Efficient Security Protocols for Constrained Devices

Gunnarsson, Martin

LUND UNIVERSITY

PO Box 117
221 00 Lund
+46 46-222 00 00

Efficient Security Protocols for Constrained Devices

Martin Gunnarsson

LUND UNIVERSITY

ISBN 978-91-8039-570-0 (printed)
ISBN 978-91-8039-571-7 (electronic)
Series of licentiate and doctoral theses
No. 154
ISSN 1654-790X154

Martin Gunnarsson
Department of Electrical and Information Technology
Lund University
Box 118
SE-221 00 Lund
Sweden

Typeset using LaTeX.
Printed in Sweden by Tryckeriet i E-huset, Lund, 2023.

Abstract

During the last decades, more and more devices have been connected to the Internet. Today, there are more devices connected to the Internet than humans. An increasingly more common type of devices are *cyber-physical devices*. A device that interacts with its environment is called a cyber-physical device. Sensors that measure their environment and actuators that alter the physical environment are both cyber-physical devices.

Devices connected to the Internet risk being compromised by threat actors such as hackers. Cyber-physical devices have become a preferred target for threat actors since the consequence of an intrusion disrupting or destroying a cyber-physical system can be severe. Cyber attacks against power and energy infrastructure have caused significant disruptions in recent years.

Many cyber-physical devices are categorized as *constrained devices*. A constrained device is characterized by one or more of the following limitations: limited memory, a less powerful CPU, or a limited communication interface. Many constrained devices are also powered by a battery or energy harvesting, which limits the available energy budget. Devices must be efficient to make the most of the limited resources.

Mitigating cyber attacks is a complex task, requiring technical and organizational measures. Constrained cyber-physical devices require efficient security mechanisms to avoid overloading the systems limited resources. In this thesis, we present research on efficient security protocols for constrained cyber-physical devices.

We have implemented and evaluated two state-of-the-art protocols, OSCORE and Group OSCORE. These protocols allow end-to-end protection of CoAP messages in the presence of untrusted proxies.

Next, we have performed a formal protocol verification of WirelessHART, a protocol for communications in an industrial control systems setting. In our work, we present a novel attack against the protocol.

We have developed a novel architecture for industrial control systems utilizing the Digital Twin concept. Using a state synchronization protocol, we propagate state changes between the digital and physical twins. The Digital Twin can then monitor and manage devices.

We have also designed a protocol for secure ownership transfer of constrained wireless devices. Our protocol allows the owner of a wireless sensor network to transfer control of the devices to a new owner. With a formal protocol verification, we can guarantee the security of both the old and new owners.

Lastly, we have developed an efficient Private Stream Aggregation (PSA) protocol. PSA allows devices to send encrypted measurements to an aggregator. The aggregator can combine the encrypted measurements and calculate the decrypted sum of the measurements. No party will learn the measurement except the device that generated it.

Acknowledgements

Now, at the end of my time as a Ph.D. student, many people deserve my thanks and gratitude for helping me complete this undertaking.

First, I want to thank my primary supervisor Christian Gehrmann. As a Ph.D. student, he has always supported me with invaluable feedback and guided me in my work. He set high expectations for which I am grateful so I could complete this work with pride.

During my time, I had the pleasure of working with several co-supervisors. Martin Hell helped me get into the work and life of a Ph.D. student. He has guided me with my research and has always been helpful. Elena Pagnin took over the role of co-supervisor from Martin and immediately took a keen interest in my work and progress.

Before starting my Ph.D., I had the privilege of working with a great team at SICS in Lund. My colleagues introduced me to the research world, and we had a really good time. Thank you to Ludwig for getting me onto this track and always being a good colleague and a source of advice. Thank you, Arash, for always lending a helping hand and being such a great sport.

I thank the people at RISE Cybersecurity Unit in Lund and Stockholm. You have all helped in creating an inspiring research environment. Thank you, Simon, for all the help with the intricacies of Tamarin. To Rikard and Marco for your help with the details of the protocols you have developed and that we have implemented together. Nicolae has been a source of sage advice and ever helpful.

I also want to thank the rest of the Ph.D. students and seniors in the Crypto and Security group. You are all a very helpful and friendly bunch that I have truly appreciated working with. There is a vast range of skills in the group, and there has always been someone to ask when I have been stuck in my work. A special thanks go to Joakim, with whom I have had the pleasure of sharing an office for most of my time as a Ph.D. student. You are a great friend and source of help and support. I wish you the best of luck with writing your thesis!

I extend my heartfelt thanks to my loving family, Svante, Karin, Maria, and Olof. You have always supported and encouraged me. You have always been there when the late workdays have turned into nights, and the task at hand has seemed insurmountable. My family, grandparents, and friends have always supported and inspired me to work hard and make this possible.

My greatest thanks go to Cecilia, who has made this journey with me. You were by my side when completing this thesis seemed impossible and always supported me when I worked late nights and weekends. We first met when I came home from my first conference. Since then, you have been in my life and accompanied me on several more journeys. I am forever grateful for your love, support, and inspiration.

Martin Gunnarsson
Lund, February 2023

Contribution Statement

The following papers are included in this dissertation:

Paper I Martin Gunnarsson, Joakim Brorsson, Francesca Palombini, Ludwig Seitz, Marco Tiloca "Evaluating the Performance of the OSCORE Security Protocol in Constrained IoT Environments". In *Internet of Things*, vol. 13, Mar. 2021, Elsevier.

Paper II Martin Gunnarsson, Krzysztof Mateusz Malarski, Rikard Höglund, Marco Tiloca "Performance Evaluation of Group OSCORE for Secure Group Communication in the Internet of Things". In *Transactions on Internet of Things*, vol. 3, issue 3, Aug. 2022, ACM.

Paper III Martin Gunnarsson "Formal Verification of the WirelessHART Protocol - Verifying Old and Finding New Attacks". In *Eight Annual Industrial Control System Security (ICSS) Workshop* in conjunction with Annual Computer Security Applications Conference (ACSAC) 2022.

Paper IV Christian Gehrmann, Martin Gunnarsson "A Digital Twin Based Industrial Automation and Control System Security Architecture". In *IEEE Transactions on Industrial Informatics*, vol. 16, no. 1, pp. 669-680, Jan. 2020, IEEE.

Paper V Martin Gunnarsson, Christian Gehrmann "Secure Ownership Transfer for Resource Constrained IoT Infrastructures". In *Information Systems Security and Privacy. ICISSP 2020. Communications in Computer and Information Science*, vol. 1545, pp. 22-47, Jan. 2022, Springer.

Paper VI Joakim Brorsson, Martin Gunnarsson "DIPSAUCE: Efficient Private Stream Aggregation Without Trusted Parties". Submitted to *Privacy Enhancing Technologies Symposium (PETS) 2023.*

Paper	*Writing*	*Concepts*	*Implementation*	*Evaluation*
I	✓	✓	**✓**	**✓**
II	✓	✓	**✓**	**✓**
III	**✓**	**✓**	**✓**	**✓**
IV	✓	–	**✓**	✓
V	**✓**	✓	**✓**	**✓**
VI	✓	✓	**✓**	✓

In the table above a checkmark indicates parts where Martin Gunnarsson contributed. A bold checkmark indicate activities where Martin Gunnarsson took primary responsibility.

The work Martin Gunnarsson did for each paper is listed in greater detail in Chapter Chapter 3. Parts of the Introduction and Background Sections in this thesis is based on Martin Gunnarsson's Licentiate Thesis [Gun20].

In Paper I, Martin was involved in writing the paper and describing OSCORE. Martin wrote the implementation and wrote the test suites. Martin did the evaluation.

In Paper II, Martin was involved in writing the paper and describing Group-OSCORE. Martin was leading the implementation work and wrote the test suites. Martin did the majority of the evaluation.

In Paper III, Martin was the solo author. Martin modeled WirelessHART and did the security analysis.

In Paper IV, Martin wrote the implementation and performed the performance evaluation.

In Paper V, Martin co-designed the secure ownership transfer protocol. Martin did parts of the security analysis. Martin implemented the protocol and performed the experimental evaluation.

In Paper VI, Martin evaluated the state-of-the-art protocols. Martin co-designed the proposed protocol. Martin implemented the protocol and performed the experimental evaluation.

A further description of the papers' contributions *to the research field* is presented in Section 3.1.

Other Contributions

Martin Gunnarsson has also contributed to the following manuscripts, not included in this thesis.

- Ludwig Seitz, Marco Tiloca, Martin Gunnarsson and Rikard Höglund: ”Secure Software Updates for Critical Infrastructure”. In *International Conference on Information Systems Security and Privacy (ICISSP)*, 2023.

- William Tärnberg, Martin Gunnarsson, Maria Kihl, Christian Gehrmann: ”Demonstration: A cloud-native digital twin with adaptive cloud-based control and intrusion detection”. In *Electronic Communications of the EASST, Vol. 80, 2021, Universitatsbibliothek TU Berlin*.

- Christian Gehrmann and Martin Gunnarsson: ”An Identity Privacy Preserving IoT Data Protection Scheme for Cloud Based Analytics”. In *2019 IEEE International Conference on Big Data (Big Data), Los Angeles, CA, USA, 2019*, pp. 5744-5753, IEEE.

- Martin Gunnarsson and Christian Gehrmann: ”Secure Ownership Transfer for the Internet of Things”. In *International Conference on Information Systems Security and Privacy (ICISSP)*, 2020.

- Martin Gunnarsson, Nils Vreman and Martina Maggio: ”Trusted Execution of Periodic Tasks for Embedded Systems”. Submitted for review to *IFAC World Congress*, 2023.

Contents

Introduction

Ever since the first cyber-physical system, a vending machine, was connected to a network, the phenomena of connected, new cyber-physical devices have been viewed with both caution and optimism. Connected devices bring the possibilities of remote control, increased performance, and extended functionality. But networking also invites all the trouble of the network domain to cross into the physical domain. Security has become more relevant with the deployment of connected devices in more areas of society. These new types of devices can easiest be categorized by what they are *not*. They are not computers, cell phones, or servers. Instead, they are *embedded*, often lacking a traditional user interface. The term Internet of Things (IoT) is often used to describe the heterogeneous devices connected to the Internet. IoT devices can be everything from connected household appliances to connected vehicles.

Today, the adoption of Cyber-Physical Systems (CPS) is ubiquitous. They are deployed in significant numbers to provide computation and connectivity to more fields. Both the advocates and opponents of CPS have been proven right in their predictions. CPS devices have provided society with huge benefits, but these benefits have come with the threat of disruption from cyber-attacks and malicious actions. Security risks with the Internet of Things have been compared to the safety risks of cars. Accidents happen, but people still drive.

