrastructure, and Algorithms," *ACM Computing Surveys*, vol. 52, no. 5, 2019.

[34] P. Pop, M. L. Raagaard, M. Gutierrez, and W. Steiner, "Enabling Fog Computing for Industrial Automation Through Time-Sensitive Networking (TSN)," *IEEE Communications Standards Magazine*, vol. 2, no. 2, pp. 55–61, 2018.

[35] W. Jiang, P. Pop, and K. Jiang, "Design Optimization for Security- and Safety-Critical Distributed Real-Time Applications," *Microprocessors and Microsystems*, vol. 52, no. C, p. 401–415, 2017.

[36] W. Jiang, H. Hu, J. Zhan, and K. Jiang, "Work-in-Progress: Design of Security-Critical Distributed Real-Time Applications with Fault-Tolerant Constraint," in *Proc. of International Conference on Embedded Software*, 2018, pp. 1–2.

[37] P. Priller, W. Gruber, N. Olberding, and D. Peinsipp, "Towards perfectly scalable real-time systems," in *Proc. of International Conference on Computer Safety, Reliability, and Security*. Springer, 2014, pp. 212–223.

Paper E

# rTLS: Lightweight TLS Session Resumption for Constrained IoT Devices

K. Tange, D. Howard, T. Shanahan, S. Pepe, X. Fafoutis, and N. Dragoni. "rTLS: Lightweight TLS Session Resumption for Constrained IoT Devices." English. In: *Proceedings of the 22nd International Conference on Information and Communications Security*. Springer, 2020. DOI: 10.1007/978-3-030-61078-4_14

# rTLS: Lightweight TLS Session Resumption for Constrained IoT Devices

Koen Tange[1], David Howard[2], Travis Shanahan[2], Stefano Pepe[3], Xenofon Fafoutis[1], and Nicola Dragoni[1,4]

[1] DTU Compute, Technical University of Denmark, {kpta,xefa,ndra}@dtu.dk
[2] Itron Idea Labs, USA
[3] UniquID, USA
[4] AASS, Örebro University

**Abstract.** The Transport Layer Security (TLS) 1.3 protocol supports a fast zero round-trip time (0-RTT) session resumption mechanism, enabling clients to send data in their first flight of messages. This protocol has been designed with Web infrastructure in mind, and requires these first messages to not change any state on the server side, as it is susceptible to replay attacks. This is disastrous for common IoT scenarios, where sensors often transmit state-changing data to servers. As bandwidth is a huge concern in the IoT, the field stands to benefit significantly from an efficient session resumption protocol that does not suffer from these limitations. Building on the observation that in IoT scenarios the set of clients is often bounded and fairly static, we propose rTLS (ratchet TLS), an efficient 0-RTT session resumption protocol that dramatically decreases bandwidth overhead, while adding forward secrecy and break-in resilience, and is not susceptible against replay attacks.

**Keywords:** Network · Security · IoT · IIoT · TLS · Protocol

## 1 Introduction

There are many examples of well-established communication protocols that are able to satisfy contextually-defined requirements and are in use in modern technology. Arguably the most well-known example is the TLS protocol [16]. This protocol is widely used in today's Internet, with Web security as its main focus. Recently, this protocol has been gaining traction in the Internet of Things (IoT) domain as well. To better suit the heterogeneous needs present in this domain, new extensions of the TLS protocol are needed, specifically to enable extremely lightweight devices to partake in TLS connections as well.

A typical TLS handshake can require anywhere between 1 and 4 KB of traffic. This is a large amount of traffic overhead for lightweight devices running on battery power, where powering a wireless radio is very costly. Therefore, there is a need to reduce this handshake overhead as much as possible. To aid in reducing bandwidth and latency, TLS 1.3 features a new session resumption protocol capable of transmitting application data already in the first flight of

messages. This allows users to quickly reopen a session without having to go through the expensive handshake again. Unfortunately, this resumption protocol is only marginally useful for IoT applications, as it does not allow for data that might change server-sided state, as a result of its weakness against replay attacks.

A second motivator for reducing traffic overhead is that the financial costs of sending this data might become unbearable. For example, it is expected that with 5G Low-Power Wide Area Networks (LPWAN) services such as Long Term Evolution - Machine (communication) (LTE-M) and Narrow Band Internet of Things (NB-IoT), network providers will charge users based on data usage [2, 21, 9]. Moreover, if the cost of setting up a secure connection is tens of times the cost of the payload itself, users might opt not to secure it at all, or implement their own cryptographic protocol, with associated risks.

In a standard TLS setup, servers are not likely to keep state on a client in between sessions, and the protocol is designed with that assumption in mind. In an IoT setting, however, the set of clients is fairly static, and often even known a priori, or traceable through some key infrastructure. Keeping state on these clients between connections can help in reducing the handshake overhead, but this is not yet utilized in TLS 1.3. There is thus a pressing need for IoT-focused TLS extensions that enable secure yet efficient communication with lightweight devices.

In this work, we introduce rTLS, a TLS extension that can authenticate two endpoints and set up a secure connection with minimal additional overhead, given that the client and server have initiated a session in the past. In particular, we introduce an extension to TLS 1.3 that changes the 0-RTT session resumption protocol, reducing overhead compared to the standard protocol, while adding new security features including replay protection, forward secrecy, and break-in protection. We build the protocol on the assumption that servers can store state on clients, with the IoT in mind. We provide equations on the lower bound for traffic overhead of any TLS resumption protocol as well as our proposed extension, and compare it to overhead observed from the OpenSSL [13] implementation of TLS 1.3. We also provide estimations for storage overhead for both client and server.

The remainder of this paper is organized as follows: In Section 2 we briefly discuss the foundations necessary to understand our proposed extension. In Section 5 we discuss related work on lightweight protocols and other TLS extensions. Then, in Section 3 we explain our extension in detail. After that, we evaluate the storage and transmission overhead as well as the security properties in Section 4, after which we conclude this work in Section 6

## 2 Preliminaries

This work proposes an improvement of session resumption for the TLS 1.3 protocol, building on Key Derivation Function (KDF)-chains, described in the double ratchet protocol description in the Signal documentation [14]. In this section, we briefly discuss the essentials needed to understand our proposed solution.

### 2.1 TLS 1.3

The TLS 1.3 protocol [16] negotiates a secure communication channel (a session) between two parties, typically referred to as client and server. In the most typical scenario, one-way authentication is provided, that is, the server authenticates itself to the client, building on the certificate authority paradigm for key distribution. The protocol also supports session resumption, allowing users to more quickly renegotiate a session, leveraging state data from past sessions between those two users. In this section, we only briefly discuss necessary elements of the protocol. For a more in-depth discussion, we refer to the standard [16].

In order to speed up session negotiation, TLS 1.3 provides several improvements over its predecessor, TLS 1.2 [17]. One of the major improvement points is the introduction of 0 Round Trip Time session resumption, or 0-RTT. This allows clients to send application data already in their first message to the server when initiating a session resumption. In the standard, this comes with the caveat that this so-called early data must be idempotent; it should not result in state changes. This is due to 0-RTT handshakes being weak against replay attacks.

The 0-RTT key data is transmitted to the client in a `NewSessionTicket` message. The server bundles up necessary data for it to continue the session later on, along with a Pre-Shared Key (PSK). The standard describes a structure for `NewSessionTicket` messages, but not for the tickets which these encapsulate, essentially leaving room for a variety of implementations from e.g. databases with lookup keys to self-encrypted and authenticated messages. In this work, we assume the mechanism first explained in RFC 5077 [10], a solution optimized for the Web, and which requires no server-side state variables on closed sessions. With this approach, the server encrypts the necessary state variables with a secret key, before handing them over to the client. Upon session resumption, the client sends over this encrypted bundle again, and these variables are then decrypted and in turn, can be used to decrypt the early data.

### 2.2 Double Ratchet Algorithm

The Double Ratchet Algorithm [14] is a cryptographic protocol enabling highly secure, asymmetric message exchange. Originally developed for Signal [20], it is now also used in WhatsApp [22]. It has received significant cryptographic attention and has been formally verified [5].

At the heart of this protocol lies a KDF-chain, which is a feedback loop where part of its output is fed back into the function as input for the next iteration, while also providing key material for encrypting messages. This creates a ratchet-like construction, because of the one-way nature of the KDF function; new keys can be generated constantly, while one can never retrieve old keys. Therefore, it is also common to refer to this construction as a ratchet. These properties provide ratchets with protection against replay attacks as well as forward secrecy.

A double ratchet is a setup where one “outer” ratchet and one or more “inner” ratchets work together to provide stronger security properties. The outer ratchet uses external entropy from a Diffie-Hellman (DH) handshake as input.

When the outer ratchet is spun (i.e. its KDF function is executed), it generates new input keys for its inner ratchets, thereby resetting them, and providing post-compromise, or break-in, protection. When only the inner ratchet is spun, it generates encryption keys for messages, and uses its own output as input for the next inner KDF execution. The outer ratchet is often called the DH ratchet, while the inner ratchets are called symmetric ratchets.

In the Double Ratchet Algorithm, both parties maintain one DH ratchet and two symmetric ratchets, for outgoing respectively incoming messages. In our work, we use only one symmetric ratchet, as only the client will ever initiate a connection, and the client will thus only need a ratchet for sending, while the server only needs one for receiving. For more details on the double ratchet algorithm, we refer the reader to [14].

## 3 ratchet TLS (rTLS)

In this section we describe our proposed extension in detail. Note that it is designed with the goal of making maximal use of existing extensions and utilities available in the TLS suite, and requiring only a minimal amount of change, to increase ease of verification and implementation.

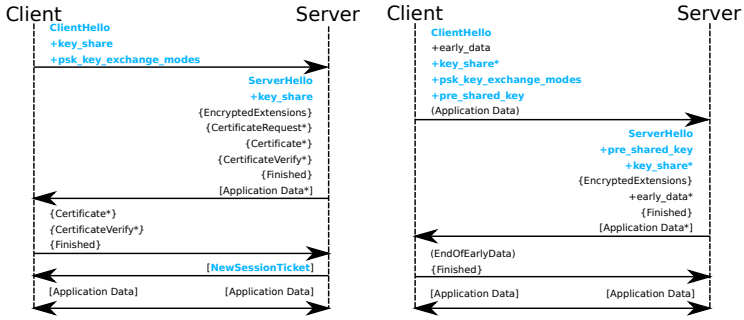
This extension uses a Symmetric Ratchet mechanism to generate the keys involved in session resumption. Additionally, it uses standard TLS mechanisms to provide an outer DH ratchet, providing forward secrecy and break-in protection. The original TLS specification leaves room to enable this elegantly by allowing us to transmit relevant data as a PSK. Then, we can make use of the existing `psk_key_exchange_modes` extension included in the RFC [16], by specifying a custom exchange mode for ratcheting to let the server know that we want to use this mode for session resumption. As we will see in the following sub-sections, this leads to a minimal number of changes in the protocol itself.

In the remainder of this section, we will first explore the differences between standard TLS handshakes and ratchet-mode handshakes in Sections 3.1 and 3.2, after which we explain the protocol setup and operation in detail in Section 3.3

### 3.1 Initial handshake

Figure 1a depicts the communication pattern of a typical initial handshake for a TLS session making use of our extension. To improve ease of comparison with the RFC [16], we have adopted the same syntax and included the same common extensions. In fact, the communication pattern of this handshake is indistinguishable from a standard TLS handshake. However, we further extend PSK-related extensions to achieve our goals. We denote those elements in the communication pattern that are relevant to this extension in blue.

The inclusion of the `psk_key_exchange_modes` extension in the first flight of messages signals to the server that the client wants to obtain a session ticket. To create our desired ratchet construction, we need to know what symmetric ciphers should be used during resumption, and also agree on a KDF. In principle, any




(a) The communication pattern of the initial handshake.

(b) The communication pattern of the resumption handshake.

Fig. 1: Figure 1a and 1b depict the initial respectively resumption handshake communication patterns. + denotes an extension, * denotes an optional or situation-dependent component while {} and [] denote encryption with a derivation of the handshake or application secret, respectively. Modifications from the original handshakes are printed in blue.

secure cipher and KDF can be used for this, however, in an effort to keep the number of required protocol changes to a minimum, we reuse the TLS cipher-suite agreed upon by the client and server, since this already includes apt choices for the required primitives while also guaranteeing that these are supported by the client. Note that the choice of cipher-suite is only definite after the server has replied with its own `ServerHello` and `key_share` messages. The DH secret key that is established through the `key_share` elements is used to derive all secrets used in TLS, including the PSK resumption secret. This means that whenever the `key_share` extension is included, the subsequently generated PSK resumption secret is derived from a fresh entropy source. The `psk_key_exchange_modes` extension list of a byte-sized enumerated type, indicating a PSK type. The currently standardized values are 0 for a static PSK and 1 for PSK with (EC)DHE key establishment. We add another value 3 indicating a PSK with key ratcheting. This list of types indicates to the server which PSK types are supported by the client.

After the initial handshake is done, the server sends a `NewSessionTicket` to the client. While it is allowed for a server to send multiple of these tickets in one session, this is not necessary: session resumption can add entropy when needed and thus provide fresh resumption tickets at a later point in time. Table 2 contains all fields in this structure, as specified in the TLS specification.

The ticket field contains an identifier that the client can later send to the server allowing it to identify the connection and access corresponding stored state. It does *not* contain an encryption key for resumption. Instead, a resumption master secret is derived as described in the standard: from the ticket nonce

Table 1: Layout of the `NewSessionTicket` structure.

| type | Field name | Description |
|---|---|---|
| `uint_32` | `ticket_lifetime` | ticket lifetime in seconds |
| `uint_32` | `ticket_age_add` | used to obscure ticket age |
| `opaque` | `ticket_nonce` | (max. 255 bytes) nonce |
| `opaque` | `ticket` | (max. $2^{32}$ bytes) ticket itself |
| `Extension` | `extensions` | (max. $2^{32}$ bytes ) extensions |

and master secret. The extension field must also contain the early data indication extension indicating that the PSK may be used for early data.

### 3.2 Session resumption

Upon resumption of a session using the ratchet PSK mode, the communication pattern once again looks identical to that of a standard TLS 0-RTT session resumption, as can be seen in Figure 1b. The blue text indicates fields that deviate in usage or content in this mode.

Firstly, the client chooses whether to include a `key_share` extension. This is not strictly necessary for every resumption, but depends on the desired granularity of break-in resilience; including a DH handshake in every resumption handshake implies that break-in recovery occurs after every resumption, while including these every $n$ resumption handshakes implies break-in recovery after every $n$ handshakes and so on. If the server receives a `key_share` from the client during resumption, it includes a `key_share` extension in its response carrying the necessary DH parameters, otherwise it does not need to include this extension.

The client also includes a `psk_key_exchange_modes` extension to indicate which PSK mode is used for the `pre_shared_key` field. This is mandated by the standard, and the content of this field is identical to the same field in the initial handshake.

Further, the client now includes a `pre_shared_key` field containing necessary data for the server to identify the connection as well as the ratchet index currently used by the client. This value is used by the server to determine if it missed any previous connection attempts, and if so, how many times it should ratchet its symmetric ratchet before decrypting the received early data. The `pre_shared_key` extension consists of two components: a list of `PskIdentity` and a list of `PskBinderEntry` structures. The latter is a list of Hash-Based Message Authentication Code (HMAC) values that authenticate the `ClientHello` up-to-and-including the list of `PskIdentity` entries, while the former consists of an `identity` and `obfuscated_ticket_age` value. The ticket age is further described in the standard and not important to this work, so we refrain from discussing it in detail. The `identity` value is defined as an opaque value in the standard, allowing us to populate it with a connection ID received from the server during the initial handshake (4 bytes), and the 1-byte ratchet index indicating the index of the symmetric KDF chain (after having derived the latest resumption master secret).

Every time the client initiates a resumption handshake with the server, the resumption master secret is ratcheted, going one step further down the KDF chain. From the ratcheted resumption master secret an early traffic secret is derived, which is used to encrypt the early application data sent by the client. Once the server has received a `ClientHello` with the necessary extensions for a ratchet-mode resumption, it can find the correct ratchet based on the connection ID obtained from the received `identity` field. It then spins this ratchet until the number of spins equals the ratchet index in the `identity` field.

When the resumption handshake includes the `key_share` extension, i.e. it initiates a DH handshake, the resulting shared secret is used to derive all subsequent secrets for a TLS session, as specified in its key schedule [16]. Notably, when a resumption secret already exists, the newly derived master secret depends on both the existing resumption secret and the DH shared secret. From this new master secret a new resumption master secret is then generated for use in future resumptions, and the ratchet index must be reset to 0. This construction ensures that an adversary cannot attack the protocol by replacing the client's `shared_key` field with its own parameters, as the adversary will not have access to the existing resumption and therefore cannot derive a correct next resumption secret.

We only reserve 1 byte for the ratchet index because we expect it to be reset to 0 well before 255 communication attempts have been made. Nevertheless, we add the requirement that if the ratchet index is 255, both parties must delete their PSK and negotiate a new PSK after a standard handshake.

### 3.3 Double Ratchet setup and operation

Next, we summarize the extra steps needed for both the initial- and resumption handshakes in a step-by-step fashion.

**Initial Handshake** The initial handshake is largely unmodified, but some special steps have to be taken by both the client and the server.

1. **ID generation**: The server generates a globally unique connection ID. This ID is transmitted to the client in the `NewSessionTicket`;
2. **Symmetric ratchet initialization**: The client and server initialize the ratchet index variable to 0. The symmetric ratchet key is the resumption master secret.
3. **Persistent state storage**: Both client and server store their state variables for anticipated session resumptions;

**Resumption** Below we describe the extra steps needed for a typical session resumption. A DH exchange may take place, but we do not consider that as an extra step – the TLS standard already accommodates for this.

**Client**

1. **Ratchet step**: The client ratchets its symmetric ratchet before the resumption master secret is used to derive any other secret. The early-data secret is thus derived from the ratcheted master secret;
2. **PSK exchange**: During the handshake, the client sends its ratchet index and connection ID to the server, as part of the `pre_shared_key`;

**Server**

1. **Access state:** The server receives a 0-RTT resumption, and after having verified the `pre_shared_key`'s HMAC field, finds the relevant state variables using the received connection ID as a key (e.g. in a hash map);
2. **Replay condition** The server ensures that $i_s < i_c$ where $i_s$ and $i_c$ are the server respectively received client ratchet indices for this connection.
3. **Ratchet step**: The server spins the symmetric ratchet $i_c - i_s$ times where $i_c$ is the received ratchet index in `pre_shared_key` and $i_s$ its own ratchet index. The early data encryption key is derived from the new state of the sym. ratchet;

**Both**

1. **Reset ratchet index**: If a DH exchange was performed during the resumption handshake, then the client and server reset their ratchet index to 0.
2. **Persistent state storage**: Both the client and server store their state variables for future session resumptions;

### 3.4 Ratchet state variables

This extension expects both the client and server to maintain some state for each connection. This state consists of the following data:

1. **Mapping**: a connection ID $\rightarrow$ ratchet mapping, to identify which ratchet belongs to which connection;
2. **Resumption Master Secret**: This is used to derive the keys used for encryption, upon next resumption (32 bytes);
3. **Ratchet Index**: To indicate the number of ratchet steps that occured since the last DH exchange (1 byte);

## 4 Evaluation

### 4.1 Security evaluation

In this section, we discuss the security properties of the proposed protocol extension. We only discuss the resumption handshake, as the initial handshake is left untouched by this extension. Firstly, note that because the `NewSessionTicket` message gets transmitted by the server as application data after the initial handshake, it is by definition authenticated, verified, and confidential. Since we require both the client and server to securely store their state variables, we can further assume that any keys derived from the `resumption_master_secret` can only be computed by the client and server.

**Replay attacks** We divide replay scenarios into two groups: those that occur within one DH handshake period, and those that span across at least one DH handshake. In the former, when an attacker replays a session resumption handshake $m$ without any modifications, the server will reject $m$ and not process the associated early data, as the replay condition $i_s < i_c$ will be violated. $c_i$ cannot be forged either, as it is protected by an HMAC and we assume security of the cryptographic hash function, and secrecy of the HMAC keys. In the second group, an attacker records $n$ different resumption message $m_0, \ldots, m_{n-1}$ where $n$ is the DH handshake frequency. Let $c_0, \ldots, c_{n-1}$ be the corresponding ratchet indices. Now, the attacker is certain that at least one DH handshake has been performed since $m_0$ was sent, and the next message $m_n$ will have ratchet index $c_n = c_0$. As the ratchet indices are equal, one could attempt to bypass the replay condition check. However, the resumption master keys for $m_0$ and $m_1$ are different, and therefore the HMAC keys used for the PSK binder fields are different. Thus, when an attacker sends $m_0$ to a server after $n$ resumptions have passed, the HMAC validation will fail *before* $i_c$ gets checked, and $m_0$ will thus be rejected.

**Forward Secrecy** The 0-RTT resumption also enjoys forward secrecy, as we only store the last resumption key. After every attempt, a key is derived using a cryptographic hash function, so it is not feasible for an adversary to compute past keys based on a compromised resumption key.

**Break-in protection** Additionally, the protocol enjoys break-in protection, proportional to the frequency of DH exchanges in resumption handshakes. These exchanges effectively function as the DH ratchet in the Signal protocol. the shared secret resulting from such a DH exchange is used as key input for the key derivation function, adding new entropy to it. This means that if an adversary compromises one of the endpoints at some moment in time, and extracts resumption keys from it, they will not be able to decrypt any messages after the next DH exchange has occurred; they do not possess the required shared secret.

### 4.2 Traffic Overhead estimation

**Initial handshake** The number of bytes transmitted by each side during the initial handshake is unchanged – the one addition to the protocol just defines an extra value for an enumerated field (`psk_key_exchange_modes`). After the handshake is done, the server transmits a `NewSessionTicket` message to the client. As this is part of the extension setup, in this context we consider this as part of the initial handshake; without it, resumption would not be possible. The structure and size of a minimal `NewSessionTicket` message is displayed in Table 2. The client does not need to send any reply to this message. We set the size of the ID field and nonce field to 4 respectively 32 bytes. Therefore, compared to no session resumption at all, minimal overhead is $14 + 4 + 32 = 50$ bytes. Compared to a session ticket in standard TLS 1.3, which is typically in the hundreds of bytes, this is a significant improvement.

Table 2: The message structure and size of a minimal `NewSessionTicket` message. Here $|ID|$ refers to the identifier length and $|N|$ to the size of the nonce.

| Size (bytes) | Field name |
|---|---|
| 4 | ticket_lifetime |
| 4 | ticket_age_add; |
| $\|N\|$ | ticket_nonce |
| $\|ID\|$ | ticket |
| 2 | extensions length |
| 4 | Early data extension |
| **Total** $14 + \|ID\| + \|N\|$ | |

Table 3: Symbol definitions for message elements, where $x \in \{c, s\}$ refers to the message sender (client resp. server).

| Symbol | Description |
|---|---|
| $H_x$ | (Client or Server) `Hello` |
| $ed_x$ | `early_data` |
| $D_x$ | Application data |
| $pex$ | `psk_key_exchange_modes` |
| $psk_x$ | `pre_shared_key` |
| $ks_x$ | `key_share` |
| $ee$ | `EncryptedExtensions` |
| $eed$ | `EndOfEarlyData` |
| $f$ | `Finished` |
| $R$ | Record Layer headers |

**Resumption handshake** The resumption handshake will ideally be performed much more often than the initial handshake, thus it is important that the traffic overhead for this handshake is as small as possible. The fixed cost for any resumption handshake consists of boilerplate parts of the handshake that cannot be eliminated without rigorous change to the protocol. In the following, we write client and server as $c$ and $s$, respectively. We map symbols to every message element in the resumption handshake in Table 3, where $x$ can be either $c$ or $s$ to indicate the message sender. We refer to the size of message $X$ as $|X|$.

We define the fixed cost $C$ of any 0-RTT resumption handshake as:

$$C = 3|R| + |H_c| + |H_s| + |ed_c| + |pex| + |ee| + 2|f| + |eed|$$

This cost is not a fixed number of bytes, but rather is not negotiable; any PSK extension will have to include these elements, and their size is independent of the actual PSK mode. The total cost of a minimal resumption handshake where the server does not respond with any early data is $C + |psk_c| + |psk_s|$. Note that $ks_c$ and $ks_s$ are not required for a minimal handshake. Conform to the standard, $pks_s$ is defined as a 2-byte value representing an identity index in $psk_c$, and is wrapped in a 4-byte TLS extension structure. $ks_c$ is more complex however, and we write the full layout in Table 4. As we only send one identity and binder, The size of $psk_c$ becomes $|psk_c| = 15 + \alpha + \beta$, where $\alpha$ denotes the size of the `identity` field, and $\beta$ the size of the binder HMAC. The identifier field $PSKID$ can be written as $PSKID = ID||i$ where $ID$ is the identifier received in the session ticket during the initial handshake and $i$ is the symmetric KDF chain index. Now, $|PSKID| = |ID| + 1 = 5$. The exact value of $\beta$ depends on the chosen HMAC function, which is usually either Secure Hash Algorithm (SHA)-256 (32 bytes) or SHA-384 (48 bytes). The complete traffic cost $c_1$ for session resumption can thus be written as $c = |psk_s| + |psk_c| + C = 26 + \beta + C$, and is $58 + C$ if SHA-256 is chosen.

Table 4: Layout of the `pre_shared_key` structure and its sub-structures, when sent by a client.

| **pre_shared_key** | | |
|---|---|---|
| **Size** | **Field name** | **Description** |
| 2 | `extension_type` | Extension type |
| 2 | `extension_data` | Size of the extension |
| 2 | `PSKIdentities_length` | Nr. of PSK identities |
| | `identities` | `PSKIdentity` values |
| 2 | `binders_length` | Nr. of PSK binders |
| | `binders` | `PSKBinder` values |
| **PSKIdentity** | | |
| 2 | `identity length` | Size of identity field |
| $\alpha$ | `identity` | value of this identity |
| 4 | `obfuscated_ticket_age` | ticket age (see [16]) |
| **PSKBinder** | | |
| 1 | `binder length` | size of the binder value |
| $\beta$ | `binder` | HMAC value (see [16]) |

When a DH exchange is included, we will have to add the size of the $ks_c$ and $ks_s$ elements. The size of $ks_c$ is of variable length depending on the number of supported DH groups the client advertises. Each key share entry takes up $4 + l$ bytes where $l$ is the size of the supported group. The smallest supported group is X25519 with a 32-byte field, while the largest is P-521 with 132 bytes. $ks_c$ also reserves 2 bytes to denote the number of listed groups, therefore $ks_c = 6 + l$. The server replies with a single key share entry, thus $ks_s = 4 + l$. As with any TLS extension, these entries are wrapped in an extension structure with a 4-byte type field. The total cost of a resumption with DH exchange is thus $c_2 = c_1 + |ks_c| + |ks_s| = c_1 + 18 + 2l$.

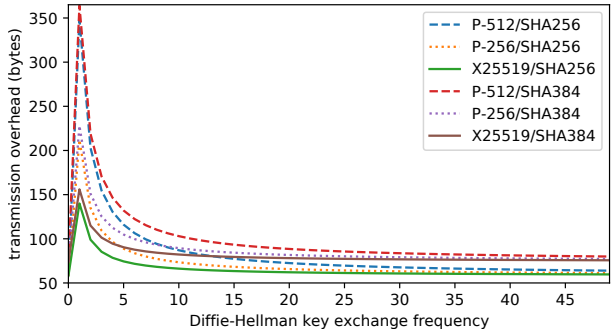



Fig. 2: Average transmission overhead v. DH key exchange frequency

If we take into account a `key_share` every $n$ messages, we arrive at the final equation for the total average cost $c_t$:

$$c_t = \left\{ \begin{array}{ll} 26 + \beta + C & \text{for } n = 0 \\ 26 + \beta + \frac{18+2l}{n} + C & \text{for } n > 0 \end{array} \right\}$$

where $\beta$ is the hash digest size, $l$ the elliptic curve coordinate length, $n$ the DH handshake rate, and $C$ the fixed cost. Figure 2 shows the average overhead versus the key exchange frequency, for various common cipher suites.