One specific class of CPS is Industrial Control Systems (ICS). ICS security is of particular importance since ICS controls industrial processes, power plants, energy infrastructure, etc. An aggravating circumstance for security in ICS is that ICS infrastructure was initially deployed isolated from outside networks, such as the Internet. Internet connectivity benefits remote supervision, management, and control for ICS. As such most ICS, today are connected to the Internet. Most legacy ICS infrastructure has been developed, assuming it will operate in isolated, friendly networks that separate the ICS hardware from the Internet. A series of segmented networks and firewalls have been developed to mitigate the risks. In the next generation of ICS, Industry 4.0, connectivity is likely more prevalent. Our research is based on the observation that current ICS infrastructure and protocols have not been developed with security as an objective [Sau+11] and that connectivity in ICS is likely to increase [SBC19].

Many tools and techniques must be utilized to secure systems, being traditional IT systems, ICS, CPS, or IoT. These include communications security, key management, and device management as essential aspects for the secure deployment, management, and operation of constrained connected devices.

All of these topics are well worth studying, and each can fill a thesis. In this work, we have focused on efficient security protocols for communications security and device management for ICS, CPS, and IoT. We have studied existing protocols, OSCORE, Group OSCORE, and WirelessHART, from security and efficiency perspectives. Further, we have introduced new concepts and protocols that can aid in securing the next generation of systems, offering new capabilities. We have studied Digital Twins, a promising concept gaining more attention, and how it can be used to create security architectures for ICS. The topic of secure ownership transfer was investigated, and we have proposed a novel, efficient protocol to transfer the control of IoT devices.

Privacy is also a concern in ICS [Gid+20; SWW15]. Privacy has traditionally been studied in the context of individuals. However, the same schemes are also relevant in ICS settings, where privacy properties instead protect industrial trade secrets. For example, measurements and time series data can reveal industrial trade secrets of processes or how equipment is used. Evaluating functions with encrypted inputs is currently under intense study and development. Private Stream Aggregation (PSA) has been proposed to enable privacy-preserving data collection and aggregation. It is a concept similar to Functional Encryption, allowing data to be collected and summed without revealing individual measurements.

Securing connected devices or *things* in the Internet of Things is challenging because of their limited capabilities in terms of memory, computing power, network connectivity, and energy budget. Efficient solutions are required to secure the current and the next generation of connected constrained devices. Efficient, secure communications protocols for constrained devices can reduce costs and enable more sustainable devices.

This thesis was mainly funded by Sec4Factory, grant RIT17-0032 by the Swedish Foundation for Strategic Research (SSF). Sec4Factory aims to develop communication and security solutions for the next generation of production environments.

1.1 Thesis Outline

After this introduction, the background and preliminaries will be presented in Chapter 2 to aid the reader in the second part of this thesis. The background will introduce constrained devices and how they often are part of cyber-physical systems. Then we will provide an introduction to lightweight, secure communications and device management. Last is a section of formal protocol verification that will aid the reader with Paper III and Paper V. The second part of this thesis presents the peer-reviewed papers (I, II, III, IV, V) and one manuscript under review (Paper VI).

Background

Alice works as the CTO for Manufacturing Corp llc. The company's executives have decided on a strategy to innovate their production facilities to increase efficiency and productivity. Alice proposes a strategy of increased monitoring and connectivity in the factory.

When the executives hear that this is sometimes called the Industrial Internet of Things (IIoT), they are hesitant. Security concerns are raised. "The S in IoT stands for security", is said. How can the company protect its assets and prevent malicious actors from taking over its systems? Further concerns about cost are raised. Can the hardware cope with the additional load if security is added as an additional feature? Furthermore, can legacy systems interoperate with newly acquired devices?

Alice informs the executives about the possibilities of predictive maintenance, better monitoring and control, and remote management, which are indeed possible to perform securely and efficiently.. The prospect of reducing factory downtime and increasing efficiency of production processes seems tempting to the executives.

How can protocols and solutions be developed to help Alice and her company implement IIoT securely and efficiently?

2.1 Constrained devices

The term *constrained devices* or *constrained nodes* can be used to describe computing devices with limited capability. These limitations can be CPU-Power, RAM and ROM memory, network capabilities such as latency and bandwidth, and energy. Energy can be limited because the device is powered by an energy harvesting device such as a solar panel or from a battery. Devices may sleep for periods to save energy and not be able to respond to communication or perform any computations during those intervals.

Because not all constrained devices are the same, the Internet Engineering Task Force (IETF) has standardized terminology for these devices [Bor+22] and has defined categories for different types of constrained devices. One limiting factor when it comes to performing complex computations is the size of the available memory. In Table 2.1, we show the categories of constrained nodes as defined by IETF. Memory is used as a metric instead of CPU-speed, because memory size drives the final cost of the device. Memory takes up a lot of space on the semiconductor die, and the size of the die directly influences the price [Koo15].

The effect of these memory limitations is that a memory-constrained device is only capable of doing a small set of computations. A small amount of ROM limits the amount of code that can be present in the system, thus only a select few tasks can be done. A small amount of RAM limits the number of intermediary states and the size of the data that can be handled. For example, in a protocol such as Datagram Transport Layer Security (DTLS), RAM limitations directly limit the number of security contexts, which in turn limits the number of simultaneous connections the device can handle.

Table 2.1: Classes of constrained devices according to RFC7228.

Class	**Data Size (RAM)**	**Code Size (ROM)**
Class 0, C0	$\ll$ 10 KiB	$\ll$ 100 KiB
Class 1, C1	$\sim$ 10 KiB	$\sim$ 100 KiB
Class 2, C2	$\sim$ 50 KiB	$\sim$ 250 KiB

Apart from memory constraints, energy is one important aspect to consider when evaluating the capabilities of a system. It is no surprise that running a CPU, peripheral, and a radio-modem consumes energy. Constraints of energy have significant ramifications when a system is designed, energy-efficient CPUs are generally less powerful, and the same goes for peripherals and radio-modems. IETF has put the energy constraints on a scale from 0 to 9, where 9 is no limitation, and 0 is energy harvesting.

The different categories can be seen in Table 2.2. For example, an E9 device can be an Ethernet-enabled surveillance camera that is powered by Power over Ethernet. A class E0 device is, for example, an RFID-tag that harvests energy when a reader interrogates it, this small amount of energy harvested is then used to send a reply.

To reduce the energy consumption of the radio, it can be periodically switched off. Radio duty-cycling lets a device to turn off the radio during regular, predicable intervals, and turning it on so it can send and receive messages. Radio duty-cycling is handled by the network stack between the MAC layer and the physical layer. A remote party might not know the radio duty-cycle of a device, so communication is often asynchronous.

Table 2.2: Classes of energy constraints according to RFC7228.

Class	Type of energy limitation	Example Power Source
E0	Event energy-limited	Event-based harvesting
E1	Period energy-limited	Periodically recharged battery
E2	Lifetime energy-limited	Non-replaceable primary battery
E9	No limitations to available energy	Mains-powered

2.1.1 Security Aspects for Constrained Devices

After describing the capabilities and limitations of Constrained devices in the previous section, we will now discuss the implications for security. Because of the limitation in CPU-power, memory, energy, and network capabilities, traditional security solutions developed for desktop and server computing environments can be unsuitable. The limited performance of constrained CPUs make public-key cryptography time and energy-consuming, although hardware-acceleration can be utilized to make it more feasible.

Traditional x509 certificates might require too much bandwidth and memory to be feasible in constrained devices. Research and development are ongoing to reduce these numbers [For+17; Mat+23]. But with limited network capability; it might be difficult to validate an entire certificate chain, thus severely limiting the usefulness of certificates.

The ubiquitous protocol for secure communication in traditional IT Transport Layer Security (TLS) [Res18] uses TCP as the underlying transport mechanism. Transport sessions are not desirable when constrained devices communicate asynchronously. Instead, DTLS is standardized as an alternative to TLS. DTLS uses UDP as the underlying transport; this removes the need for TCP sessions. Using UDP also reduces the overhead of each transmitted packet.

The security protocols and solutions developed for constrained devices must consider these limitations [GKS18]. Security solutions must be resource-efficient. Limiting message overhead to save bandwidth and energy is a requirement. When selecting cryptographic primitives, efficient algorithms must be prioritized. This means using symmetric-key cryptography where possible and limiting the use of public-key cryptography. Reducing the transmitted message size is also an essential goal since sending and receiving data consumes energy. The OSCORE protocol, recently standardized by IETF, was designed to provide end-to-end communication security in the presence of untrusted proxies and was designed with low message overhead as one of its design goals [Sel+19a]. We have evaluated the efficiency of the OSCORE protocol in Paper I; we measured message overhead, processing time, and energy consumption to evaluate the efficiency of the protocol. In Paper II we have evaluated the Group OSCORE protocol, that provide secure end-to-end group communication.

In Paper V, we have developed a secure ownership transfer protocol for very constrained devices. We have designed an efficient protocol, only using symmetric cryptography. The topic of secure ownership transfer will be described in detail in Section 2.4.1. In Paper VI, we have designed a protocol that allow sensors to send encrypted data to an aggregator. The aggregator can then combine the encrypted data and compute a sum of the encrypted values, without learning the decrypted values. The technique we have used is called private stream aggregation and is described in Section 2.3.3. This protocol is mainly using symmetric cryptography to meet the efficiency requirements.

2.2 Cyber-Physical Systems and Industrial Internet of Things

One application area of particular note for constrained devices is Cyber-Physical Systems (CPS). Sensors, devices that measure the physical environment, and *actuators* that alter the physical environment are often constrained devices. The impact of a compromised CPS is more than just loss of data or availability. One particular case of CPS is connected manufacturing systems. Today, most manufacturing is done in a connected facility, which poses extra security challenges since legacy systems did not always consider security a priority. Since these devices interact with their physical environment, security is essential. In this section, we will introduce traditional industrial control systems and look at how they have evolved into Industrial Internet of Things (IIoT).

The first automated industrial control system can maybe be said to be the loom developed by Joseph Marie Jacquard in 1804. It can be seen as the first step in the evolution into modern automation systems. It was a primitive computer ”programmed” by punch cards to weave patterns in cloth. Since then, automated systems that control physical processes have become increasingly prevalent and increasingly sophisticated. The technological evolution has evolved from pneumatic systems through electro-mechanic systems to digital systems. Virtually everything on a modern factory today is automated. This has increased efficiency and the precision of machinery.

But a second technological revolution has happened in parallel to the automation of the industry. The first device connected to the Internet was a vending machine at Carnegie Mellon University in 1982. Almost 40 years later the number of cyber-physical devices is projected to reach 75 billion by 2025[1].

Modern industrial control systems are an amalgamation of information and communications technology and the evolved automation systems used in the industry.

During the 1980s, drawbacks were identified with having isolated industrial control systems (ICS) [WSJ17]. Connecting manufacturing systems and distribu-

[1] https://www.statista.com/statistics/471264/iot-number-of-connected- devices-worldwide/

tion systems with the organization's Enterprise Resource Planning systems (ERP) was desired to increase the efficiency and agility of process control. This development lead to more complex systems, such as Supervisory Control And Data Acquisition (SCADA). Furthermore, by connecting ICS to the internet allow operators to configure or control processes and machines remotely.

However, this has opened a new attack surface in the form of remote attacks. We will describe the security aspects of ICS shortly. First, we will give the reader an overview of the differences between IT environments and ICS environments to provide a better understanding of the security implications.

Figure 2.1 shows the main parts of an industrial control system, according to NIST [SFS11]. The most important part of the system is the controlled process; without a process to control, there are little use for a control system and its associated parts. The Controller uses input values from sensors, which can, for example, be temperature sensors or flow-meters. These input values are used by the Controller to calculate commands that the Controller then sends to actuators. Actuators can be valves, pumps, and industrial robots. The actuators, in turn, alter the physical properties of the controlled process to acheive the desired results.

The controlled process is, although automatic, still supervised by a human operator using a Human-Machine Interface (HMI). From the HMI, an operator can monitor the status of the process and control it manually. Part of the HMI is also an easily overlooked component essential for safe operations, the STOP mechanism. The stop buttons are not only found on the screens in the control rooms but also as big red buttons scattered around the premise.

Lastly, most systems today are connected to remote diagnostics and maintenance systems. These systems can be used to control the process remotely and to collect and process data, not only about the process but also about each part in the system, such as individual robots and machines.

A more detailed model, made to represent a total view on an ICS deployment, is the Purdue Model [Wil92]. The Purdue Enterprise Reference Architecture, as is its full name, was developed in 1990 by members of the Industry-Purdue University. The model is shown in Figure 2.2; and it gives a hierarchical view of different parts of an ICS system.

Starting at the bottom with Level 0, we have the devices that form the interface between the physical process and the control system, i.e. sensors, actuators, and robots. At Level 1, we have different systems of local control, continuous and discrete control of processes and the essential safety control. Moving up to Level 2, (this is the highest level in what is called a Cell) we have the Human-Machine Interfaces (HMI) and Engineering Workstations. A plant can have more than one Cell. At Level 3, the systems that manage the Cells are located; this is also where the site operations and manufacturing operations systems are.

Above Level 3 is the Demilitarized Zone (DMZ). This area separates the critical and sensitive parts of process from the rest of the IT environment of the organization. The DMZ is separated from both sides by firewalls that filter the network

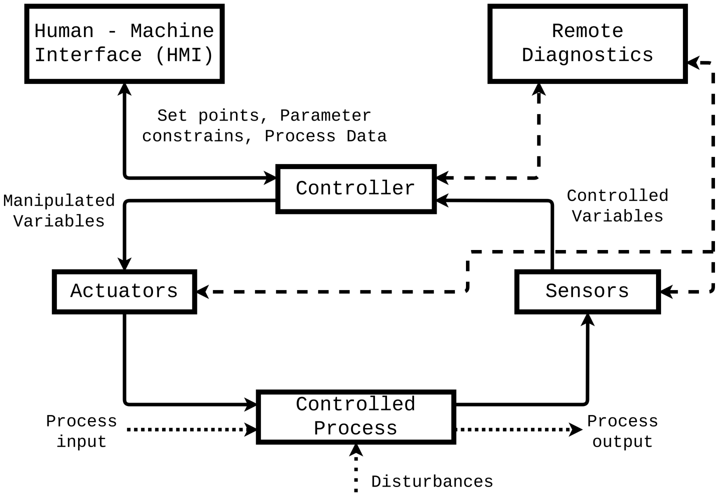


Figure 2.1: ICS operation according to NIST, from NIST Special Publication 800-82

traffic flowing through DMZ. The idea behind the DMZ is to have no direct connection into or out from the Levels 3 and below. If remote access is used in a system, this is where gateways for a remote access system is situated. Level 4 and 5 are where traditional IT resides, email servers, the Intranet, and business planning are located here.

Perhaps the most crucial difference between IT and ICS is that in industrial control systems, the *process* is the end goal. The process generates value by producing something; thus, it gets prioritized when resources are limited.

Another key technical difference between familiar IT systems and ICS systems is the aspect of real-time tasks in ICS. A process that controls, for example, a chemical process or an electricity grid, must have predictive latencies with variability, i.e. jitter. A correct control signal that arrives too late is of no use. In IT, there is often no need for real-time deadlines. Most IT systems process data at the request of a user; as long as the user perceives the system as responsive, the performance is good enough.

Close to the aspect of real-time deadlines is the property of availability; it is easier to have redundancy in an IT system. Multiple instances of a cloud server with a load-balancer can keep a service available even when parts of the system is undergoing maintenance. But in ICS, an outage can have severe consequences, e.g., an electricity grid or drinking water supply can impact thousands of people. To guarantee the availability of critical processes, the process control system must be available. The control system is often redundant to prevent outages caused by a faulty control system.

Systems used to control different processes are highly specialized. Not only as ICS devices but also within the field of ICS control systems for different types of

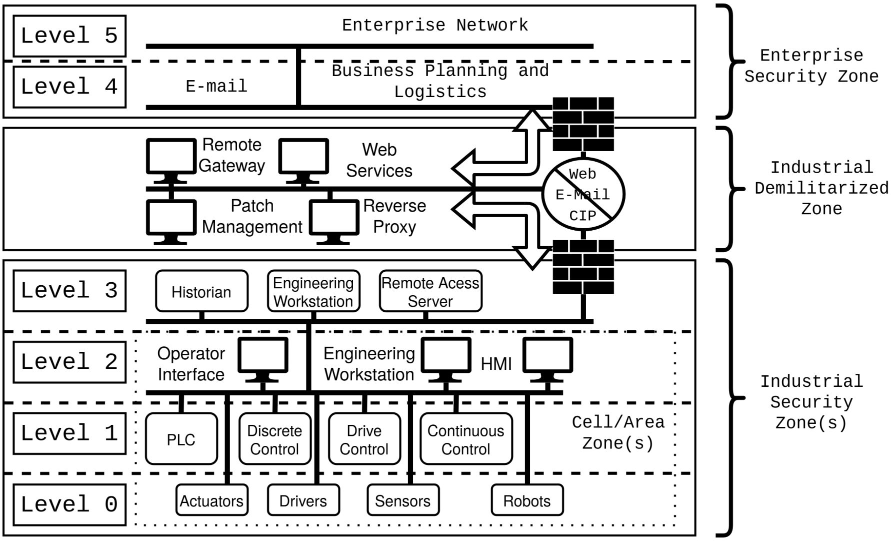


Figure 2.2: The Purdue Enterprise Reference Architecture, a model for ICS.

processes have significant differences. There is very little commonality between, for example, a welder robot from a car building assembly line and a phase-controller from an electricity grid. The complexity of the process by itself, together with the specialized systems, makes almost all ICS deployments unique.

In ICS, component and system lifetimes are often long [KKG19]. Systems and parts are expensive, a stop in production to install and deploy a system might be too expensive for an organization. Patches are also slowly applied to systems, not only because of the risk of breaking some functionality but also because a certified system might lose the certification when a patch is applied.

Because ICS devices have been developed in silos separated from ordinary IT devices, the protocols, standards, and technologies used in ICS is different compared to a traditional IT environment. This not only affects the interoperability of ICS and IT systems but also does not let ICS systems take advantage of the development of better IT security protocols and mechanisms.

As we have shown here, many aspects differentiate ICS from IT. Of course, these differences impact security, and we will discuss that in Section 2.2.2.

2.2.1 Industry 4.0 and Next-Generation Manufacturing

The third Industrial Revolution was the digitalization of manufacturing from the mid of the 1900s and onwards. The fourth industrial revolution, is the next generation in the development of industry.

In 2012 a German research project presented a set of recommendations to the German government about the future of the industry [Kag+13]. The goal of *Industrie 4.0* or Industry 4.0 is to improve productivity. The productivity improvements will be gained from an increase in flexibility, where factories build to demand instead of producing to inventory. A critical factor in achieving this will be collecting, sharing, and spreading information through the factory, together with decentralized decision making.

The list of technologies and concepts that will realize Industry 4.0 is long; among them are IoT, Cybersecurity, Cloud Computing, Big Data, and Simulation. Other technologies are listed, such as Augmented Reality and additive manufacturing, but we will focus on the technologies relevant to this thesis.

IoT is a key component of Industry 4.0, used in this industrial setting. Industrial IoT (IIoT) is often used to describe the connected devices in manufacturing systems. In Section 2.3.1 we will discuss IoT and IIoT. IoT is also key in Paper I, II, III, IV, and VI.

ICS data collection is needed for advanced analytics and to improve production performance, to give two examples. Collecting data from a production environment will often result in large data sets; doing analytics on these big data sets is a discipline called Big Data analytics. In Paper VI, we look at this collecting of data from a privacy perspective. We have developed a protocol that allows IoT devices to send encrypted data to an aggregator for analytics where the aggregator only learn the results of the computation, and not the individual data items.

2.2.2 Security Aspects of Industrial Control Systems

The properties we have described above highlights that security for ICS faces different challenges compared to IT security. In IT security, Confidentiality, Integrity, and Availability (CIA) triad is often used [Per08] to describe the goals in IT security activities. Note that the CIA refers to the *data* used *in* the system and not the system itself. The data shall be confidential; that is, it shall not be readable by any unauthorized entity. The data shall be integrity protected, which means that an unauthorized entity shall not be able to manipulate the data without being detected. Lastly, the data shall be available since data is of no use if it not readily accessible.

According to several researchers, for example, [GK16] and [SFS11], the CIA security model does not map well to ICS. For instance, while the CIA model considers theft or manipulation of data, in an ICS setting, risk of personal injuries or equipment damage must also be taken into account. See [SFS11] for a more detailed analysis of this issue.

The availability of systems is more critical in ICS. A production plant can take days to come back online after a stop. The resulting downtime could cause high costs for the owner and operator.

In this thesis, we have included papers that deal both with traditional security properties, like the above mentioned CIA triad, and security life cycle management. Since outages due to maintenance and cyber-attacks should be avoided, methods for doing security life cycle management in ICS are needed. This security life cycle management must not waste the limited resources in ICS. In Paper IV, we have developed a security architecture using the concept of Digital Twin. Digital Twin are further discussed in Section 2.4.2, and as shown in the included paper, it is a powerful tool for security monitoring with a low impact on legacy systems and real-time critical systems.

2.3 Lightweight secure communications

A protocol is a set of rules on how computers shall communicate. A protocol can detail how fast information is transmitted, the size of messages, and the types of messages sent. At first, security features such as encryption, integrity protection, replay protection, and source authentication were not considered necessary. A lot has happened since the 1960s, and today communications security is a crucial concern for communicating devices. The development of protocols reflects this.

In the previous section, we described constrained devices and how they are limited in the computations and communication they can perform. One limitation is that low-powered constrained devices often utilize radio communication with limited bit rates and high latency. To avoid congesting the network and to limit the time when the radio is in the power-hungry transmit state, messages must be compact. Constrained devices also have limited memory, so protocols must be easy to process and not require extensive and complex code and RAM. The processing time of messages must be kept short to preserve power since an idle or sleeping CPU consumes less energy than an active CPU. Cryptographic operations are computationally intensive [De +08], but the security features they offer are often a requirement for the secure operations of devices.