Giving an exact value for $C$ is somewhat difficult: multiple fields in $H_c$, $H_s$, and $ee$ can vary a lot in length, depending on the supported cipher suites and provided extensions among other things. Instead, we count the minimum size for these fields as they are defined in the standard, thereby giving a lower bound for $C$. Note that in practice, a handshake with so few extensions is not useful for overhead minimization, as more round-trips will be needed to establish necessary parameters such as the cipher suite. Moreover, it leaves out extensions meant to increase overall security. Minimal sizes, including all headers, for $H_c$ and $H_s$ are 50 and 48 bytes, respectively. $ed_c$ and $eed$ both require 2 and 4 bytes. $pex$ is at least $3 + m$ bytes in size, where $m$ is the number of supported modes (at least 1). $ee$ is at least 6 bytes in size, but may vary a lot, depending on the supported extensions. The length of $f$ is determined by the chosen hash function. The record layer headers are 5 bytes in size. With one PSK key exchange mode and the SHA-256 hash function, the total cost of $C$ is then at least 193 bytes. Therefore, the lower bound on transmission overhead of a resumption handshake with our extension is 251 bytes without, or 333 bytes with a key exchange.

### 4.3 Storage overhead estimation

Both the client and server need to store some state variables in between sessions. This differs from the standard session resumption protocol where only the client stores the PSK. The client needs to securely store the secret KDF key (depends on digest size), as well as its connection ID (4 bytes) and the ratchet index (1 byte). The client thus needs to store 37 bytes if SHA-256 is used.

The server needs to store the same amount of state, but for every client that it shares a ratchet for resumption with. This can be done through e.g. a hash map using the connection ID as a key, and a structure containing the other state variables as value. If state is being kept for the maximum amount of clients of $2^{32}$ (with a 4-byte connection ID), this amounts to roughly 270GB worth of data.

### 4.4 Overhead comparison with TLS 1.3

Based on measurements performed on OpenSSL [13], a standard PSK in TLS 1.3 adds 571 and 603 bytes of overhead, when SHA-256 respectively SHA-384 is used. In Table 5 we compare the overhead of rTLS for various values of $n$ to that of a standard TLS 1.3 PSK. We use a higher value of $C$, obtained from handshake measurements in OpenSSL, which includes a minimal number of extensions by

default, and acts as an indicative value that represents a lightweight use case. In this table, the values are computed using the smallest allowed hash function (SHA-256) and curve (X25519). As can be seen, a rTLS PSK requires only roughly 11% of the traffic overhead compared to a standard TLS PSK, and can be expected to reduce the total amount of transmitted data roughly by half.

Table 5: A comparison between rTLS session resumption and openSSL standard session resumption

**Indicative Lightweight Use** ($C = 408$)

| **Scenario** | **Avg. Overhead (b)** | **Avg. Total size (b)** |
|---|---|---|
| rTLS, $n = 0$ | 58 | 466 |
| rTLS, $n = 1$ | 108 | 516 |
| rTLS, $n = 10$ | 63 | 471 |
| Standard TLS 1.3 | 571 | 979 |

## 5 Related Work

There exist ample communication security protocols aimed at embedded devices [1]. We look at the TLS protocol and its variants, specifically those that are relevant to the usage of this protocol in embedded environments.

Initially developed for Web security, TLS is now gaining traction in the IoT world, partly due to widely available libraries and broad support in software relevant to IoT. For example, many Message Queuing Telemetry Transport (MQTT) brokers support TLS as a security layer.

While this is fine for most devices (mostly upwards from class 1 in the Internet Engineering Task Force (IETF) classification [4]), it becomes problematic when working with class 0 or low-end class 1 devices, as they do not possess the capability to maintain TLS connections or can simply not afford it due to resource constraints (e.g. due to a power budget). To address this, several optimizations have been proposed over the years. One of the first was Sizzle [7], which is an implementation of the Secure Socket Layer (SSL) protocol, and is capable of running on extremely constrained devices with only tens of kilobytes of memory. While the authors showed that heavyweight cryptographic operations required for the protocol to function were certainly possible on heavily constrained devices, they did not attempt to reduce the amount of transmitted data.

Datagram Transport Layer Security (DTLS) [18] modifies the TLS protocol to work over User Datagram Protocol (UDP), while retaining most of the security guarantees provided by TLS. This reduces the data overhead and latency somewhat. There exist multiple open-source implementations [23], and several works exist detailing extremely lightweight implementations [3, 11]. In these works,

lightweight mostly pertains to computational and memory cost, while transmission overhead is either not addressed or addressed to a much lesser degree. Other approaches have been taken as well, such as [15], compressing DTLS messages to fit into 6LowPAN frames.

Several extensions for TLS have been proposed that also bring the potential to lower message overhead. The TLS Cached Info specification [19] allows clients to store server certificates and certificate requests, making it possible to leave these out in future handshakes. The TLS Raw Public Key extension [24] allows clients and servers to authenticate each other through public keys, instead of X.509 certificates. This can significantly reduce the handshake size. This method does require an out-of-band means of verifying public keys, which might very well be possible in a controlled environment such as a factory. Another promising adaptation of TLS that might lower the size overhead of TLS significantly is the Compact Transport Layer Security (CTLS) IETF draft [6]. In this draft, the authors propose optimizing the TLS protocol for size by eliminating redundancy where possible and making aggressive use of space-optimization techniques such as variable-length integers. The result is isomorphic to TLS, but not interoperable.

TLS is also proposed as the default mechanism to secure connections in the QUIC protocol, a network protocol building on UDP that provides advanced features such as multiplexing and authenticated encryption of its data by default.

Session resumption in TLS 1.3 has been subject to debate, as it is vulnerable to replay attacks and provides no forward secrecy [16]. While for a Web environment, there exists some justification for these design choices, for an IoT environment where short conversations with short messages are the norm, this is less than ideal, as it effectively removes the possibility to optimize overhead through use of the session resumption protocol. None of the extensions discussed in this section address session resumption, which means that this is an open issue we think has significant potential for minimizing protocol overhead, when designed carefully.

At the time of writing, National Institute of Standards and Technology (NIST) is hosting an ongoing competition for lightweight cryptographic primitives [12]. Many of the candidates specifically target very short messages. Once the candidates have received sufficient cryptanalytic attention, these can become valuable tools in future lightweight communication protocols, as well as potentially helping protocols such as TLS adapt to constrained devices.

In [8], Hall-Andersen et al. acknowledge the complexity of TLS and propose nQUIC as a lightweight, less complex alternative to QUIC's default TLS configuration. Their experiments show a significant reduction in bandwidth compared to TLS.

## 6 Conclusion

In this work, we proposed an IoT-friendly and standard-compliant adaption of the TLS 1.3 0-RTT session resumption protocol. We first argued that in order

to be applicable to IoT, replay resistance is a necessary property, as lightweight sensor devices are much more likely to transmit data that will change server state.

Building from the observation that in IoT scenarios the group of possible clients for a server changes relatively slowly and is typically much smaller than possible clients for a Web server, we argued that it is reasonable to require a server to keep some state variables for each of its clients. We then took inspiration from the Double Ratchet algorithm to design a 0-RTT resumption protocol that fits neatly into the existing message structure, and makes use of existing functionality where possible. In our extension, the PSK utilizes a ratchet construction, which provides replay protection as well as forward secrecy and break-in resilience to early data transmitted in a 0-RTT handshake. The introduction of these properties in the 0-RTT subprotocol is a step towards making TLS suitable for IoT scenarios.

We estimated a lower bound of 193 bytes on traffic overhead for any 0-RTT resumption protocol in TLS 1.3, and then showed that our protocol requires at least 251 bytes of traffic overhead. Compared to the standard session resumption overhead of roughly 764 bytes, this is a significant improvement.

In future work, we aim to further reduce the transmission overhead by exploring different opportunities, such as replacing the original message structure for resumption altogether, thereby reducing the fixed cost.

**Acknowledgements**

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 764785, FORA – Fog computing for Robotics and Industrial Automation

## References

1. Authentication protocols for internet of things: A comprehensive survey. Security and Communication Networks
2. AT&T: LTE-M and NB-IoT, `https://www.business.att.com/products/lpwa.html`
3. Bergmann, O., Gerdes, S., Bormann, C.: Simple keys for simple smart objects. In: Workshop on Smart Object Security (2012)
4. Bormann, C., Ersue, M., Keränen, A.: Terminology for Constrained-Node Networks. RFC 7228 (May 2014). https://doi.org/10.17487/RFC7228, `https://rfc-editor.org/rfc/rfc7228.txt`
5. Cohn-Gordon, K., Cremers, C., Dowling, B., Garratt, L., Stebila, D.: A formal security analysis of the signal messaging protocol. In: 2017 IEEE European Symposium on Security and Privacy (EuroS&P). pp. 451–466 (April 2017). https://doi.org/10.1109/EuroSP.2017.27
6. E. Rescorla, R. Barnes, H.T.: Compact TLS 1.3 (IETF draft), `https://datatracker.ietf.org/doc/draft-rescorla-tls-ctls/`

7. Gupta, V., Wurm, M., Zhu, Y., Millard, M., Fung, S., Gura, N., Eberle, H., Shantz, S.C.: Sizzle: A standards-based end-to-end security architecture for the embedded internet. Tech. rep., USA (2005)
8. Hall-Andersen, M., Wong, D., Sullivan, N., Chator, A.: NQUIC: Noise-Based QUIC Packet Protection. In: Proceedings of the Workshop on the Evolution, Performance, and Interoperability of QUIC. p. 22–28. EPIQ'18, Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3284850.3284854
9. Hologram: Hologram pricing, `https://hologram.io/pricing/`
10. J. Salowey, H. Zhou, P.E.H.T.: Transport Layer Security (TLS) Session Resumption without Server-Side State. RFC 5077 (Jan 2008). https://doi.org/10.17487/RFC5077, `https://rfc-editor.org/rfc/rfc8446.txt`
11. Kothmayr, T., Schmitt, C., Hu, W., Brünig, M., Carle, G.: A dtls based end-to-end security architecture for the internet of things with two-way authentication. In: 37th Annual IEEE Conference on Local Computer Networks - Workshops. pp. 956–963 (Oct 2012). https://doi.org/10.1109/LCNW.2012.6424088
12. NIST: Lightweight Cryptography, `https://csrc.nist.gov/projects/lightweight-cryptography`
13. OpenSSL Software Foundation: OpenSSL, `https://www.openssl.org`
14. Perrin, T., Marlinspike, M.: The double ratchet algorithm (2016), `https://www.signal.org/docs/specifications/doubleratchet/doubleratchet.pdf`
15. Raza, S., Trabalza, D., Voigt, T.: 6LoWPAN Compressed DTLS for CoAP. In: 2012 IEEE 8th International Conference on Distributed Computing in Sensor Systems. pp. 287–289 (May 2012). https://doi.org/10.1109/DCOSS.2012.55
16. Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446 (Aug 2018). https://doi.org/10.17487/RFC8446, `https://rfc-editor.org/rfc/rfc8446.txt`
17. Rescorla, E., Dierks, T.: The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Aug 2008). https://doi.org/10.17487/RFC5246, `https://rfc-editor.org/rfc/rfc5246.txt`
18. Rescorla, E., Modadugu, N.: Datagram Transport Layer Security. RFC 4347 (Apr 2006). https://doi.org/10.17487/RFC4347, `https://rfc-editor.org/rfc/rfc4347.txt`
19. Santesson, S., Tschofenig, H.: Transport Layer Security (TLS) Cached Information Extension. RFC 7924 (Jul 2016). https://doi.org/10.17487/RFC7924, `https://rfc-editor.org/rfc/rfc7924.txt`
20. Systems, O.: Signal, `https://www.signal.org`
21. Verizon: Verizon thingspace, `https://thingspace.verizon.com/service/connectivity/`
22. WhatsApp: Whatsapp encryption overview, `https://www.whatsapp.com/security/WhatsApp-Security-Whitepaper.pdf`
23. WolfSSL: TLS 1.3 Protocol Support, `https://www.wolfssl.com/docs/tls13/`
24. Wouters, P., Tschofenig, H., Gilmore, J., Weiler, S., Kivinen, T.: Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS). RFC 7250 (Jun 2014). https://doi.org/10.17487/RFC7250, `https://rfc-editor.org/rfc/rfc7250.txt`

Paper F

# rTLS: Secure and Efficient TLS Session Resumption for the Internet of Things

K. Tange, S. A Mödersheim, A Lalos, X. Fafoutis, and N. Dragoni. "rTLS: Secure and Efficient TLS Session Resumption for the Internet of Things." In: *Sensors* (2021). DOI: 10.3390/s21196524

sensors

MDPI

*Article*

# rTLS: Secure and Efficient TLS Session Resumption for the Internet of Things †

**Koen Tange *, Sebastian Mödersheim, Apostolos Lalos, Xenofon Fafoutis and Nicola Dragoni**

DTU Compute, Department of Applied Mathematics and Computer Science, Technical University of Denmark, Richard Petersens Plads, 2800 Kongens Lyngby, Denmark; samo@dtu.dk (S.M.); lalosapost@gmail.com (A.L.); xefa@dtu.dk (X.F.); ndra@dtu.dk (N.D.)

* Correspondence: kpta@dtu.dk

† This paper is an extended version of our paper published in the 2020 International Conference on Information and Communications Security as "rTLS: Lightweight TLS Session Resumption for Constrained IoT Devices".