This section will give an overview of communications protocols for constrained networked devices, sometimes called Wireless Sensor Networks (WSN) and the Internet of Things (IoT). We will begin by giving a more detailed definition and description of WSN and IoT. Then we will provide an overview of existing communications protocols for WSN and IoT. A relatively new concept in communications security is object security. Paper I and Paper II in this thesis evaluate protocols with object security. In the next section, we will give a definition and the rationale of object security. Finally, privacy is a fundamental concern in WSN and IoT. One technique for privacy-preserving data aggregation is private stream aggregation. In the last section, we will describe private stream aggregations.

2.3.1 Wireless Sensor Networks and Internet of Things

Wireless Sensor Networks (WSN) are a designation for a network of, often constrained, devices that communicate using wireless technology. The purpose of the network is often to collect sensor readings from several different places for further processing in a central server. The increased performance and decreased price of micro-controllers and associated devices have made it possible to deploy sensors with a microcontroller and some kind of networking capability very cheaply. Often these systems are powered by a battery, combined with the need to keep costs down the resulting systems can usually be classified as Constrained devices, as described in Section 2.1.

Internet of Things has become the catch-all term for all kinds of connected devices. Everything from a factory connected to a SCADA network to a refrigerator with WiFi can be called an IoT device. Sometimes distinctions are made such as Industrial IoT (IIoT) for connected devices used in an industrial setting. The difference between IIoT and connected control systems described in Section 2.2 is that IIoT has a more direct connection to computing resources such as a cloud environment [MM16]. An IIoT deployment will differentiate from the Purdue reference model we showed in Figure 2.2 in that an IIoT deployment will have a direct connection between the edge devices and the cloud. There is no DMZ in IIoT, like the one that can be found in the Purdue model.

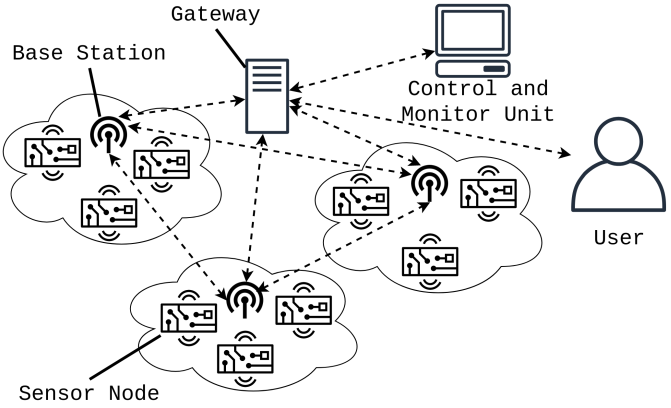


Figure 2.3: A schematic of a Wireless Sensor Network in an industrial setting.

Communications Protocols for WSN and IoT

Many actors have developed Wireless Sensor Networks; as a result of this, there exists a large number of communication protocols and network stacks. WiFi, Bluetooth [Haa00], Bluetooth Low Energy (BTLE) [HH12] and ZigBee [All10] all use the unlicensed 2.4 GHz frequency band. ZigBee use the IEEE 802.15.4 [IEE11] as a link-layer protocol and provides its own networking layer on top of that. IEEE 802.15.4 is also used in many other low-power IoT protocol stacks. LoRa [Sor+15] uses unlicensed frequencies in the sub-gigahertz range to increase the range compared to the protocols in the 2.4 GHz band. NB-IoT[Rat+16] uses optimized cellular technology and base stations to achieve wide coverage.

Several communications protocols exist, the two most common are MQ Telemetry Transport (MQTT) [HTS08] and Constrained Application Protocol

(CoAP) [SHB14]. MQTT is of the type publish-subscribe; subscribers subscribe to topics, and publishers publish data to these topics. Message brokers then act as intermediaries to forward the data from the publishers to the relevant subscribers. MQTT is usually transmitted over Transmission Control Protocol (TCP), and TLS is used to secure TLS connections and, in extension, MQTT. CoAP is a RESTful protocol like Hypertext Transfer Protocol (HTTP). It is commonly transmitted over User Datagram Protocol (UDP), and the most common way of securing it is with DTLS. It can also be transmitted over TCP or WebSockets [MF11], possibly secured with TLS, or even raw IEEE 802.15.4 or BLE. In this thesis, we have evaluated OSCORE, an alternative approach to securing CoAP. OSCORE uses a security concept called object security that we introduce in Section 2.3.2.

Protocols are constantly being proposed, developed, and deployed. Standardization organizations such as IEEE, IETF, IEC, and many more constantly develop novel protocols with new features and update old protocols, sometimes to respond to vulnerabilities. From the academia, universities a research institutes are also active in this area. Some of these activities are in cooperation with standardization organizations.

Standardization organizations involve participants from the public and private sectors. But protocol development is also conducted independently in industry, without standardization in the usual bodies, or with specific industry working groups. Some of these initiatives are the Salt Channel protocol developed by Assa Abloy AB [LJ19]. Salt Channel is a secure channel developed for constrained devices. It builds on TweetNaCl, a compact cryptographic library [Ber+15]. Assa Abloy has also developed the Julia Key agreement protocol [LF21]. Julia requires fewer scalar multiplications during its execution than Diffie-Hellman, making it less computationally expensive.

Thread [KKC19] is a protocol stack that builds on IETF standards. It has been developed by the Thread Group and is intended for home automation. Thread uses IEEE 802.15.4-2006 as physical and MAC layers and uses 6LoWPAN, IPv6, UDP, and DTLS. The application layer can be either HTTP, CoAP, or MQTT. A companion to Thread is Matter [All22]. Matter has been developed by the Connectivity Standards Alliance and is intended to allow connected home devices from different vendors to connect and communicate. Matter can be used over Thread, Zigbee, BLE, or Wi-Fi and provide device enrollment, device attestation, and a data and interaction model, among other features.

Noise is a *protocol framework* [Per18], it specifies a handshake phase for key exchange and secure transport messages that is used to transport messages. Noise does not specify a specific transport layer or a use case. But it can add a key exchange message and secure messages to a protocol that lacks these features.

2.3.2 Object Security

The earliest reference to Object Security was made in 1995 in RFC1848 [Cro+95] titled *MIME Object Security Services*. The document details how Multipurpose Internet Mail Extensions (MIME) objects shall be encrypted and processed. MIME is a standard that relates to email. Encrypting each mail in a self-contained object is a good solution. The sender can not know if the recipient can receive the email at the time of sending, requiring a store-and-forward mail server. Since the email first goes to a server, and then, at a later time is forwarded to the recipient means that setting up a secure session between the sender and the recipient is not possible. The problem with the recipient not being available at the time of sending is solved by using intermediate servers that store and forward the emails. Protecting each mail in a self-contained object eliminates the need for secure sessions between the intermediate servers.

One schematic diagram of an object security message can be seen in Figure 2.4. It does not show any actual message format, but rather a sample of some fields that might be present in such a structure. What differs between formats and standards, not shown in the figure, is encoding.

Key ID	Algorithm ID	Nonce	Encrypted Data	MAC or Signature

Figure 2.4: A schematic of a message or data item protected with object security.

Object security is a good fit for when a device sends messages to several receivers. Transmitting is only done at intervals, thus object security eliminates the need for keeping a session alive. Apart from email, wireless senor networks and constrained networked devices have proved a good fit for object security. Because of the energy limitations and constrained nature of devices, messages are only sent sporadically.

Object security has also been used in web contexts, such as JavaScript Object Notation (JSON) Web Signatures (JWS) [JBS15], JavaScript Object Signing and Encryption (JOSE) [Bar14] also XML encryption [Ima+13]. A similar standard to JOSE is CBOR Object Signing and Encryption (COSE) [Sch22b; Sch22a]. CBOR stands for Concise Binary Object Representation and has been standardized by IETF as a more compact alternative to JSON [BH20]. The difference between JOSE and COSE is the encoding, JOSE uses JSON while COSE uses CBOR. Due to the compact serialized format of CBOR, COSE is more compact than JOSE [Kal15].

One benefit of the object security concept is that it can be used to provide end-to-end encryption. If a message takes a winding route to its destination, encrypting the message in a self-contained way is a practical solution to protect the contents until it arrives at the destination. This is why PGP and other email encryption schemes work so well; encrypted email can travel between many email-servers until they arrive at the receiver. The receiver, provided they possess the correct keys, can then decrypt the message. These schemes and protocols are quite old now, but they are still used in email applications today.

Perhaps the first implementation of object security for a constrained wireless device can be found in [Bro+00] were the authors port Pretty Good Privacy (PGP) to a Research In Motion (RIM) pager. The RIM pager has more memory than a Class 2 constrained device, but it is still a relatively limited device, considering it uses a 10 MHz Intel 386 CPU from the 1990s. In the paper, the authors note that Elliptic curve cryptography (ECC) operations, such as signing messages and verifying signatures, can be performed in a couple of seconds. Elliptic curve cryptography is a type of public-key algorithms that require smaller keys and less computation than alternative algorithms for a given level of security. These qualities make ECC suitable for use in constrained devices. The authors argue that the performance of ECC can be acceptable for an email solution.

Another recent application for object security is end-to-end security for instant messaging apps. Asynchronous communication makes this method of encrypting messages a suitable solution. The person you send a message to might not have a direct connection to you. Instead of setting up a secure channel, encrypting the message in a self-contained way and sending it through intermediaries that do not possess the key, encrypts and secures the message end-to-end.

This use-case is very similar to the problem statement behind OSCORE. Messages pass through intermediate proxies and middle-boxes, possibly without the endpoints knowledge. The receiving server might be sleeping to preserve energy, requiring a proxy that forward the message to the server when the server is active. Because of these reasons, setting up a secure session is not desirable, since a client would have to wait until the sever wakes up. For a CoAP proxy to work the messages have to be decrypted by the proxy, requiring termination of DTLS or TLS sessions.

Even if object security can provide confidentiality, integrity protection, replay resistance, and source authentication, the issue of privacy remain. A message to a receiver like the message shown in Figure 2.4 can have its origin revealed by metadata such as the Key ID. There also exists scenarios where the contents of a message should remain secret even from the recipient. Collection of data introduce privacy risks. If the data aggregator receives data from individual devices, the collected data might reveal unintended extra information. In the next section we will look at ways to perform privacy-preserving data aggregation.

2.3.3 Efficient Privacy-Preserving Data Aggregation

As sensors that collect data are deployed in environments close to individuals, privacy becomes important. Sensors were initially deployed in industrial control systems for measuring the state of physical processes; thus, privacy was of little concern. Most communications protocols presented in this thesis have been developed with confidentiality and integrity protection as the main objectives.

When data is collected and analyzed in large scale, valuable analytics results can be obtained. A manufacturer of IIoT devices or other manufacturing equipment might want to gather statistics from the devices to improve their products. The users of the devices will probably hesitate to send data from their potentially sensitive ICS environments to another company. Such data could be used to determine how a user of a product have set up their facilities, and reveal trade secrets.

The problem also exists in the context of building automation, or smart homes. Data collected from households, such as electricity consumption, water consumption, and door locks, reveal the habits and actions of the members of the household.

Functional Encryption (FE) [BSW11] is a technique that allows evaluation of a function over encrypted data. Only a party that knows the function key can compute the plaintext result of that function. There exists a subclass of Functional Encryption called (Decentralized) Multi-Client Functional Encryption ((D)MCFE). DMCFE is a class of FE for multiple parties that contribute encrypted data, a perfect fit for distributed deployments such as IoT. However, the most efficient DMCFE that allows computation of the inner product of encrypted data is still too computationally intensive for constrained devices [Cho+18; Abd+19; Cho+20]. These DMCFE constructions utilize bilinear pairings or require ciphertext of size $O(n)$ where n is the number of users in the system.

For most schemes there exists a trade of between the types of operations that can be performed on encrypted data and performance. Fully homomorphic encryption can perform any computation on encrypted data but is very computationally intensive. A more efficient scheme is necessary for constrained devices that collect sensor data. The privacy preserving aggregation of time-series data can be performed with a technique called Private Stream Aggregation (PSA).

Private Stream Aggregation

One common data collection scenario is the collection and addition of measurements. Private Stream Aggregation was first proposed in [Shi+11] and allowed a system to compute sums of *labeled* encrypted measurements. The system consists of n users that generate measurements d_i, and the data is then encrypted and sent to an aggregator. The aggregator can then compute the sum of the n measurements. We show a schematic PSA system in Figure 2.5.

Each encrypted measurement is encrypted with a device unique key sk_i and a label l. Only if all measurement with the same label is decrypted the aggregator will learn the result. It is, therefore, not possible for an aggregator to only sum some of the measurements to reveal one individual measurement. PSA only requires parties to communicate with the aggregator, reducing communication overhead and making PSA feasible in a segmented network.

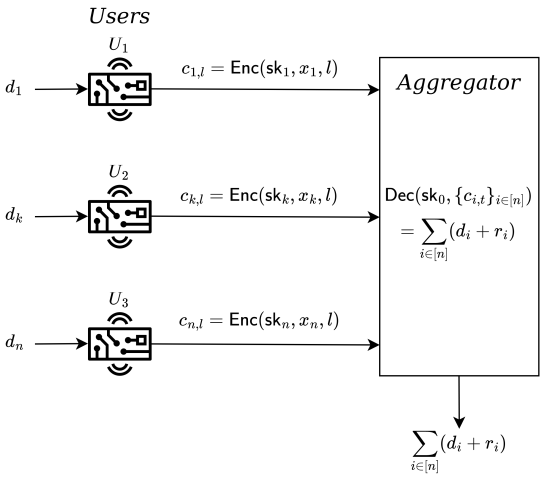

Users
U_1
U_2
U_3
d_1
d_k
d_n
$c_{1,l} = \mathsf{Enc}(\mathsf{sk}_1, x_1, l)$
$c_{k,l} = \mathsf{Enc}(\mathsf{sk}_k, x_k, l)$
$c_{n,l} = \mathsf{Enc}(\mathsf{sk}_n, x_n, l)$
Aggregator
$\mathsf{Dec}(\mathsf{sk}_0, \{c_{i,t}\}_{i\in[n]})$
$= \sum_{i\in[n]}(d_i + r_i)$
$\sum_{i\in[n]}(d_i + r_i)$

Figure 2.5: A schematic pricture of PSA, adapted from [Wal+21].

Several PSA schemes have been proposed [CSS12; JL13; LEM14; BJL16] [Emu17; BGZ18; EK21; Wal+21; Tak+23]. All proposed PSA schemes require a trusted third party that distributes keying material. In Paper VI, we argue that this proposition is weak and define PSA with a distributed setup. We then propose a novel PSA scheme, and prove it is secure. Our new scheme offers better performance than the state-of-the-art PSA schemes.

2.4 Device Management for constrained Cyber Physical Systems

Securing one device by itself can be a challenge. If a larger number of devices in a system shall be secured, the problem is more considerable. If these devices are constrained in nature, and lack of resources to be highly performant and a traditional user interface, a sound system for device management is required.

Device management encompasses the entire life cycle of the devices. From procurement, deployment, operation, access control, monitoring, update and decommissioning. For constrained cyber-physical devices often deployed in large numbers, sometimes in difficult-to-reach locations, remote management is often required. The topic of remote device management is too large to be covered in one thesis. In this thesis we have investigated the problem of *ownership transfer* for constrained wireless devices. Furthermore, we have proposed a Digital Twin-based architecture for device management, with state synchronization and a software update use case. This section will introduce the concepts of ownership transfer and Digital Twins.

2.4.1 Ownership Transfer

Secure ownership transfer is the process of transferring the control of a secure system from one entity to another. The general premise is that each device has some kind of key or credentials; which are shared with the owner. Some kind of server usually represents the owner. Here we will stick to using *key* for any such credential.

The process of transferring the keys from the old owner to the new is not a suitable solution. The terms *New owner privacy* and *Old owner privacy* have been used to describe desirable features [Taq+18]. Old owner privacy is that the new owner shall not be able to decrypt recorded traffic and access data from the old owner. New owner privacy is that the old owner shall not be able to learn secrets from the new owner after the transfer is complete.

The topic of ownership transfer has been studied both for IoT and networked devices but more intensively for RFID-tags. RFID-tags are a relevant problem because RFID-tags attached to things, such as parcels, change hands, and move around. RFID-tags can be read remotely close by the tag; this has raised privacy issues. In [Jue06], the author describes a scenario were RFID-tags carried on a person can be read to reveal sensitive information about their owner. Using keys to enable authorization of RFID-tag access and encryption of the data in transmission has been proposed as a solution to this privacy issue.

When items with RFID-tags that use keys to authorize reading and encrypt the transmitted data change hands, the new owner must be able to access the RFID-tag after the transfer. Ownership transfer is the name given to this problem. The first publication that tries to solve this problem was [SIS05]. Several approaches for ownership transfer exists and different protocols have been proposed for single tag and multiple tag transfer. There is also another aspect of proposed solutions, with protocols featuring a trusted third-party and protocols only involving the old and new owner.

A schematic view of RFID deployment and ownership transfer can be seen in Figure 2.6. One crucial property for RFID-tags is that they are only powered on when they are read, i.e., interrogated. The RFID-tag reader is an essential part of the system since that is the only device that can directly read the RFID-tags. The RFID-tag reader is usually able to do more advanced computation and is not usually limited in energy. Thus it can be used in the system to perform more complicated calculations.

A recent and comprehensive survey of the research into ownership transfer can be found in [Taq+18]. Some ownership transfer solutions for IoT [TN04] use public-key cryptography. However, constrained systems might not be able to handle the complex computations needed for public-key encryption. Besides the computational issue, not all devices might have the memory needed to store the required code.

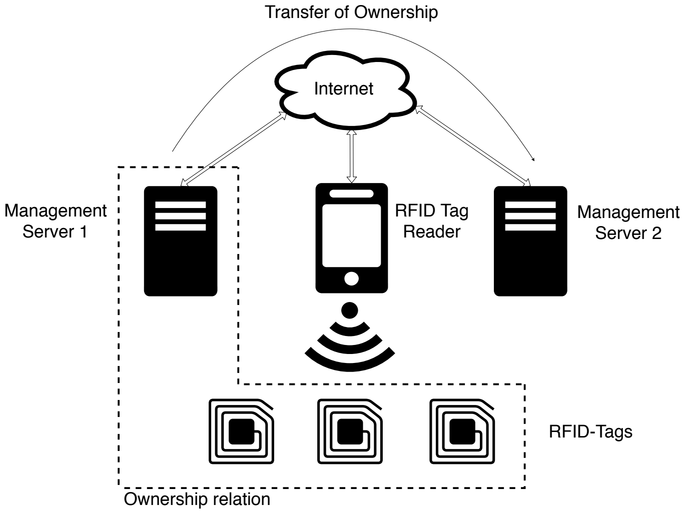


Figure 2.6: RFID-System and ownership transfer

In Figure 2.7, we show a schematic overview of an IoT deployment. The management server does not directly connect to the individual devices, but often communicates over the internet, through some gateway. The presence of the gateway is an essential property of such a system. This gateway sometimes needs to translate protocols and terminate DTLS sessions to work correctly. Since individual IoT devices are connected to the Internet, the attack surface is larger compared to an RFID tag where the attacker must be in proximity to an RFID-tag to be able to communicate with it and to intercept messages. Thus many of the security requirements for RFID-systems can not be directly applied to IoT systems.

Security requirements for secure ownership transfer protocols are the same as for conventional security protocols. Properties such as confidentiality, integrity, and authentication are essential to a secure protocol. But then there are new properties that need to be also considered. According to [Taq+18], the authors have proposed the security requirements stated below. These requirements apply equally to both RFID-tag solutions and IoT protocols since they are general to the problem of ownership transfer:

- New owner privacy: The old owner shall not be able to access data after ownership transfer is completed.
- Old owner privacy: The new owner shall be unable to learn anything that the old owner has done before the transfer.
- Windowing problem: There shall be no place in time where both the new and the old owner has access to the device at the same time.

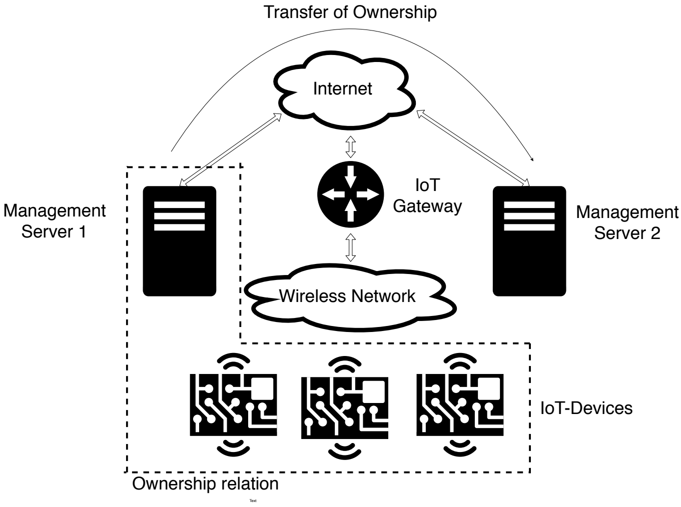


Figure 2.7: IoT deployment and ownership transfer

- Exclusive ownership transfer: It shall be possible to verify that the ownership transfer has gone according to plan.

The property of new owner privacy means that the old owner shall not be able to have access to the device or to intercept and decrypt traffic between the new owner and the device after an ownership transfer has occurred. If the ownership transfer process was designed so that the old owner simply send symmetric keys to the new owner, the old owner could intercept the traffic between the new owner and the device. Suppose that the new owner were to change the symmetric keys, the old owner could decrypt the messages containing the new keys and learn them.