**Abstract:** In recent years, the Transport Layer Security (TLS) protocol has enjoyed rapid growth as a security protocol for the Internet of Things (IoT). In its newest iteration, TLS 1.3, the Internet Engineering Task Force (IETF) has standardized a zero round-trip time (0-RTT) session resumption sub-protocol, allowing clients to already transmit application data in their first message to the server, provided they have shared session resumption details in a previous handshake. Since it is common for IoT devices to transmit periodic messages to a server, this 0-RTT protocol can help in reducing bandwidth overhead. Unfortunately, the sub-protocol has been designed for the Web and is susceptible to replay attacks. In our previous work, we adapted the 0-RTT protocol to strengthen it against replay attacks, while also reducing bandwidth overhead, thus making it more suitable for IoT applications. However, we did not include a formal security analysis of the protocol. In this work, we address this and provide a formal security analysis using OFMC. Further, we have included more accurate estimates on its performance, as well as making minor adjustments to the protocol itself to reduce implementation ambiguity and improve resilience.

**Keywords:** network; security; protocol; formal verification



**Citation:** Tange, K.; Mödersheim, S.; Lalos, A.; Fafoutis, X.; Dragoni, N rTLS: Secure and Efficient TLS Session Resumption for the Internet of Things. *Sensors* **2021**, *21*, 6524. https://doi.org/10.3390/s21196524

Academic Editor: Wenjuan Li

Received: 8 September 2021
Accepted: 26 September 2021
Published: 29 September 2021



## 1. Introduction

There are many examples of well-established communication protocols that are able to satisfy contextually-defined requirements and are in use in modern technology. Arguably the most well-known example is the TLS protocol [1]. This protocol is widely used in today's Internet, although originally designed for the Web. Recently, this protocol has been gaining traction in the IoT domain, as well. To better suit the heterogeneous needs present in this domain, adaptions and new extensions of the TLS protocol are needed, specifically to enable extremely lightweight devices to partake in TLS connections, as well.

While securely browsing the Web, it is not unusual for a TLS handshake to require between 1 and 4 KB of traffic. For consumer devices with browsers, this is often not an issue, but it is a lot of traffic overhead for lightweight devices running on battery power, where powering a wireless radio is very costly. Therefore, there is a need to reduce this handshake overhead as much as possible. To reduce bandwidth overhead, as well as latency, TLS 1.3 features a new zero round-trip time (0-RTT) session resumption protocol capable of transmitting application data already in its first flight of messages. This allows for the quick reopening of a session without having to go through an expensive full handshake again. Unfortunately, this resumption protocol is susceptible to replay attacks, a design decision deemed acceptable by the TLS committee since web connections usually start an HTTP GET request, which is idempotent. For IoT applications, however, this assumption does not hold. For example, consider a temperature sensor that periodically reports its

readings to a server. This is our primary motivation for extending the 0-RTT protocol with an IoT-friendly alternative.

As a secondary goal, we aim to further reduce traffic overhead of the resumption handshake. The reason for this is that the financial costs of sending TLS handshakes for every periodic IoT transmission could increase rapidly. For example, it is expected that many network providers for 5G and Low-Power Wide Area Networks (LPWAN) will charge their users based on data usage [2–4]. Additionally, the transmission cost of setting up a secure connection should be within reasonable proportions to the size of the payload itself. If it is tens of times higher, users might choose not to use secure channels, or alternatively implement their own cryptographic protocols, with associated risks.

The standard TLS protocol is designed with the assumption in mind that servers do not keep state on a client in between sessions. This is justifiable for the Web, where the set of potential clients is unbounded and it is often hard to predict if a client will resume a session at all. However, for many IoT systems, it is reasonable to assume that the set of clients is fairly static and known a priori, or otherwise traceable through a key infrastructure. Thus, keeping state on these clients in between sessions is a lot easier than for the Web, and having state information at the ready can aid in further reducing handshake overhead. TLS 1.3 does not offer any such mechanism, leaving this as a gap that can be filled by IoT-friendly extensions.

In previous work, we introduced rTLS [5], a TLS extension that can authenticate two endpoints and set up a secure connection with minimal additional overhead, given that the client and server have initiated a session in the past. We described how the extension changes the 0-RTT session resumption protocol to reduce overhead compared to the standard protocol, while adding new security features including replay protection, forward secrecy, and break-in protection. We built the protocol on the assumption that servers can store state on clients, with the IoT in mind. Additionally, we provided equations on the lower bound for traffic overhead of any TLS resumption protocol, as well as our proposed extension, and compared it to overhead observed from the OpenSSL [6] implementation of TLS 1.3. We also provided estimations for storage overhead for both client and server.

In this work, we extend upon the original work on multiple points. The first and main extension point is the addition of a formal analysis of protocol security using the Open-source Fixed-point Model Checker (OFMC) [7], including the development of a new intermediate specification language to help with this verification. To this end, the original text covering security analysis has been completely revised and is included in a new separate section, Section 4. Other points include the addition of more accurate performance and storage overhead estimates, based on better observations and a better understanding of TLS implementations. The protocol itself has also received some minor updates relating mainly to which data is stored between sessions. Further, we have made minor improvements to the presentation of the protocol design section.

The remainder of this paper is organized as follows: In Section 2, we briefly discuss the foundations necessary to understand our proposed extension. We then explain our extension in detail in Section 3. After that, we provide a formal analysis of several security properties in Section 4. Then, we evaluate the storage and transmission overhead in Section 5, after which we discuss related work in Section 6. Finally, we conclude in Section 7.

## 2. Preliminaries

In this section, we briefly discuss the essentials needed to understand the concepts upon which our proposed solution is built. First, we summarize the TLS 1.3 protocol, after which we discuss the Double Ratchet algorithm and Key Derivation Function (KDF)-chains. Both of these are described in more detail in the the Signal documentation pages [8].

### *2.1. TLS 1.3*

The TLS 1.3 protocol [1] establishes a secure communication channel (a session) between a client (the initiator) and a server. The most common use establishes one-way authentication; only the server is authenticated, using a key distribution method, such as certificate authorities.

In the most typical scenario, one-way authentication is provided, that is, the server authenticates itself to the client, building on the certificate authority paradigm for key distribution. The protocol also supports session resumption, allowing users to quickly renegotiate a session in fewer round-trips, leveraging state data from past sessions between those two users. In this section, we only briefly discuss necessary elements of the protocol. For a more in-depth discussion, we refer to the standard [1].

In order to speed up session negotiation, TLS 1.3 provides several improvements over its predecessor, TLS 1.2 [9]. One of the major improvement points is the introduction of 0 Round-Trip Time session resumption, or 0-RTT. The defining feature of 0-RTT resumption is that application data can already be transmitted to the server in the first message sent by the client. The standard refers to this as early data. The standard specification comes with a caveat: early data must be idempotent, that is, it should result in a state change on the server. This is because 0-RTT handshakes are susceptible to replay attacks.

The 0-RTT protocol is set up as follows: After the initial (non-resumption) handshake, 0-RTT key data is transmitted to the client in a `NewSessionTicket` message. The message contains a ticket, as well as other data later needed by the server to continue a session. When the client later initiates a session resumption, it will send this ticket to the server as part of the first message, enabling the server to continue the session without needing a full TLS handshake. Note that, while the standard describes a structure for `NewSessionTicket` messages, it does not not prescribe a specific structure for the tickets which these encapsulate, essentially leaving room for a variety of implementations from, e.g., databases with lookup keys to self-encrypted and authenticated messages. In this work, we assume the mechanism first introduced in RFC 5077 [10], a solution optimized for the Web, and which requires no server-side state variables on closed sessions. This method has already seen use in TLS 1.2 and is supported by many TLS libraries, such as OpenSSL [6] and WolfSSL [11]. With this approach, the ticket contains all state variables needed by the server and is encrypted with a key known only to the server. From the client's perspective, it receives an opaque blob of encrypted data. When the client initiates session resumption, it will send over this ticket, which can then be decrypted by the server, enabling it to restore the session state.

### *2.2. Double Ratchet Algorithm*

The Double Ratchet Algorithm [8] is a cryptographic protocol enabling highly secure, asymmetric message exchange between multiple parties. The protocol was originally developed for Signal [12] but is now also used in the popular messaging app WhatsApp [13]. It has received significant cryptographic attention and has been formally verified [14].

At the heart of this protocol lies a KDF-chain. This is a KDF function that can be iteratively applied to its own output, creating a feedback loop where part of the output of each iteration is fed back into the function as input for the next iteration, while also providing key material for encrypting messages. This construction is commonly referred to as a ratchet because of the one-way nature of the KDF function; new keys can be generated constantly, but one cannot reverse the process to produce old keys. Because of this, when used correctly ratchets are resilient against replay attacks and can provide forward secrecy.

A double ratchet is a combination of multiple ratchet-like constructions. Firstly, it contains one "outer" ratchet, and secondly one or more "inner" ratchets. These ratchets work together to provide stronger security properties. The outer ratchet receives periodic (e.g., every 10 messages) external entropy from a Diffie-Hellman (DH) handshake as part of its input. Whenever this outer ratchet is spun (i.e., its KDF function is invoked), its output includes new input keys for the inner ratchets. These inner ratchets are reset completely and

seeded with the new input keys. This provides post-compromise, or break-in, protection. One can also spin just one of the inner ratchets to produce keys that can be used to encrypt or decrypt messages. The outer ratchet is also commonly referred to as the DH ratchet, and the inner ratchets are often called symmetric ratchets (because their input keys are symmetric). Figure 1 illustrates the ratchet process. As can be seen in the figure, with the progression of time, multiple symmetric ratchets may be instantiated in succession (or, in other words, the same inner ratchet is reset whenever the outer ratchet is spun). The first inner ratchet, producing keys $K1$, $K2$, and $K3$ received entropy from the first DH handshake (this is visualized by a yellow color). When the outer ratchet is spun a second time, the inner ratchet will be reset and receive fresh entropy from the outer ratchet, which we emphasize with a green color, to indicate that the inner ratchets have different entropy.

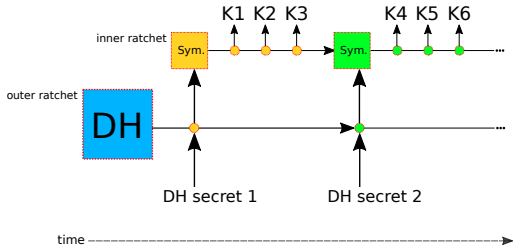



**Figure 1.** The double ratchet process and structure. Rectangles indicate initial states, circles indicate "spins" of the ratchets, and colors indicate the flow of entropy from a DH exchange. The outer ratchet is depicted on the bottom, with the inner ratchet above it.

In the standard Double Ratchet Algorithm, both parties maintain one DH ratchet and two symmetric ratchets, respectively, for outgoing incoming messages. In our work, we use only one symmetric ratchet, as only the client will ever initiate a connection, and, thus, the client will only need a ratchet for sending, while the server only needs one for receiving. For more details on the double ratchet algorithm, we refer the reader to Reference [8].

## 3. Ratchet TLS (rTLS)

In this section, we specify our proposed extension, ratchet TLS (rTLS). Note that we have designed the specification with the following design goals in mind: firstly, to maximize the use of existing extensions and utilities in the TLS suite; secondly, to require only minimal changes to those parts that are changed; thirdly to minimize bandwidth overhead; and, finally, to provide stronger 0-RTT security properties.

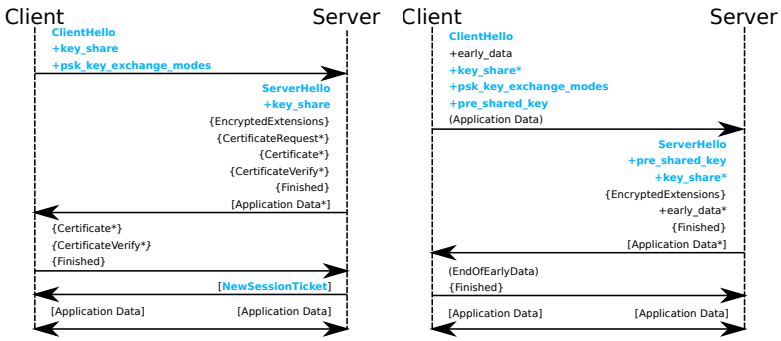
Our extension relies on a Symmetric Ratchet to generate the encryption keys for early data encryption in session resumption. Further, it builds on standard TLS extensions to provide an (outer) DH ratchet, providing forward secrecy and break-in protection. We can elegantly transmit the data relevant to our new extension as a Pre-Shared Key (PSK), and can signal support for rTLS by making use of the `psk_key_exchange_modes` extension specified in RFC 8446 [1]. As the following sub-sections will show, the changes necessary to the TLS protocol to achieve this are kept to a minimum.

We will first discuss the differences between standard TLS handshakes and ratchet-mode handshakes. First, we will discuss the changes to the initial handshake in Section 3.1, after which we look at the differences for the resumption handshake in Section 3.2. Finally, we specify the protocol setup and operation in detail in Section 3.3.

### *3.1. Initial Handshake*

Figure 2a shows the communication pattern of the initial handshake of a typical rTLS-enabled TLS sessions. For ease of comparison with the RFC [1], we utilize the same syntax and have adopted the common extensions depicted in the standard. The communication

pattern of this handshake looks identical to a standard TLS handshake. This is because no new extensions are added. Rather, we further extend the existing PSK-related extensions. In the figure, the extensions that are affected by rTLS are denoted in blue.




(**a**) The communication pattern of the initial handshake.

(**b**) The communication pattern of the resumption handshake.

**Figure 2.** (**a**,**b**) The initial resumption handshake communication patterns, respectively. + denotes an extension, and * denotes an optional or situational component, while {} and [] denote encryption with a derivation of the handshake or application secret, respectively. Modifications from the original handshakes are printed in blue.