The naive ownership transfer process where the old owner sends symmetric keys to the new owner also violate old owner privacy. The new owner can record messages between the old owner and the device before the ownership transfer took place and decrypt these messages once the new owner gets the symmetric keys. Secure ownership transfer protocols that fulfill these requirements can be designed using public-key cryptography. For scenarios with very constrained devices, where only symmetric cryptography achieving new owner privacy and old owner privacy is more challenging.

The windowing problem means that the transfer must be immediate, so there can be no point in time where both the new owner and the old owner have access to the device that is switching hands. The challenge here is not as apparent; when the step that transfers the ownership occurs, it will either succeed or fail. If it completes, then the New owner will have control of the device, but if it fails, the Old owner shall retain control of the device. This has to be done to prevent the device from becoming *orphaned* and left in a state where neither the New owner or the Old owner can access it. Exclusive ownership transfer means that the New owner shall be able to verify that devices have been transferred and that they are now under the New owner's control. The requirement here is that the new owner must be able to authenticate all devices after a transfer is complete.

2.4.2 Digital Twin

Digital Twin is the name given to techniques where a physical device is *mirrored* to a digital copy. This Digital Twin can then be used to perform computations, such as optimizations that can be implemented on the *physical* twin. Several definitions of the term exist, "A Digital Twin is a real-time digital replica of a physical device" is a succinct definition from [Bac19]. Digital Twin as a concept has its origin in aviation manufacturing, where aircraft engines were one of the first applications. The concept was developed during the 1990s and was published in 2002 [Gri19]. Since then, the application of Digital Twin has spread to Wind Turbines, HVAC (Building Automation), health applications, and many more.

In Figure 2.8, we show how such a workflow with continuous improvements can look. Academia [BR16] has studied the concept of continuous simulation.

In [Gri14b][GV17], the authors present their idea of how to use Digital Twin to facilitate life-cycle management for complex systems. They discuss how to test, simulate, and improve the physical systems using a Digital Twin. But this is not the only application; many industries and fields investigate what benefits they can get from Digital Twins. A summary of these results can be found in [El 18].

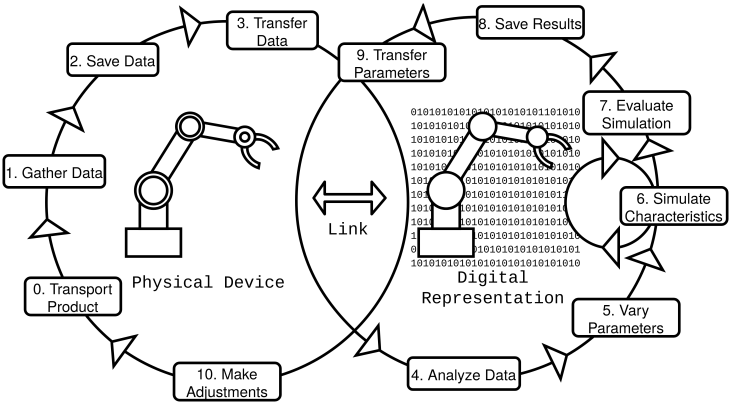


Figure 2.8: A schematic representation of a physical device and its Digital Twin surrounded by the workflow of continuous optimization.

A Digital Twin can also be used to improve security. In heterogeneous systems, it can be challenging to establish a picture of the system. A Digital Twin of the system can provide such a view. This twin can be used for finding vulnerabilities, both by scanning for known vulnerabilities, static threat modeling and also to create a replica of the system to be used in a *Cyber Range*.

ICS systems are often so vulnerable to a cyberattack that techniques used in penetration testing such as port scanning can cause systems to crash. Since these systems are connected to a process, a crash is unacceptable, but stopping the process to do a penetration test is usually not possible either. If this penetration test can be made on a *twin* of the system, it would solve both these problems.

In [Bit+18b], the authors provide a way to generate a Digital Twin of a system that can be used in a penetration test. The Digital Twin can also be used in a cyber range to teach operators of ICS about cybersecurity applied to *their* system.

Digital Twin has been proposed for a variety of areas, such as documentation and continuous improvement. For cybersecurity in industrial control systems and constrained devices, the ability to synchronize the physical device to a Digital Twin can be used to overcome the limitations we described in Section 2.2 and 2.1.

The ability to replicate a state from a device to a remote entity makes it possible to add functionality to the remote entity. This entity can be a cloud environment, and with a state replication protocol, the results of this added security functionality can be *mirrored* to the physical device. We have investigated such a concept in Paper IV, where we propose a simple state synchronization protocol for use in industrial control systems.

2.4.3 State Machine Replication

Finite-state machines can be used for representing and modeling a variety of computer and automation systems. State machines can also be used to design and specify the behavior of a system. State Machine Replication is a technique to synchronize the states between two or more Finite-state machines [Lam84].

Computers and automation systems sometimes fail, either from a flaw in the design and implementation of the system or from an outside malicious attack. Adding redundancy to provide fault-tolerance is one way to overcome this problem. By viewing a part of the system as a state machine and then replicating the state to another part of the system, one can achieve redundancy and reducing the probability of a system-wide outage.

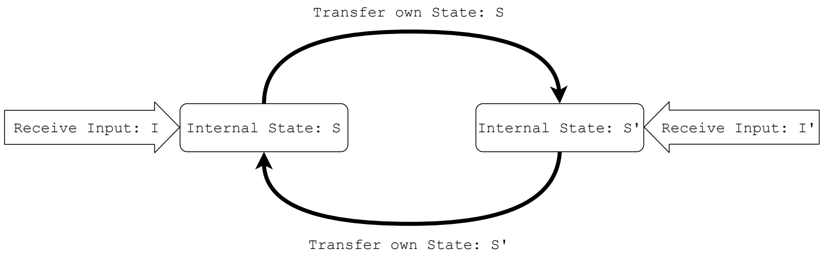


Figure 2.9: A conceptual model of a state replication mechanism

As can be read in [CPS10], Replication and State Machine Replication have been investigated for over 30 years. The techniques have been applied to different fields, such as distributed systems and databases. The goal of replication has been both performance gains, by scaling a system and fault tolerance, by duplication of stored data. By mirroring a physical system with a Digital Twin, a new type of application emerges. Here the goal is to provide a single digital image of a system that can be used for further processing.

Above, Digital Twin was defined as "... a real-time digital replica of a physical device", State machine synchronization is one way of achieving *real-time replication*.

Using replication to improve the security of IoT devices has been suggested in [GA16b]. The authors present a method to *mirror* an IoT device to a server. The server can provide more extensive security mechanisms than the constrained IoT device. By using a rigid communications protocol that only allows for synchronization between the device and the mirror, a high level of security can be achieved for a constrained device.

The technique of state machine replication has been applied to industrial control systems. In [EE18a], the authors propose a state replication mechanism to be used for intrusion detection in ICS. The authors use a state replication approach to avoid prohibitive overhead in terms of network and computation overhead in the physical devices.

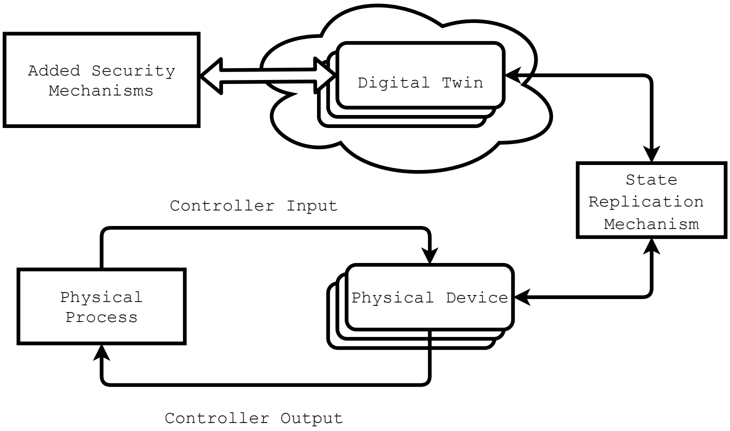


Figure 2.10: Adding security mechanism by replicating the physical devices to Digital Twin and perform the complex security mechanisms there.

Both [GA16b] and [EE18a] propose a similar approach to adding more complex security mechanisms to constrained devices and industrial control systems. In Figure 2.10, we show a system overview of such a solution. For constrained devices, this type of solution is attractive because of the limitations discussed in Section 2.1. A state synchronization protocol can be implemented with small overhead, so this approach is workable. For ICS, the limitations in available resources that we discussed in Section 2.2, the long lifetimes of devices in ICS, and the complexity of these devices can benefit from the added security mechanisms with the relatively low cost of implementing a State Replication Protocol.

In Paper IV, we have used a State machine synchronization protocol to synchronize a physical device with a Digital Twin. The low overhead of a state machine synchronization protocol makes this an attractive solution to realize a Digital Twin for ICS, considering the limitations described in Section 2.2.

Applications of Digital Twins for Security

Since Paper IV was published in 2020, it has been cited over 130 times. Digital Twins are a very active research topic, and Digitals Twins for ICS are no exception. This section will investigate how Paper IV has been cited and derivative works.

Or work has been referenced in works from a wide variety of fields, from ICS [Yan+22; ZFT22], to 6G Edge Cloud [Tan+22; Sun+20], and vehicular networks [ZCZ21; Sun+20]. We will focus here on the most interesting works and the works most closely aligned with ours.

Other researchers have referenced our work proposing Digital Twin architectures for ICS. In [Mih+22], the authors note that the designers of Digital Twins should prioritize security features. The authors claim we are the first to use Digital Twins as an enabler of communications security between the physical plant and the Digital Twin. In [Kas+22], a Digital Twin platform for industrial energy systems is proposed. Our work is used to derive security and safety requirements. The authors of [Zha+21] utilize our state replication mechanism in resilience dynamics and control approach for an electronics assembly line. [Liu+23] remark that our protocol might not be efficient enough for a large-scale ICS deployment. In [Li+21], the authors propose a system for robot-environment interaction monitoring. The authors note that passively updating the model from sensor data is inefficient and that our paper has mitigated this by having an active state synchronization protocol.

Our work has also been cited in more security-related work. The authors of [Lei+21] note that harmful state transitions can be detected. This is the first step to intrusion detection systems. The authors of [Tär+21] directly build on our work and implement a complete system with intrusion detection in a cloud-native Digital Twin. In [Var22], another approach to intrusion detection for ICS using Digital Twins is taken.

The authors of [DEP21] present a system where state replication and recording can be used for forensics in case of an attack. The authors note the close similarity with our state replication protocol. The authors assume that if the physical twin is compromised, the digital twin will also be compromised. The basic idea is to record all states and later replay the state transitions to figure out what happened.

The authors of [Mar+21] use Digital Twins for automotive cybersecurity testing by performing finite state modeling. Our paper is cited as a system that utilizes active state monitoring. The authors also mention similar approaches [EE18a] and [HGM21].

[Vie+21] discuss our state replication model and protocol as a way of constructing Cyber Ranges. The authors note that the proposed state synchronization method might not be suitable for this scenario.

2.5 Formal protocol verification

A protocol is a distributed algorithm with communicating parties. A *security protocol* should also provide *security properties* to the communicating parties. Security properties can be confidentiality or authentication to give two common examples. These properties shall hold even in the presence of hostile actors. A *secure* protocol is a protocol for which some security properties are true. Security protocols take a lot of work to design correctly and to determine if a protocol is *secure* some sort of *verification* of the security properties have to be done. Verification is the process of testing the validity of a property. For the purpose of this thesis, we are only interested in the verification of security properties. For security properties this can be done in a variety of ways, for example, if one finds a counterexample to a property it can be falsified. *Formal verification* is a process that utilize mathematical reasoning and proofs to verify properties. The process of manually verifying security properties is sometimes called *semi-formal* verification.

Many security protocols have been presented and accepted as secure only for attacks to be discovered, sometimes much later. The Needham-Schroeder Public-Key Protocol [NS78] was published in 1978, and an attack was discovered in 1995 [Low95]. Modern, common Wi-Fi is protected by a protocol called WPA2. WPA2 was proposed in 2004, as a response to attacks against WPA [TB09]. However, WPA2 was also vulnerable, one of the vulnerabilities is the KRACK attack [VP17], published in 2017.

Formal verification is a good fit for security protocols as it is a well-contained problem but complex enough to make semi-formal verification error-prone. In recent years formal protocol verification has become a more and more utilized tool in the research and development of security protocols. We have used formal protocol verification with the Tamarin Prover in papers III and V. Formal verification can be used to analyze existing protocols. Paper III presents the first comprehensive formal verification of the security properties of WirelessHART. We found a novel attack in addition to the previously published attacks. Formal protocol verification can also be used to find vulnerabilities when designing new protocols. In paper V, we formally verified security properties of our proposed protocol.

2.5.1 Background and Tools

Research into formal protocol verification started in the late 1970s but picked pace in the 1990s [Mea95].

The first research efforts and tools for formal protocol verification were based on state machine-based analytics and modal logic. In the latter half of the 1990s, algebras started getting research interest.

The first research effort was mainly focusing on secrecy. But authentication became more relevant [Low97]. The security properties of a protocol is verified against an adversary model. One commonly used adversary model is the Dolev-Yao model [DY83]. In the Dolev-Yao model the adversary can learn all transmitted messages in the system. The adversary can drop messages, replay messages, and create arbitrary messages and transmit them to any party in the system. Cryptographic primitives are modeled as being unbreakable for the adversary.

Modern formal protocol verification tools can be categorized by what underlying model they use.

Computational model In the computational model, the adversary is assumed to be a polynomial-time adversary. The most prominent tool in this category is CryptoVerif. CryptoVerif generates proofs as a sequence of games commonly found in cryptographic proofs. The properties that can be proved are secrecy, authentication, and indistinguishability. CryptoVerif works with a bounded number of sessions against an active adversary. The analysis can be automatically or manually guided. It gives a probability bound of an attack. CryptoVerif has been used to verify, among others, SSH [CB13], TLS 1.3 [BBK17], and WireGuard [LBB19]. The benefits of the computational model are the finer granularity of analysis and the probability bound of an attack. The drawback is that protocols are challenging to model.

Symbolic model In the symbolic model messages and data is represented as abstract terms in a free algebra. This is often used with some kind of *rewriting* technique. The rewriting rules can the be used by the constraint solver that produces the output. The most commonly used tools all use the symbolic model.

The AVISPA tool [Vig06] allows users to model protocol using the High-Level Protocol Specification Language (HLPSL). Protocols can be falsified by finding attacks, and *abstraction-based verification* is used for an unbounded number of sessions. DeepSec [CKR18] allow users to model protocols using applied pi calculus [RS11] and is tailored to verify indistinguishability properties with trace equivalence.

ProVerif [Bla+18] is a widely used tool to model protocols and verify security properties. Protocols are modeled in applied pi calculus.

Tamarin is another protocol verifier [Bas+17] that has been widely adopted and used [Bas+22]. We will devote extra time to explain Tamarin, since all formal protocol verifications in this thesis have been done using Tamarin.

In [Che+22] the author compares Tamarin, ProVerif, and DeepSec. The authors state that ProVerif usually offers better automation and speed compared to Tamarin but might find false attacks or hinders attack reconstruction. Tamarin guarantees correctness and no false attacks if it terminates. ProVerif has to be manually guided, which is a challenging task. Tamarin also has a GUI that can be used to guide a proof, and the process can be automated with oracles. Tamarin has more cryptographic primitives, more precise Diffie-Hellman, and exclusive or (XOR).

2.5.2 Current research Efforts in Formal Protocol Verification

Formal protocol verification is an active research field. Research is done on tools and by utilizing the tools. In this section, we will briefly examine both areas' current state-of-the-art.

Improvements and Development of Tools The development of tools for formal protocol verification is an active research area. Research is done on improving existing tools by enhancing their performance, expanding their capabilities, and increasing their ease of use. Researchers are also publishing new tools.

In [BCC22] the authors present an overhaul of ProVerif. ProVerif is extended with lemmas, axioms, natural numbers, and temporal queries, among other features. These features are intended to reduce the number of false attacks. Furthermore optimizations result in faster execution times.

Tamarin prover might not find a prof and continue running, perhaps indefinitely, without finding a proof or a counterexample. To prevent this, researchers can write *source lemmas* to help Tamarin. The authors of [Cor+22] propose a method of automatically generating source lemmas. With this extension, Tamarin can automatically analyze certain protocols that previously had to be manually guided.

In [Che+22], the authors propose SAPIC+, a protocol modeling language that can be automatically translated for use by Tamarin, ProVerif, and DeepSec. This promising line of research lets researchers model protocols once in a common language and utilize the strengths of each of the tools to their advantage.

One prevailing drawback of formal protocol verification tools is their complexity. To be feasible to use outside an academic setting or organizations with considerable resources, easy-to-use tools are necessary. VerifPal is one such tool [KNT20] that simplifies the modeling and verification of protocols. VerifPal outputs a graphic representation of the modeled protocol that can aid the tool's user. The tool also outputs a text representation of an attack on a stated security property. VerifPal is a new software in the beta stage. In [Zha+20] the authors have verified the security properties of QUIC [Lan+17] using both ProVerif and VerifPal. The authors found a man-in-the-middle attack on the QUIC handshake protocol. The attack was found both by ProVerif and VerifPal, but VerifPal required a longer execution time.

In descriptions of protocols in papers Alice and Bob is often used to denote the communicating parties A and B. Describing protocols using Alice&Bob notation aid the reader in understanding the protocol. In [Bas+15] the authors propose a formal Alice&Bob notation and a way of automatically translating Alice and Bob notation into Tamarin's input specification language. The idea of formally defining the Alice&Bob notation so that the security properties of the protocol can be formally verified is a very intriguing prospect. Tamgram [Li22] has been developed as a new high level protocol specification language for Tamarin. The new language is intended to simplify the modeling of complex protocols.

Current Formal Protocol Verification Activity New tools are being developed, and the tools are used to verify both new and old protocols. In [Mea95], the author notes that most formal verification has been done on existing protocols. She also remarks that it would be more efficient to do formal verification when the protocol is designed. This was stated in 1995 and has gained traction only in the last ten years with TLS1.3 and 5G AKA.

Tamarin lets the user model a protocol, state the security properties of the protocol, and then verify the security properties using automated reasoning. In this section, we will give a cursory introduction to the use of Tamarin. A (simplified) Tamarin workflow can look something like the following:

1. Study the protocol definition.
2. Model the protocol using Tamarin rules.
3. State security properties as lemmas.
4. Run the Tamarin prover to verify lemmas.

Protocols can be modeled using *rules* or the SAPIC language. For the papers included in this thesis, we have only used rules. So we will leave SAPIC out of scope for this thesis. When the protocol is modeled, security properties, called *lemmas*, are stated. Lemmas are *trace properties*. A trace is a labeled fact in a rule that happened at a specific time. It can be that a message was sent or a key was derived. Lemmas can be properties like secrecy of messages or keys and authentication properties. We will discuss lemmas and the verification of protocols further in the next section.

Tamarin rules consist of a precondition and a postcondition. The example rule below generates a *fresh* variable `~i_sk`. The postcondition is the *facts* `Initiator($I,~i_sk)` and `Out('g'~ i_sk)`. The `Initiator(...)` fact can be used as a precondition for further rules. `Out(...)` sends out the public key. The adversary knows the key but can also be received by the legitimate recipient *Responder*. Now that the Diffie-Hellman protocol has been modeled, security properties must be specified.

```
rule r1:
  [Fr(~i_sk)]
  -->
  [!Initiator($I, ~i_sk), Out(‘g’^~i_sk)]
```

In the second rule, the `In(...)` precondition requires a message for the rule to execute. The responder generates its secret key and a session key `session_key`. The responder then sends its public key to the initiator.

```
rule r2:
  let:
  session_key = i_pk^~r_sk
  [In(i_pk), Fr(~r_sk)]
  --[ Resp_S(session_key) ]->
  [!Responder($R,~r_sk, session_key), Out(‘g’^~r_sk)]
```

In the third rule, the Initiator receives the responder's public key and computes its own session key.

```
rule r3:
  [In(r_pk), !Initiator($I, ~i_sk)]
  --[ Init_S(r_pk^~i_sk) ]->
  [!Initiator_1($I, ~i_sk, r_pk^~i_sk)]
```

In rule `r2` and `r3` *action facts* can be found. Action facts are not used as preconditions during the execution of the protocol, but is used to state security properties, as we will show in the next section. The action fact `Resp_S(session_key)` and `Init_S(session_key)` denote that the responder and initiator has derived the session key `session_key`.

2.5.3 Verifying Security Properties with Tamarin

As we introduced in the previous section, security properties are specified using lemmas in Tamarin. Lemmas are expressed as first-order logic formulas over action facts and timepoints, called traces.

In the first equation below, we show a sample lemma. The lemma states, in English, that for all traces, there exists one session key, `Ses_K` and two timepoints `#i` and `#j`. If the Responder has derived the session key `Ses_K` at time `#i` and the initiator derived the session key `Ses_K` at `#j`, and that `#i` happened before `#j`. This implies that there does not exists a timepoint #k where the adversary knows `Ses_K`.

```
lemma session_key_secrecy:
  “All Ses_K #i #j.
  Resp_S(Ses_K) @ #i
  & Init_S(Ses_K) @ #j &
  #j < #i ==>
  not(Ex #k. K(Ses_K) @ #k)”
```

Verifying the lemma session_key_secrecy using Tamarin yields a proof. If both the initiator and the responder derive the same session key, the adversary cannot learn it.