In the first flight of messages, the `psk_key_exchange_modes` extension is included by the client to tell the server that it wants to obtain a session ticket. The client and server must agree on a KDF and which ciphers to use for resumption. In principle, any secure KDF and cipher can be used; however, to keep the number of protocol changes to a minimum, we reuse the ciphers included in the cipher-suite, agreed upon by both parties through the TLS handshake. This way, we can ensure that both client and server support the chosen ciphers. Additionally, this makes reasoning about the protocol easier because we only have to consider one type of KDF. Note that the agreement on which cipher-suite to use is only finalized after the server has sent its `ServerHello` message. The secret key derived from the DH handshake conducted through the `key_share` extension is used in the derivation of all secrets used in TLS, including the PSK resumption secret. Thus, whenever a `key_share` extension is part of a handshake, a fresh entropy source is introduced into the key schedule. The `psk_key_exchange_modes` extension consists of a list of a byte-sized enumerated type. This type indicates the PSK mode to use. Currently, TLS supports value 0 for a static PSK and 1 for an (EC)DHE established PSK. We extend this by adding another value indicating a PSK with key ratcheting. This list of PSK modes advertises which types the client supports to the server.

After finalizing the initial handshake, the server sends a `NewSessionTicket` to the client. The specification explicitly allows for sending multiple tickets in one session, although this is not necessary, since session resumption by itself can add fresh entropy when needed (through a DH handshake), thereby introducing freshness into resumption tickets at a later time. All fields specified in the TLS specification for the `NewSessionTicket` structure are listed in Table 1.

Since the specification enables the ticket field to carry opaque binary data, we specify it to include a 4-byte "connection identifier" that the server can later use to uniquely identify the session so that it may access the locally stored state for that session. Note that it does *not* include a shared resumption key. The standard defines the resumption key as being derived from the ticket nonce and TLS master secret. Further, we include a fresh DH public key generated by the server, which the client will use to initialize its DH ratchet for the first

resumption. In later resumptions, DH parameters can be shared through the `key_share` extensions; however, for the very first resumption, we have to make an exception since the client needs to be able to initialize the ratchet. The `extensions` field should include the `early_data` extension, which tells the client that this PSK can be used to transmit early data.

**Table 1.** Layout of the `NewSessionTicket` structure.

| Type | Field Name | Description |
|---|---|---|
| `uint_32` | `ticket_lifetime` | ticket lifetime in seconds |
| `uint_32` | `ticket_age_add` | used to obscure ticket age |
| `opaque` | `ticket_nonce` | (max. 255 bytes) nonce |
| `opaque` | `ticket` | (max. $2^{32}$ bytes) ticket itself |
| `Extension` | `extensions` | (max. $2^{32}$ bytes ) extensions |

### *3.2. Session Resumption*

Figure 2b shows the resumption handshake communication pattern. Again, it looks indistinguishable from a standard TLS 0-RTT resumption handshake, but the elements noted in blue text indicate that they deviate in usage or content in rTLS.

Firstly, the client can optionally include a `key_share` extension. This is not necessary for every resumption handshake, and the exact frequency with which these should be included depends on the desired granularity of break-in resilience; if it is included in every handshake, then break-in recovery occurs with every resumption, while including it only every $n$ resumption handshakes will imply break-in recovery every $n$ resumptions and so on. We refer to the frequency with which $n$ is included as the DH exchange period. If the server receives a `key_share` from the client, it will reply with a `key_share` of its own, to complete the DH handshake.

Secondly, the client includes a `psk_key_exchange_modes` extension indicating which PSK mode is used for the `pre_shared_key` field. This should be set to the enumerated type value representing rTLS.

The `pre_shared_key` field contains the connection identifier, which the server can identify this session, as well as the current ratchet index used by the client. Based on this index, the server can determine if it missed any previous resumption attempts and spin its ratchet enough times to catch up and ensure the encryption keys are synchronized with the client. The `pre_shared_key` contains a list of `PskIdentity` structures, as well as a list of `PskBinderEntry` structures. Each entry in the `PskIdentity` consists of an `identity` value and a `obfuscated_ticket_age` value. We do not make any changes to the ticket age, and refer to the standard for details on how to derive the obfuscated ticket age. The `identify` field is defined as opaque binary data, which allows us to use it to transmit the 4-byte connection ID that was transmitted by the server in the initial handshake, as well as a 1-byte ratchet index representing the current index (after having derived the latest resumption master secret) of the symmetric KDF-chain. The `PskBinderEntry` list is a list of Hash-Based Message Authentication Code (HMAC) values which authenticate the handshake from the `ClientHello` up to (and including) the list of `PSKIdentity` entries.

The client spins its symmetric ratchet whenever it initiates a resumption handshake, thereby ensuring that the resumption master secret changes all the time. As described in the standard, an early traffic is derived from the resumption master secret, which, in turn, is used as an encryption key for the early data. The server can decrypt this early data once it has received a `ClientHello` with the necessary extensions for ratchet-mode resumption. It is then able to access the ratchet state for the given connection ID and spin this ratchet until it is equal to the received ratchet index, thereby obtaining the keys necessary to decrypt the early data.

When a DH handshake occurs during the resumption handshake (i.e., a `key_share` extension is included by both parties), the shared DH secret is used to derive all subsequent

secrets for a TLS session as specified in the key schedule [1]. The TLS key schedule is included in Figure 3, with rTLS additions marked in red. This figure is an adaption of the one included in RFC 8446 [1]; for details on the key schedule itself, we refer the reader to the RFC. If a resumption secret already exists (e.g., because this is not the first session resumption), then the derivation will depend on both the existing resumption secret and the DH shared secret. This produces a new master secret, which (as per the key schedule) eventually generates a new resumption master secret, as can be seen by following the arrows in Figure 3. Whenever a DH handshake occurs, the ratchet index *must* be reset to 0, as the inner ratchets will be completely reset. Additionally, this makes it harder for a Man In The Middle (MITM) adversary to replace the client's `shared_key` field with its own parameters, as it will also need to know the existing resumption secret, implying it would need to have access to either client or server already. Note that, when no DH handshake is performed, the ratchet Root Key is not updated at all. Instead, the Chain key feeds into itself (a ratchet step) and into the Early Secret.

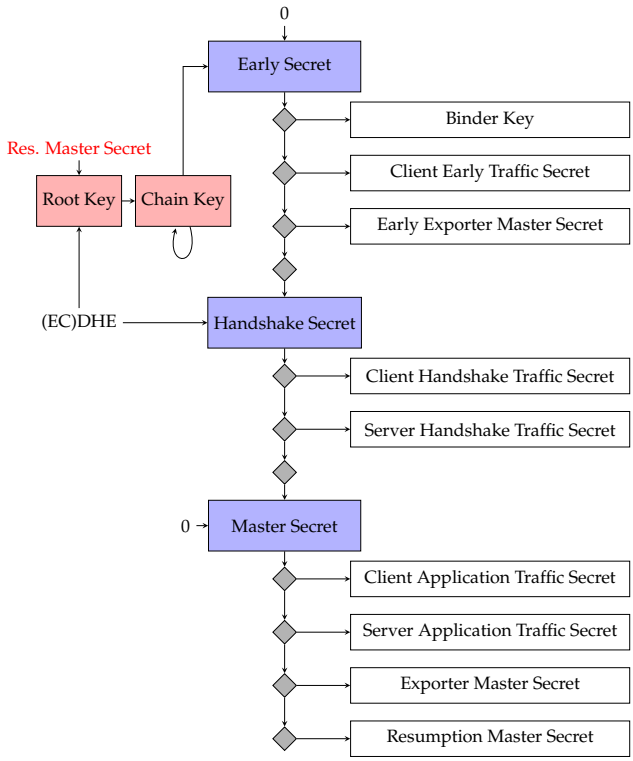



**Figure 3.** The rTLS key schedule. Red indicates added KDF instances. Blue indicates a default TLS HKDF instance. Grey diamonds indicate applications of the KDF function to produce a key.

Since we expect that, for virtually every scenario, one will want to reset the ratchets well before 255 communication attempts have been made, we only reserve 1 byte for the ratchet index. Additionally, when the ratchet index hits 255, we require both parties to delete their PSK and negotiate a new PSK with a standard handshake.

### *3.3. Double Ratchet Setup and Operation*

Next, we summarize the extra steps needed for both the initial and resumption handshakes in a step-by-step fashion.

#### 3.3.1. Initial Handshake

The initial handshake is largely unmodified, but some special steps have to be taken by both the client and the server.

1. **ID generation**: The server generates a globally unique connection ID. This ID is transmitted to the client in the `NewSessionTicket`, together with a DH public key that the Client can use to initialize future resumption handshakes.
2. **Symmetric ratchet initialization**: The client and server initialize the ratchet index variable to 0. The symmetric ratchet root key is the resumption master secret.
3. **Persistent state storage**: Both client and server store their state variables for anticipated session resumptions.

#### 3.3.2. Resumption

Below we describe the extra steps needed for a typical session resumption. A DH exchange may take place, but we do not consider that as an extra step—the TLS standard already accommodates for this.

**Client**

1. **Ratchet step**: The client ratchets its symmetric ratchet before the resumption master secret is used to derive any other secret. Thus, the early-data secret is derived from the ratcheted master secret.
2. **PSK exchange**: During the handshake, the client sends its ratchet index and connection ID to the server, as part of the `pre_shared_key`. If a DH exchange happens, the ClientHello includes a `key_share` structure, as well.

**Server**

1. **Access state:** The server receives a 0-RTT resumption, and after having verified the `pre_shared_key`'s HMAC field, finds the relevant state variables using the received connection ID as a key (e.g., in a hash map).
2. **Replay condition** The server ensures that $i_s < i_c$, where $i_s$ and $i_c$ are, respectively, the server received client ratchet indices for this connection.
3. **Ratchet step**: The server spins the symmetric ratchet $i_c - i_s$ times, where $i_c$ is the received ratchet index in `pre_shared_key`, and $i_s$ its own ratchet index. The early data encryption key is derived from the new state of the sym. ratchet.

**Both**

1. **Reset ratchet index**: If a DH exchange was performed during the resumption handshake, then the client and server reset their ratchet index to 0.
2. **Persistent state storage**: Both the client and server store their state variables for future session resumptions.

### *3.4. Ratchet State Variables*

This extension expects both the client and server to maintain some state for each connection. This state consists of the following data:

1. **Mapping**: a connection ID $\rightarrow$ ratchet mapping, to identify which ratchet belongs to which connection. We set the connection ID to be 4 bytes in size as an initial estimate. It can be increased if necessary.

2. **Ratchet Index**: To indicate the number of ratchet steps that occurred since the last DH exchange (1 byte).
3. **Private DH key**: Current private DH key, used to compute a DH secret from which a common root key can be derived (32 bytes).
4. **Remote public DH key**: Last received remote public DH key for deriving aforementioned secret (32 bytes). Additionally, the Client and Server are expected to keep track of the Resumption Master Secret. We do not list it with the above state variables as this is something that already comes with standard TLS, thus not being unique to rTLS.

## 4. Security Evaluation

In this section, we discuss and analyze the security properties of the rTLS protocol extension. We formally define the intruder model, and then present a formal model of the rTLS protocol extension itself. Various security properties are automatically verified by the OFMC software, thereby giving us high certainty that they hold for the protocol, as well.

### *4.1. Formal Verification*

Now, we present a formalization and verification in OFMC [7], a tool for formal verification of security protocols. It uses a symbolic Dolev-Yao-style model of cryptography, i.e., messages are represented in a term algebra where the algebraic properties of operators are represented (e.g., the properties of exponentiation needed for Diffie-Hellman). It formalizes a state-transition system through multi-set rewriting rules, and the main technique is a constraint-based representation of the intruder, dubbed the lazy intruder, which allow for verification without bounding the number of steps that the intruder can perform. However, the steps that the honest participants can perform needs to be bounded (or the tool will not terminate, in general). This choice of formal analysis software is motivated by the fact that most tools, such as ProVerif and Tamarin, run into problems with the ratchets since in an unbounded number of sessions, this creates structures for which the usual abstractions and bounding lemmata fail, but they do work in OFMC due to the bounds, allowing us to express the ratchets without problems. There are several input languages for OFMC, the native one being the AVISPA Intermediate Format IF [15] based on set-rewriting (similar to the input language of Tamarin). This can be considered kind-of a "protocol assembly language", i.e., it is hard to write by hand. The high-level languages available are Alice-and-Bob-style language AnB, but this language is too limited to express ratchets. There is also the AVISPA [16] High-Level Protocol Specification Language HLPSL [17] and its successor ASLan from the AVANTSSAR project [18]. Both languages would be suitable for our purposes, but the updating of local states that we have to perform make them not much more easy for the specification than IF, so we directly relied on IF for an initial formal verification [19]. We have, however, inspired by this work, developed a more high-level notation for protocols of this style and are currently working on a general compiler from this notation to IF to benefit in similar projects from it. We will use this high-level notation in the following presentation to explain our formal model.

#### 4.1.1. Intruder Model

We define two roles, Client and Server. Each role can, in principle, be instantiated arbitrarily often by any number of clients and servers. We need to limit this for OFMC to two sessions, albeit symbolic ones, meaning that the name of the client and the server is a variable where the intruder can determine who is playing. Thus we include at all kinds of two-session scenarios, e.g., an honest Alice as client with the intruder as server in parallel with a session between honest Alice and Bob as client and server. Note that the intruder can play any of the roles under his real name, where he has access to appropriate initial key material shared with a client or server; the payload messages exchanged in such a session are of course not secret. To allow the intruder to participate as a "normal" agent is essential to capture attacks where an intruder is, for instance, a dishonest server contacted by an

honest Alice, and uses part of the messages from this session to attack another session, as in the famous Needham-Schroeder PublicKey Protocol (NSPK) attack [20].

In the style of the Dolev-Yao intruder model, the intruder also controls the network, i.e., every message an honest agent sends goes to the intruder, and every message an honest agent receives comes from the intruder. The intruder can perform normal cryptographic operations with keys he knows, just as any other agent.

The starting point is that a Client and Server have successfully established a secure TLS 1.3 session in the past and, thus, share a resumption master secret; moreover, the Client has obtained a session ticket containing a DH public key, as well as connection ID from the Server.

#### 4.1.2. Resumption Handshake Model

Next, we present a detailed model of the resumption handshake protocol. This is effectively the standard TLS 1.3 0-RTT resumption protocol, with early data protected through a rotating (ratcheted) key.

First, every session of an agent is characterized by a number of state variables that are updated during the course of the session. These are shown in Table 2. Both share the same resumption master secret (`RES_MASTER_SECRET`) and connection ID (`CONN_ID`). In OFMC, we model this by a secret function `resMasterSecret(C,S,CONN_ID)` that, for a given client name, servername, and connection ID, returns a unique strong key; the intruder is given all keys where he is $C$ or $S$. The root key $RK$ is derived from `RES_MASTER_SECRET`. Note that `CONN_ID` is simply a unique identifier.

**Table 2.** The initial state for both client and server.

| **Client State** | | **Server State** | |
|---|---|---|---|
| **State Variable** | **Initial State** | **State Variable** | **Initial State** |
| `RES_MASTER_SECRET` | from TLS | `RES_MASTER_SECRET` | from TLS |
| $RK$ | ... | $RK$ | ... |
| `CONN_ID` | from TLS | `CONN_ID` | from TLS |
| *ServerDHsPub* | $g^X$ | *ClientDHsPub* | - |
| *currPrivate* | - | *currPrivate* | $X$ |
| *ClientCKs* | - | *ServerCKr* | - |
| *ClientNs* | 0 | *ServerNr* | 0 |
| *CHR* | - | *CHR* | - |
| | | *SHR* | - |
| *Step* | 0 | *Step* | 0 |

Both the Client and Server store the latest DH public key received from the other side as `ServerDHsPub` and `ClientDHsPub`, respectively. They also store their own latest DH private key `currPrivate`. Because the Client has received a DH public key from the server during the first session in a NewSessionTicket, we assume that `ServerDHsPub` and the Server's `currPrivate` are initially populated. In OFMC, we model the initial private key of the server again with a private function `secret_exponent(S,CONN_ID)` for the server (known to the intruder whenever $S = i$).

Finally, the Client needs to store its sending chain and the server needs to store its receiving chain. These consist of a chain key and a chain index, which are defined as `ClientCKs` and `ClientNs` for the Client, and `ServerCKs` and `ServerNs` for the server. However, these do not need to be initialized at the start. The Client will compute its private key before transmitting the first resumption message.

Similar to the session bounding in OFMC, we also need to bound how many ratchet turns each agent can make in each session. Again we have to limit ourselves to a quite low bound of 2 repetitions, but this should cover all likely scenarios. As a modeling trick, we just initialize both counters with 2, and, in each resumption, we decrease until it is 0.

4.1.3. Step 1: ClientHello

Now, we use the state variables to construct a detailed description of an execution of the 0-RTT protocol. Note that all steps come in two variants: with a new DH key exchange and without. In the OFMC implementation, the client can choose which variant. We describe only the variant in detail that does the DH key exchange, and we only mention the difference when no DH key exchange is done.

If a DH key exchange is to be included in the handshake, then, the first action is that the Client generates a new private key, as well as a shared DH key, together with the Server's DH public key. When the Client has computed this DH secret, it passes the key into its inner ratchet, by applying the KDF function on the DH secret combined with a root key $RK$, and, finally, obtains the `ClientCKs`:

```
new currPrivate
RK         := bkdf(RK,exp(ServerDHsPub,currPrivate))
ClientCKs  := kdf(RK)
```

where we use $kdf$ and $bkdf$ to model the corresponding key derivation functions.

Now, we can describe the initial message sent from a Client. First, it spins its ratchet and increases `ClientNs` by one (i.e., actually in the OFMC model, decreases, if not yet zero). The key generated through this is used as input material for the Early Secret in the TLS key schedule. We focus only on the relevant parts of a resumption `ClientHello` message, specifically, the early data itself (`MOUT`, TLS session ID, client randomness and the relevant resumption parameters. The keys $K_1$, the `client_early_traffic_secret` and $K_2$, the `binder_key` are derived from the Early traffic secret as can be seen in Figure 3. At this point, it is important to note that since the Client can choose to include an optional `key_share` extension (DH handshake) in the ClientHello, the inclusion of `ClientDHsPub` in the resumption handshake is also optional. The early data is encrypted with `client_early_traffic_secret`. Additionally, the plaintext data is integrity protected through a MAC with key `binder_key`. Both keys are derived from the master key conform the TLS standard:

```
let MSG1=step0(ClientNs,exp(g,currPrivate),CHR)
let K1=hkdf(ClientCKs,MSG1)
let K2=hkdf(ClientCKs,pair(exp(ServerDHsPub,currPrivate),
 pair(C,S)))
send(step1(scrypt(K2,MOUT),hmac(K1,MSG1),MSG1))
```

Here, $step_0$ and $step_1$ are message formats that represent how the cleartext data is serialized (i.e., every agent, including the intruder, can compose and decompose such messages without any keys). $hkdf$ is another key-derivation function, $pair$ stands for pure string concatenation, and $scrypt(k, m)$ stands for symmetric encryption of message $m$ with key $k$, and $hmac(k, m)$ stands for a hash-mac with key $k$ of message $m$.

When the Server receives a ClientHello with early data indication, it first has to spin its inner ratchet to derive an `early_secret` identical to that of the Client. Included in this step is the incrementing of `ServerNr`. The Server can then derive the keys necessary to authenticate and decrypt the received early data. After this point, the Server proceeds differently based on whether the `key_share` extension was included by the Client. If the extension was not included, the Server continues using the current chain for future resumptions and can simply continue the current handshake as usual. If the extension was included, the Server will have to spin its DH ratchet, as well, which, in turn, leads to an update of the Server's receiving chain root key. Note that this new DH secret is not just for future sessions and is already used in the remainder of this handshake as it normally would be in a TLS session, as we explain in the next paragraph.

In the high-level notation, we have:

```
receive(step1(SM2,HM1,M1))
try step0(SN,ClientDHsPub,CHR)==M1
```

```
let DH      = exp(ClientDHsPub,currPrivate)
RK         := bkdf(RK,DH)
ServerCKr  := kdf(RK)

let K1=hkdf(ServerCKr,M1)
try HM1==hmac(K1,M1)

let K2=hkdf(ServerCKr,pair(DH,pair(C,S)))
try MIN==dscrypt(K2,SM2)
```

Note that the `try` is used to describe operations that might fail, such as trying to decrypt, parse, or check for equality. When it fails, the agent simply does aborts the transaction and rolls back to the state before the transaction. In particular, the first `try` in the above code snippet parses the received message `M1` as the `step0` format, extracting the three components of the message. The next `try` is checking that the received hmac `HM1` is the same as constructing `hmac(K1,M1)`, and the last `try` is trying to decrypt the message `SM2`. Note that we assume here symmetric encryption with MACs that tells us if decryption succeeded. Observe the contrast to the `let x=t` command, which simply means replacing all further occurrences of `x` with `t`, and the `x:=t` command, which means that the state variable `x` is set to `t`.

#### 4.1.4. Step 2: ServerHello

Next, the Server will reply with a `ServerHello` message. If the Client included a `key_share` extension, then the server will reply with its newly generated DH public key from the previous step. Before the response can be sent, the Server has to compute all the remaining keys from the TLS key schedule. This starts with computing the `handshake_secret`. The KDF function for this secret takes two inputs, one being the hash product of the previous phase in the key schedule, and another being fresh Input Key Material (IKM). If a DH handshake occurred, then the resulting DH secret should be used as IKM here. If no DH handshake occurred, the IKM is simply set to 0. The `ServerHello` response itself includes a number of fields which are not relevant for our verification, so we leave them out. We do include `EncryptedExtensions` ($EE$) as a representative message payload and the contents of the `Finished` message type, which has a field `verify_data`, containing an HMAC of the handshake context. This HMAC protects the integrity of `ServerDHsPub` and `Server_rand`, as well; therefore, we add these to the encrypted payload, while leaving other parts out to keep the model concise. We can do this, as the `HMAC` key is directly derived from the `server_handshake_traffic_secret`. We include `Server_rand`, as this is 32 bytes or randomness that is used for various cryptographic purposes and acts as a nonce. Finally, the Server has the opportunity to already send application data (`App_Data`) with its response.

Different parts of the transmission are encrypted with different keys derived from the master secret conform the TLS standard. The remainder of the handshake, i.e., most of the `ServerHello` message is encrypted with the `server_handshake_traffic_secret`. If the Server chooses to include a response payload, then this optional response can already be encrypted with the `server_application_traffic_secret`.

```
new currPrivate
new SHR
let DH          = exp(ClientDHsPub,DHs)
RK             := bkdf(RK,DH)
ServerCKr      := kdf(RK)

let K2=serverK(hkdf(DH,pair(ServerCKr,pair(CHR,pair(C,S))))
)
let MSG2=scrypt(K2,pair(exp(g,DHs),SHR))
```

```
let K1=serverK(hkdf(DH,pair(ServerCKr,pair(SHR,pair(CHR,
pair(C,S))))))
let MSG1=scrypt(K1,MOUT)
send(step2(MSG1,MSG2,SHR,exp(g,DHs)))
```

When the Client receives the Server's `ServerHello`, it first has to continue with its own execution of the TLS key schedule. If the Client initiated with a new DH public key and, thus, a `key_share` extension, the server replied with a fresh DH public key in its own `key_share`. This is then used by the Client as input for the handshake secret identically to how the server processed the DH secret. With this, the Client can continue the TLS key schedule until all keys are derived. Note that, for both the Server and Client, the newly computed Resumption Master Secret is assigned to the inner chain's root key, but not necessarily included in the current chain; if no DH handshake was included, the inner chain is not reset. This does not matter, as no new entropy was introduced during the handshake either way. As is evident from the description of the operations of the rTLS resumption process given in this section, the optional DH exchanges feed into the TLS keyschedule and provide new entropy that gets propagated through to the inner chains and as a result future executions of the key schedule.

```
receive(step2(M1,M2,SHR,ExpgDHs))
let DH=exp(ExpgDHs,currPrivate)

ServerDHsPub      :=ExpgDHs
RK        :=bkdf(RK,DH)
ClientCKs:=kdf(RK)
ClientNs  :=s(ClientNs)

let K2=serverK(hkdf(DH,pair(ClientCKs,pair(CHR,pair(C,S))))
)
try pair(ExpgDHs,SHR)==dscrypt(K2,M2)

let K1=serverK(hkdf(DH,pair(ClientCKs,pair(SHR,pair(CHR,
pair(C,S))))))
try MIN==dscrypt(K1,M1)
```

#### 4.1.5. Step 3: Finished

The Client finishes the 0-RTT handshake with an `EndOfEarlyData` message and a `Finished` message. The `EndOfEarlyData` message is simply an indicator that the Client has no more early data to transmit and that all future data will be encrypted with the `client_application_traffic_secret`.

```
let TMP=pair(pair(C,S),pair(ExpgDHs,pair(CHR,SHR)))
let K3=clientK(hkdf(DH,pair(ClientCKs,TMP)))
send (scrypt(K3,MOUT))
```

#### 4.1.6. Verification

We verify a number of security goals, the first of which is secrecy. We want the early data, i.e., `MOUT`/`MIN` payloads, to be secret between Client and Server. The second security goal we verify is injective agreement [21]. This means that, when an honest party $B$ receives a payload message apparently from $A$, then, either $A$ is the intruder under his real name (no authentication guarantees) or $A$ indeed sent that payload message for $B$ (and they agree on all roles). Moreover, this is injective in the sense that $B$ does not accept the same payload more often than it was sent by $A$, so there is no replay.

Using OFMC, we verify the described properties to hold for the rTLS resumption protocol. Due to an exponential increase of the search spaces with the number of sessions and resumptions, we bounded the number of sessions to 2, and the number of resumptions in each session also to 2. Note, however, that we have here symbolic sessions, i.e., they can

be arbitrarily instantiated, including with the intruder as a client or server. Moreover, in each session and resumption, the client can decide to either perform a new DH key or not. We also extensively tested the specification, namely that all expected steps could be taken, in particular that honest agents can communicate, and the intruder can play each of its roles under his real name as a normal participant.

OFMC reported that no attacks were found in any runs which gives a high assurance that the rTLS session resumption protocol provides secrecy and injective agreement: While this is only proved for 2 sessions and with 2 resumptions each, it seems unlikely that further sessions and resumptions would allow for additional attacks because of the symmetry of all further repetitions.

Finally, we want to look at the so-called selfie attack [22]: this is an attack that works on some pre-shared-key deployments of TLS 1.3, where a client $C$ and server $S$ use the same pre-shared key $psk(C,S) = psk(S,C)$ in both directions of communication, allowing for reflection attacks. Similarly, if we allow in our rTLS model:

$$RES\_MASTER\_SECRET(C,S,CONN\_ID) = RES\_MASTER\_SECRET(S,C,CONN\_ID),$$

then we still do not get a selfie-attack because the setup of the Diffie-Hellman ratchet is different for client and server role. This is, however, looking only at the initial state of the resumption handshake rTLS, not at the preceding steps of the original TLS. This means that, if the setup of TLS is such that it does not allow for a selfie attack, then, by construction, rTLS cannot induce a selfie attack either.

## 5. Performance Evaluation

In this section, we present numeric estimates of the performance of rTLS, with as performance indicators traffic overhead and storage overhead. The numerical data is based on the estimated data structure size of the state variables and TLS message structures as they are defined in the TLS standard.

### *5.1. Traffic Overhead estimation*

#### 5.1.1. Initial Handshake

The rTLS initial handshake does not differ in traffic overhead from a normal TLS handshake, since the only change defines an extra value for an enumerated field, which is (`psk_key_exchange_modes`). After finishing the handshake, the server transmits a `NewSessionTicket` message to the client. While technically not part of the initial handshake, we consider it as such in this context; without it, resumption would not be possible. The structure and size of a minimal `NewSessionTicket` message are displayed in Table 3. Here, $|X|$ indicates the size in bytes of element $X$. The client does not need to send any reply to this message. The ticket field itself has to contain the connection identifier, as well as a public DH key that the client can use for the first resumption; so, we set the size of the ticket field to $4 + 32$ bytes, and we include a 32 byte nonce, as well. We do not need to explicitly include a ratchet index here, as it can be initialized to 0 by both parties. Therefore, compared to no session resumption at all, minimal overhead is $14 + 36 + 32 = 72$ bytes. Compared to a session ticket in standard TLS 1.3, which, in OpenSSL, is typically around 528 bytes, this is a significant improvement of 86 percent.

#### 5.1.2. Resumption Handshake

It is important to reduce traffic overhead for the resumption handshake as much as possible, since this will typically be performed much more often than an initial handshake. The minimal cost for any resumption handshake consists of boilerplate parts of the handshake that cannot be eliminated without rigorous change to the TLS protocol. In the following, we write client and server as $c$ and $s$, respectively. We map symbols to every message element in the resumption handshake in Table 4, where $x$ can be either $c$ or $s$ to indicate the message sender.

**Table 3.** The message structure and size of a minimal `NewSessionTicket` message. Here, $|T|$ refers to the ticket length, and $|N|$ to the size of the nonce.

| | Size (bytes) | Field Name |
|---|---|---|
| | 4 | ticket_lifetime |
| | 4 | ticket_age_add; |
| | $\|N\|$ | ticket_nonce |
| | $\|T\|$ | ticket |
| | 2 | extensions length |
| | 4 | Early data extension |
| **Total** | $14 + \|T\| + \|N\|$ | |

**Table 4.** Symbol definitions for message elements, where $x \in \{c, s\}$ refers to the message sender (client resp. server).

| Symbol | Description |
|---|---|
| $H_x$ | (Client or Server) `Hello` |
| $ed_x$ | `early_data` |
| $D_x$ | Application data |
| $pex$ | `psk_key_exchange_modes` |
| $psk_x$ | `pre_shared_key` |
| $ks_x$ | `key_share` |
| $ee$ | `EncryptedExtensions` |
| $eed$ | `EndOfEarlyData` |
| $f$ | `Finished` |
| $R$ | Record Layer headers |

We define the minimal traffic overhead cost $C$ of any 0-RTT resumption handshake as:

$$C = 3|R| + |H_c| + |H_s| + |ed_c| + |pex| + |ee| + 2|f| + |eed|. \quad (1)$$

This cost is not a fixed number of bytes but, rather, is not negotiable; any PSK extension will have to include these elements, and their size is independent of the actual PSK mode. The total cost of a minimal resumption handshake is then $C + |psk_c| + |psk_s|$. Note that $ks_c$ and $ks_s$ are not required for a minimal handshake. Conforming to the standard, $psk_s$ is defined as a 2-byte value representing an identity index in $psk_c$ and is wrapped in a 4-byte TLS extension structure. However, $ks_c$ is more complex, and we write the full layout in Table 5. Note that the term "identifier" here refers in the standard to the ticket field itself, but we use it to transmit a concatenation of the connection identifier and ratchet index. Because we only send one identity and binder, The size of $psk_c$ becomes $|psk_c| = 15 + \alpha + \beta$, where $\alpha$ denotes the size of the `identity` field, and $\beta$ the size of the binder HMAC. The identifier field $PSKID$ can be written as $PSKID = ID||i$, where $ID$ is the identifier received in the session ticket during the initial handshake, and $i$ is the symmetric KDF chain index. Now, $|PSKID| = |ID| + 1 = 5$. The exact value of $\beta$ depends on the chosen HMAC function, which is usually either Secure Hash Algorithm (SHA)-256 (32 bytes) or SHA-384 (48 bytes). The complete traffic cost $c_1$ for session resumption can, thus, be written as $c = |psk_s| + |psk_c| + C = 26 + \beta + C$, and it is $58 + C$ if SHA-256 is chosen.

When a DH exchange is included, we will have to add the size of the $ks_c$ and $ks_s$ elements. The size of $ks_c$ is of variable length depending on the number of supported DH groups the client advertises. Each key share entry takes up $4 + l$ bytes, where $l$ is the size of the supported group. The smallest supported group is X25519 with a 32-byte field, while the largest is P-521 with 132 bytes. $ks_c$ also reserves 2 bytes to denote the number of listed groups. If we only transmit one group, the size is, therefore, $ks_c = 6 + l$. The server replies with a single key share entry; thus, $ks_s = 4 + l$. As with any TLS extension, both $ks_s$

and $ks_c$ are wrapped in an extension structure with a 4-byte type field. The total cost of a resumption with DH exchange is, thus, $c_2 = c_1 + |ks_c| + |ks_s| = c_1 + 18 + 2l$.

**Table 5.** Layout of the `pre_shared_key` structure and its sub-structures, when sent by a client.

| Size | Field Name | Description |
|---|---|---|
| | **pre_shared_key** | |
| 2 | `extension_type` | Extension type |
| 2 | `extension_data` | Size of the extension |
| 2 | `PSKIdentities_length` | Nr. of PSK identities |
| | `identities` | `PSKIdentity` values |
| 2 | `binders_length` | Nr. of PSK binders |
| | `binders` | `PSKBinder` values |
| | **PSKIdentity** | |
| 2 | `identity length` | Size of identity field |
| $\alpha$ | `identity` | value of this identity |
| 4 | `obfuscated_ticket_age` | ticket age (see Reference [1]) |
| | **PSKBinder** | |
| 1 | `binder length` | size of the binder value |
| $\beta$ | `binder` | HMAC value (see Reference [1]) |

If we take into account a `key_share` every $n$ messages, we arrive at the final equation for the total average cost $c_t$:

$$c_t = \left\{ \begin{array}{ll} 26 + \beta + C & \text{for } n = 0 \\ 26 + \beta + \frac{18+2l}{n} + C & \text{for } n > 0 \end{array} \right\}, \quad (2)$$

where $\beta$ is the hash digest size, $l$ the elliptic curve coordinate length, $n$ the DH handshake rate, and $C$ the minimal cost. Figure 4 shows the average overhead (i.e., without $C$) versus the key exchange period, for various common cipher suites.

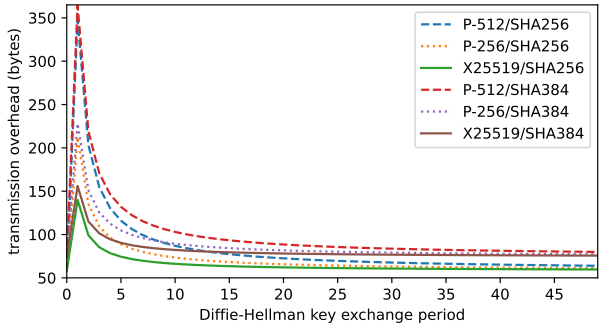



**Figure 4.** Average transmission overhead versus DH key exchange period.

Giving an exact value for $C$ is somewhat difficult: multiple fields in $H_c$, $H_s$, and *ee* can vary a lot in length, depending on the supported cipher suites and provided extensions among other things. Instead, we count the minimum size for these fields as they are defined in the standard, thereby giving a lower bound for $C$. Note that, in practice, a handshake with so few extensions is not useful for overhead minimization, as more round-trips will be needed to establish necessary parameters, such as the cipher suite. Moreover, it leaves out extensions meant to increase overall security. Minimal sizes, including all headers, for

$H_c$ and $H_s$ are 50 and 48 bytes, respectively. $ed_c$ and *eed* both require 2 and 4 bytes. *pex* is, at least, $3 + m$ bytes in size, where $m$ is the number of supported modes (at least 1). *ee* is, at least, 6 bytes in size but may vary a lot, depending on the supported extensions. The length of $f$ is determined by the chosen hash function. The record layer headers are 5 bytes in size. With one PSK key exchange mode and the SHA-256 hash function, the total cost of $C$ is then, at least, 193 bytes. Therefore, the lower bound on transmission overhead of a resumption handshake with our extension is 251 bytes without, or 333 bytes with a key exchange. If we include several extensions for a more realistic minimal handshake, we can expect the cost to be between 400 and 600 bytes.

### *5.2. Storage Overhead Estimation*

Both the client and server need to store some state variables in between sessions. This differs from the standard session resumption protocol where only the client stores the PSK. The client needs to securely store the secret KDF key (depends on digest size), as well as its connection ID (4 bytes) and the ratchet index (1 byte). Additionally, the client needs to keep track of its current DH private key and the last received DH public key from the server, the size of these depends on the chosen group. Thus, the client needs to store 101 bytes if SHA-256 and X25519 are used.

The server needs to store the same amount of state, but for every client that it shares a ratchet for resumption with. This can be done through, e.g., a hash map using the connection ID as a key, and a structure containing the other state variables as value. If state is being kept for the maximum amount of clients of $2^{32}$ (with a 4-byte connection ID), this amounts to roughly 433 GB worth of data. When there is a large set of clients connecting to the server and, thus, a large amount of state variables, one should be mindful of access times and pick data structures that minimize access time, such as hash maps, to provide some protection against denial of service attacks.

### *5.3. Overhead comparison with TLS 1.3*

Based on measurements performed on OpenSSL [6], a standard PSK in TLS 1.3 adds 571 and 603 bytes of overhead, respectively, when SHA-256 SHA-384 is used. In Table 6, we compare the overhead of rTLS for various key exchange periods $n$ to that of a standard TLS 1.3 PSK. We use a higher value of 408 for $C$, obtained from handshake measurements in OpenSSL, which includes a minimal number of extensions by default, and acts as an indicative value that represents a lightweight use case. In this table, the values are computed using the smallest allowed hash function (SHA-256) and curve (X25519). As can be seen, a rTLS PSK requires only roughly 11% of the traffic overhead compared to a standard TLS PSK and can be expected to reduce the total amount of transmitted data roughly by half.

**Table 6.** A comparison between rTLS session resumption and OpenSSL standard session resumption.

| | **Indicative Lightweight Use ($C = 408$)** | |
|---|---|---|
| **Scenario** | **Avg. Overhead (b)** | **Avg. Total Size (b)** |
| rTLS, $n = 0$ | 58 | 466 |
| rTLS, $n = 1$ | 108 | 516 |
| rTLS, $n = 10$ | 63 | 471 |
| Standard TLS 1.3 | 571 | 979 |

## 6. Related Work

There exist ample communication security protocols aimed at embedded devices [23]. We look at the TLS protocol and its variants, specifically those that are relevant to the usage of this protocol in embedded environments. We also briefly look at QUIC.

Initially developed for Web security, TLS is now gaining traction in the IoT world, partly due to widely available libraries and broad support in software relevant to IoT. For

example, many Message Queuing Telemetry Transport (MQTT) brokers support TLS as a security layer.

While this is fine for most devices (mostly upwards from class 1 in the IETF classification [24]), it becomes problematic when working with class 0 or low-end class 1 devices, as they do not possess the capability to maintain TLS connections or can simply not afford it due to resource constraints (e.g., due to a power budget). To address this, several optimizations have been proposed over the years. One of the first was Sizzle [25], which is an implementation of the Secure Socket Layer (SSL) protocol, capable of running on extremely constrained devices with only tens of kilobytes of memory. While the authors showed that heavyweight cryptographic operations required for the protocol to function were certainly possible on heavily constrained devices, they did not attempt to reduce the amount of transmitted data.

Datagram Transport Layer Security (DTLS) [26] modifies the TLS protocol to work over User Datagram Protocol (UDP), while retaining most of the security guarantees provided by TLS. This reduces the data overhead and latency somewhat. Recently, the DTLS 1.3 draft [27] was approved by the IETF. This revision brings 0-RTT and other TLS 1.3 improvements to DTLS. There exist multiple open-source implementations [28], and several works exist detailing extremely lightweight implementations [29,30]. In these works, lightweight mostly pertains to computational and memory cost, while transmission overhead is either not addressed or addressed to a much lesser degree. Other approaches have been taken, as well, such as Reference [31], compressing DTLS messages to fit into 6LowPAN frames. Recently, a performance comparison of TLS 1.3 and DTLS 1.3 on lightweight IoT devices was published [32], showing that, while both TLS and DTLS 1.3 add suffer from larger overhead in terms of memory usage and transmission overhead, these are within bounds for these protocols to be used on devices that can already run the 1.2 version. Additionally, the authors state that there is room for optimizations in software to further reduce the overhead. In Reference [33], a DTLS fast session resumption mechanism is proposed, making use of free UDP ports on the server-side. However, the proposed protocol does not address forward security and provides no analysis of its security claims.

While DTLS has less bandwidth overhead than TLS, it is still not ideal for lightweight scenarios with message proxies (e.g., brokers, such as in MQTT). To address this, the recently standardized application-layer Object Security for Constrained RESTful Environments (OSCORE) protocol aims to enable selective encryption of parts of the Constrained Application Protocol (CoAP) protocol. Gunnarsson et al. [34] show that this provides a slight performance improvement over the default DTLS security option. Due to OSCORE's selective encryption approach, it can provide end-to-end encryption in situations where messages are relayed through proxies, whereas TLS-based protocols have to setup separate secure channels between each proxy. However, when no proxies are needed, TLS-based protocols might offer better performance especially when 0-RTT is taken into account.

Several extensions for TLS have been proposed that also bring the potential to lower message overhead. The TLS Cached Info specification [35] allows clients to store server certificates and certificate requests, making it possible to leave these out in future handshakes. The TLS Raw Public Key extension [36] allows clients and servers to authenticate each other through public keys, instead of X.509 certificates. This can significantly reduce the handshake size. This method does require an out-of-band means of verifying public keys, which might very well be possible in a controlled environment, such as a factory. Another promising adaptation of TLS that might lower the size overhead of TLS significantly is the **cTLS!** (**cTLS!**) IETF draft [37]. In this draft, the authors propose optimizing the TLS protocol for size by eliminating redundancy where possible and making aggressive use of space-optimization techniques, such as variable-length integers. The result is isomorphic to TLS, but not interoperable.

Additionally, in our previous work [5], we introduced rTLS, a TLS 1.3 protocol extension that focuses specifically on the 0-RTT session resumption protocol, with the goal of

making it more usable for the IoT. In our original work, we presented the protocol and included numerical estimates on its performance but did not include a thorough analysis of its security properties. In this work, we extended upon that and present a formal security analysis, as well as some fixes to the protocol that were overlooked in the original work. The rTLS extension is compatible with the aforementioned cTLS draft, as well as other TLS extensions. For DTLS, it is very likely that some adjustments are necessary as the DTLS resumption protocol is slightly different.

DTLS is also proposed as the default mechanism to secure connections in the QUIC protocol, a network protocol building on UDP that provides advanced features, such as multiplexing and authenticated encryption of its data by default.

Session resumption in TLS 1.3 has been subject to debate, as it is vulnerable to replay attacks and provides no forward secrecy [1]. While, for a Web environment, there exists some justification for these design choices, for an IoT environment where short conversations with short messages are the norm, this is less than ideal, as it effectively removes the possibility to optimize overhead through use of the session resumption protocol. None of the extensions discussed in this section address session resumption, which means that this is an open issue we think has significant potential for minimizing protocol overhead, when designed carefully.

At the time of writing, National Institute of Standards and Technology (NIST) is hosting an ongoing competition for lightweight cryptographic primitives [38]. Many of the candidates specifically target very short messages. Once the candidates have received sufficient cryptanalytic attention, these can become valuable tools in future lightweight communication protocols, as well as potentially helping protocols, such as TLS adapt to constrained devices.

In Reference [39], Hall-Andersen et al. acknowledge the complexity of TLS and propose nQUIC as a lightweight, less complex alternative to QUIC's default TLS configuration. Their experiments show a significant reduction in bandwidth compared to TLS.

## 7. Conclusions

In this work, we extended upon an IoT-friendly and standard-compliant adaption of the TLS 1.3 0-RTT session resumption protocol. We first argued that, in order to be applicable to IoT, replay resistance is a necessary property, as lightweight sensor devices are much more likely to transmit data that will change server state.

Building from the observation that, in IoT scenarios, the group of possible clients for a server changes relatively slowly and is typically much smaller than possible clients for a Web server, we argued that it is reasonable to require a server to keep some state variables for each of its clients. We then took inspiration from the Double Ratchet algorithm to design a 0-RTT resumption protocol that fits neatly into the existing message structure, and makes use of existing functionality where possible. In our extension, the PSK utilizes a ratchet construction, which provides replay protection, as well as forward secrecy and break-in resilience to early data transmitted in a 0-RTT handshake. The introduction of these properties in the 0-RTT sub-protocol is a step toward making TLS suitable for IoT scenarios.

We estimated a lower bound of 193 bytes on traffic overhead for any 0-RTT resumption protocol in TLS 1.3 and then showed that a resumption handshake with our protocol would result in around 466–516 transmitted bytes, depending on the chosen DH key exchange period. Compared to the standard session resumption transmission size of roughly 979 bytes, this is a significant improvement.

Extending our previous work, the protocol received minor updates relating to what state should be kept. Additionally, we improved the presentation of the protocol, including a more detailed description of how the TLS key schedule is affected. These minor changes are also propagated into the performance evaluation estimates, affecting mainly the storage overhead estimates. We also added a new section detailing a formal security analysis of

the protocol in the Dolev-Yao model. The results of this analysis give high assurance that the protocol provides secrecy, as well as security against replay attacks.

In future work, we aim to further reduce the transmission overhead by exploring different opportunities, such as replacing the original message structure for resumption altogether, thereby reducing the fixed cost. Moreover, we are currently testing a proof-of-concept implementation to support the overhead estimations with empirical results.

**Author Contributions:** Conceptualization, K.T.; Methodology, K.T. and X.F.; Validation, K.T. and S.M.; Formal Analysis, S.M. and A.L.; Writing—original draft preparation, K.T. and S.M.; writing—review and editing, K.T., S.M., A.L., X.F. and N.D.; supervision: X.F. and N.D. All authors have read and agreed to the published version of the manuscript.

**Funding:** This is work was part of the Fog Computing for Robotics and Industrial Automation (FORA) European Training Network (ETN) funded by the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No 764785.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Conflicts of Interest:** The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

## References

1. Rescorla, E. The Transport Layer Security (TLS) Protocol Version 1.3. Available online: https://rfc-editor.org/rfc/rfc8446.txt (accessed on 9 August 2021).
2. AT&T. LTE-M and NB-IoT. Available online: https://www.business.att.com/products/lpwa.html (accessed on 9 August 2021).
3. Verizon. Verizon Thingspace. Available online: https://thingspace.verizon.com/services/connectivity.html (accessed on 9 August 2021).
4. Hologram. Hologram Pricing. Available online: https://hologram.io/pricing/ (accessed on 9 August 2021).
5. Tange, K.; Howard, D.; Shanahan, T.; Pepe, S.; Fafoutis, X.; Dragoni, N. rTLS: Lightweight TLS Session Resumption for Constrained IoT Devices. In Proceedings of the 22nd International Conference on Information and Communications Security, Copenhagen, Denmark, 24–27 August 2020; pp. 243–258, doi:10.1007/978-3-030-61078-4_14.
6. OpenSSL Software Foundation. OpenSSL. Available online: https://www.openssl.org (accessed on 9 August 2021).
7. Basin, D.A.; Mödersheim, S.; Viganò, L. OFMC: A symbolic model checker for security protocols. *Int. J. Inf. Sec.* **2005**, *4*, 181–208.
8. Perrin, T.; Marlinspike, M. The Double Ratchet Algorithm. Available online: https://www.signal.org/docs/specifications/doubleratchet/doubleratchet.pdf (accessed on 9 August 2021).
9. Rescorla, E.; Dierks, T. The Transport Layer Security (TLS) Protocol Version 1.2. Available online: https://rfc-editor.org/rfc/rfc5246.txt (accessed on 9 August 2021).
10. Salowey, J.; Zhou, H.; Eronen, P.; Tschofenig, H. Transport Layer Security (TLS) Session Resumption without Server-Side State. Available online: https://rfc-editor.org/rfc/rfc4507.txt (accessed on 9 August 2021).
11. WolfSSL. WolfSSL Embedded SSL/TLS Library. Available online: https://www.wolfssl.com/ (accessed on 21 September 2021).
12. Systems, O. Signal. Available online: https://www.signal.org (accessed on 9 August 2021).
13. WhatsApp. WhatsApp Encryption Overview. Available online: https://www.whatsapp.com/security/WhatsApp-Security-Whitepaper.pdf (accessed on 9 August 2021).
14. Cohn-Gordon, K.; Cremers, C.; Dowling, B.; Garratt, L.; Stebila, D. A Formal Security Analysis of the Signal Messaging Protocol. In Proceedings of the 2017 IEEE European Symposium on Security and Privacy (EuroS&P), Paris, France, 26–28 April 2017; pp. 451–466, doi:10.1109/EuroSP.2017.27.
15. Armando, A.; Basin, D.A.; Boichut, Y.; Chevalier, Y.; Compagna, L.; Cuéllar, J.; Drielsma, P.H.; Héam, P.; Kouchnarenko, O.; Mantovani, J.; et al. The AVISPA Tool for the Automated Validation of Internet Security Protocols and Applications. In *Proceedings of the Computer Aided Verification, 17th International Conference, CAV 2005, Edinburgh, UK, 6–10 July 2005*; Etessami, K.; Rajamani, S.K., Eds.; Springer: Berlin/Heidelberg, Germany, 2005; Volume 3576, pp. 281–285.
16. European Union. The AVISPA Project. Available online: http://www.avispa-project.org/main.html (accessed on 9 August 2021).
17. Yannick, C.; Compagna, L.; Cuellar, J.; Drielsma, P.; Mantovani, J.; S. Mödersheim, A.L.V. A High Level Protocol Specification Language for Industrial Security-Sensitive Protocols. In Proceedings of the SAPS'04, Linz, Austria, 20–24 September 2004.
18. Viganò, L. Automated validation of trust and security of service-oriented architectures with the AVANTSSAR platform. In Proceedings of the 2012 International Conference on High Performance Computing Simulation (HPCS), Madrid, Spain, 2–6 July 2012; pp. 444–447, doi:10.1109/HPCSim.2012.6266956.
19. Lalos, A. A Formal Library of IoT Protocols. 2021. http://findit.dtu.dk (accessed on 9 August 2021).

20. Lowe, G. An attack on the Needham-Schroeder public-key authentication protocol. *Inf. Process. Lett.* **1995**, *56*, 131–133. doi:10.1016/0020-0190(95)00144-2.
21. Lowe, G. A hierarchy of authentication specifications. In Proceedings of the 10th Computer Security Foundations Workshop, Rockport, MA, USA, 10–12 June 1997; pp. 31–43.
22. Lowe, G. Selfie: reflections on TLS 1.3 with PSK. *J. Cryptol.* **2021**, *34*, 27, doi:10.1007/s00145-021-09387-y.
23. Ferrag, M.A.; Maglaras, L.A.; Janicke, H.; Jiang, J.; Shu, L. Authentication Protocols for Internet of Things: A Comprehensive Survey. *Secur. Commun. Netw.* **2017**, *2017*, 6562953, doi:10.1155/2017/6562953.
24. Bormann, C.; Ersue, M.; Keränen, A. Terminology for Constrained-Node Networks. Available online: https://rfc-editor.org/rfc/rfc7228.txt (accessed on 9 August 2021).
25. Gupta, V.; Wurm, M.; Zhu, Y.; Millard, M.; Fung, S.; Gura, N.; Eberle, H.; Shantz, S.C. Sizzle: A Standards-Based End-to-End Security Architecture for the Embedded Internet. *Pervasive Mob. Comput.* **2005**, *1*, 425–445.
26. Rescorla, E.; Modadugu, N. Datagram Transport Layer Security. Available online: https://rfc-editor.org/rfc/rfc4347.txt (accessed on 9 August 2021).
27. Rescorla, E.; Tschofenig, H.; Modadugu, N. The Datagram Transport Layer Security (DTLS) Protocol Version 1.3. Available online: https://www.ietf.org/archive/id/draft-ietf-tls-dtls13-41.txt (accessed on 9 August 2021).
28. WolfSSL. TLS 1.3 Protocol Support. Available online: https://www.wolfssl.com/docs/tls13/ (accessed on 9 August 2021).
29. Bergmann, O.; Gerdes, S.; Bormann, C. Simple keys for simple smart objects. In Proceedings of the Workshop on Smart Object Security, Paris, France, 23 March 2012.
30. Kothmayr, T.; Schmitt, C.; Hu, W.; Brünig, M.; Carle, G. A DTLS based end-to-end security architecture for the Internet of Things with two-way authentication. In Proceedings of the 37th Annual IEEE Conference on Local Computer Networks—Workshops, Clearwater, FL, USA, 22–25 October 2012; pp. 956–963, doi:10.1109/LCNW.2012.6424088.
31. Raza, S.; Trabalza, D.; Voigt, T. 6LoWPAN Compressed DTLS for CoAP. In Proceedings of the 2012 IEEE 8th International Conference on Distributed Computing in Sensor Systems, Hangzhou, China, 16–18 May 2012; pp. 287–289, doi:10.1109/DCOSS.2012.55.
32. Restuccia, G.; Tschofenig, H.; Baccelli, E. Low-Power IoT Communication Security: On the Performance of DTLS and TLS 1.3. In Proceedings of the 2020 9th IFIP International Conference on Performance Evaluation and Modeling in Wireless Networks (PEMWN), Berlin, Germany, 1–3 December 2020; pp. 1–6, doi:10.23919/PEMWN50727.2020.9293085.
33. Caminati, G.; Kiade, S.; D'Angelo, G.; Ferretti, S.; Ghini, V. Fast Session Resumption in DTLS for Mobile Communications. In Proceedings of the 2020 IEEE 17th Annual Consumer Communications Networking Conference (CCNC), Las Vegas, NV, USA, 10–13 January 2020; pp. 1–6, doi:10.1109/CCNC46108.2020.9045119.
34. Gunnarsson, M.; Brorsson, J.; Palombini, F.; Seitz, L.; Tiloca, M. Evaluating the performance of the OSCORE security protocol in constrained IoT environments. *Internet Things* **2021**, *13*, 100333, doi:doi/10.1016/j.iot.2020.100333.
35. Santesson, S.; Tschofenig, H. Transport Layer Security (TLS) Cached Information Extension. Available online: https://rfc-editor.org/rfc/rfc7924.txt (accessed on 9 August 2021).
36. Wouters, P.; Tschofenig, H.; Gilmore, J.; Weiler, S.; Kivinen, T. Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS). Available online: https://rfc-editor.org/rfc/rfc7250.txt (accessed on 9 August 2021).
37. E. Rescorla, R. Barnes, H.T. Compact TLS 1.3 (IETF Draft). Available online: https://datatracker.ietf.org/doc/draft-rescorla-tls-ctls/ (accessed on 9 August 2021).
38. NIST. Lightweight Cryptography. Available online: https://csrc.nist.gov/projects/lightweight-cryptography (accessed on 9 August 2021).
39. Hall-Andersen, M.; Wong, D.; Sullivan, N.; Chator, A. NQUIC: Noise-Based QUIC Packet Protection. In Proceedings of the Workshop on the Evolution, Performance, and Interoperability of QUIC—EPIQ'18, Heraklion, Greece, 4 December 2018; Association for Computing Machinery: New York, NY, USA, 2018; pp. 22–28, doi:10.1145/3284850.3284854.