In this second lemma, we show a lemma very similar to the one presented above. The difference is that the session keys are expressed as two different variables.

```
lemma MITM_impossible:
  “All Ses_K1 Ses_K2 #i #j.
  (Resp_S(Ses_K1) @ #i
  & Init_S(Ses_K2) @ #j &
  #j < #i) ==>
  not(Ex #k1 #k2. K(Ses_K1) @ #k1 | K(Ses_K2) @ #k2)”
```

Tamarin will find a counterexample to this lemma. The counterexample is a man-in-the-middle attack on Diffie-Hellman. These lemmas show the nuance in which security properties are stated and the importance of careful protocol modeling and property specification.

Contributions and Conclusions

3.1 Contributions

The following sections introduce each contribution, the individual contributions of the author, and the changes made to the publications for print in this thesis.

Author names and acronyms: Joakim Brorsson (JB), Christian Gehrmann (CG), Martin Gunnarsson (MG), Rikard Höglund (RH), Krzysztof Mateusz Malarski (KMM), Francesca Palombini (FP), Ludwig Seitz (LS), Marco Tilcoa (MT).

3.1.1 Evaluating the Performance of the OSCORE Security Protocol in Constrained IoT Environments

Content

OSCORE is a protocol recently standardized by the IETF. It is a protocol for Constrained devices that used the Object security concept to protect CoAP messages. We have discussed the limitations of Constrained devices in Section 2.1 and the concept of Object security in Section 2.3.2. In this work, we have evaluated the first constrained implementation of OSCORE and compared it against DTLS 1.2, the state of the art solution for protecting CoAP messages. Our evaluation showed that OSCORE has performance comparable to DTLS 1.2 when implemented on a constrained device.

Individual Contribution

MG wrote the constrained OSCORE implementation. MG and JB performed the performance evaluation. MG, JB, MT, FP, LS, collaborated in writing the background and the description of OSCORE.

For this Thesis

The paper has been formatted to match the rest of the thesis.

3.1.2 Performance Evaluation of Group OSCORE for Secure Group Communication in the Internet of Things

Content

Group OSCORE is a protocol being standardized by the IETF. Group OSCORE is an extension of the OSCORE protocol, with support for group communication. To preserve source authentication in group communications public-key cryptography must be used. Public-key cryptography adds extra computational complexity to message processing, and increase the energy consumption of devices. In this work, we have evaluated the first constrained implementation of Group OSCORE and compared it against unicast CoAP, CoAP over multicast and OSCORE. We have implemented and evaluated OSCORE and Group OSCORE using both software cryptography and hardware-accelerated cryptography. Our evaluation showed that Group OSCORE can, with the right implementation of public-key cryptography be feasible in constrained devices.

Individual Contribution

MG and KMM wrote the constrained Group OSCORE implementation. MG designed and performed the performance evaluation. MG, KMM, MT, and RH collaborated in writing the background and the description of Group OSCORE.

For this Thesis

The paper has been formatted to match the rest of the thesis.

3.1.3 Formal Verification of the WirelessHART Protocol - Verifying Old and Finding New Attacks

Content

WirelessHART is a protocol specified by the HART foundation and standardized by the IEC. It is intended to replace the HART protocol in industrial control systems. The security of WirelessHART has been studied before, but no comprehensive formal verification has been done. In this paper we have modeled the WirelessHART protocol using Tamarin. We have demonstrated previously published attacks and demonstrated a novel *malicious-re-keying attack*. In this new attack an adversary can change the keys of devices in the system to prevent legitimate communication with the devices.

Individual Contribution

MG was the sole author and did all the work for this paper.

For this Thesis

The paper has been formatted to match the rest of the thesis.

3.1.4 A Digital Twin Based Industrial Automation and Control System Security Architecture

Content

In this paper, we have proposed a novel security architecture for industrial control systems based on the concept of Digital Twin. Digital Twin is a concept that has been previously used for process simulation and continuous optimization. We have discussed the concept of Digital Twins in detail in Section 2.4.2.

We proposed a way to utilize Digital Twins to automate security mechanisms that provide scanning of firmware for vulnerabilities and automated patching of industrial control systems. By using Digital Twins and state machine synchronization, we have shown that it is possible to offload complex security mechanisms to a remote Digital Twin. This can be used to overcome the limitations in industrial control systems that operate under strict real-time deadlines, as we described in Section 2.2.

Individual Contribution

CG designed the Digital Twin replication model and security architecture. CG performed the security analysis. MG implemented the state synchronization protocol and performed the performance evaluation.

For this Thesis

The paper has been formatted to match the rest of the thesis.

3.1.5 Secure Ownership Transfer for Resource Constrained IoT Infrastructures

Content

In this paper, we investigated the problem of secure ownership transfer. The process of transferring ownership of devices has mainly been studied for RFID-tags but not for IoT devices. The core problem with ownership transfer is *new owner privacy* and *old owner privacy*. After the transfer of ownership, the new owner shall be unable to learn anything that has happened on the device or any message sent. The old owner shall not learn anything the new owner does after the transfer.

The work that has been done on secure ownership transfer for IoT has focused on solutions using public-key cryptography. In our intended system, the devices we consider for ownership transfer are constrained devices, as described in Section 2.1. Because of the limitations in performance, we have developed a secure ownership transfer protocol using symmetric-key cryptography.

Individual Contribution

MG has together with CG, designed the secure ownership transfer protocol. CG stated security requirements and, together with MG, did the security analysis of the protocol. MG did the Tamarin model and formal verification of the protocol. MG implemented the experimental evaluation and produced the experimental results. MG extended the original version of the paper with more related work, security analysis and experimental results, for the journal publication.

For this Thesis

The paper has been formatted to match the rest of this thesis.

3.1.6 DIPSAUCE: Efficient Private Stream Aggregation Without Trusted Parties

Content

Private Stream Aggregation (PSA) is a subset of functional encryption that allow participants in a protocol to send labeled and encrypted messages to a central *aggregator*. The aggregator can sum the ciphertexts from each individual participant in the protocol. If all ciphertexts are summed the aggregator is left with the sum of the individual messages each participant sent. This allow time-series data, to be collected while still not revealing consumption of individual households. Several PSA protocols have been proposed. A common weakness is that they utilize a centralized setup where a trusted party generates all keying material. Furthermore the performance of the schemes have been evaluated and found not suitable to constrained devices. In this work we have proposed a novel PSA scheme called DIPSAUCE. DIPSAUCE does not rely on a centralized setup, and outperform previously published PSA schemes.

Individual Contribution

JB and MG surveyed current state of the art in private stream aggregation. MG implemented the current state of the art protocols and evaluated their performance. JB and MG designed a, more efficient, protocol for private stream aggregation. JB proved the security of our proposed protocol. MG implemented the protocol and evaluated its efficiency.

For this Thesis

The paper has been formatted to match the rest of the thesis.

3.2 Conclusions

In this thesis, we have investigated protocols for constrained cyber-physical devices. Although many protocols we have studied and developed are general, our primary focus has been on future industrial applications. Industry 4.0 is a concept where increased connectivity and data-sharing, together with new technologies such as cloud computing, can increase the productivity of industrial systems. We focused on edge devices in these networks; many such devices are limited in terms of performance and can be categorized as Constrained devices.

Efficient security protocols has been the main topic of this thesis. The research within this area can be divided into two topics; efficient, secure communications protocols for Constrained devices and security life cycle management for industrial control systems and Wireless Sensor nodes.

The first paper on communication protocols for constrained wireless devices evaluated the OSCORE protocol. We evaluated the recently standardized protocol OSCORE against the current state-of-the-art method of securing CoAP messages, DTLS1.2. We found that OSCORE performs roughly equal to DTLS in terms of computational complexity, while OSCORE has lower network per-message overhead. At the same time, unlike DTLS, OSCORE provides end-to-end message protection in the presence of (untrusted) intermediaries. That is, more security assurances with a similar communication overhead.

The second paper on the same topic follows up on the previous work with an evaluation of Group OSCORE. The proposed standard Group OSCORE extends OSCORE to protect CoAP messages exchanged in a group communication setup. This includes one-to-one messages, as well as one-to-many messages sent over, for example, IPv6 multicast. To achieve source authentication in a group communication scenario, digital signatures are utilized. Such signatures are computationally intensive to compute and verify. To evaluate Group OSCORE's suitability for deployment in constrained devices, we performed an extensive evaluation suite on two hardware platforms with two different signature schemes. We also evaluated the performance of cryptographic operations implemented in software against the hardware accelerated. Since there was no state-of-the-art protocol to evaluate against, we evaluated against CoAP, CoAP over IPv6 Multicast, and OSCORE. With a fast implementation of digital signatures in either software or hardware, performance could be suitable for deployment in constrained networks.

Papers I and II were experimental evaluations of recently standardized protocols and protocols under standardization. Paper III provides a formal protocol verification of the WirelessHART protocol. WirelessHART is intended for industrial control systems to communicate with strong requirements for round-trip-time. This has influenced the designers of WirelessHART only to use symmetric cryptography to achieve this performance. Although attacks have been published on WirelessHART, no formal verification has been done. In this work, we have done such a verification, modeling all relevant parts of WirelessHART and verifying all previously published attacks. During our analysis, we also discovered a new attack. This novel *malicious re-keying* attack enables an attacker to leverage the credentials of a compromised device to trick other devices into changing their encryption keys. Since the remote party still uses the old, legitimate keys, the remote party cannot contact the devices. This enables an attacker to perform a denial-of-service attack on larger parts of the network with only one compromised device.

Regarding security life cycle management, we have presented a novel security architecture for industrial control systems using Digital Twin. We have evaluated the state synchronization protocol that synchronizes the physical devices with the Digital Twin and found it lightweight and suitable for industrial control systems. This paper has been well cited, and the techniques and architecture we propose have been used in a wide variety of settings. Papers proposing new architectures for ICS, 6G edge-clouds, and ICS intrusion detection systems have cited our work.

We have also presented a protocol to securely transfer the ownership of constrained wireless devices from one owner to a new owner. The protocol uses a Trusted Third Party to enable symmetric cryptography while providing the desired security properties. The protocol was formally verified to prove that the stated security requirements hold, and it was evaluated in terms of performance. We found it to be suitable for deployment in constrained environments.

Constrained cyber-physical systems are not only deployed in ICS environments, and connected production can connect the producer to the end-users. Consider the Smart Grid, where household electricity consumption is measured to balance and plan power generation. In such a scenario, a household's measured electricity consumption does not only have to be confidential from an outside adversary but also from the entity collecting the data. The privacy of the users has to be preserved. In the last paper, we present a new protocol that provides privacy-preserving stream aggregation for sensor data in a wireless sensor network. We show that the proposed protocol can be implemented more efficiently than in state-of-the-art solutions for constrained devices.

References

[Abd+19] M. Abdalla et al. “Decentralizing inner-product functional encryption”. In: *IACR International Workshop on Public Key Cryptography*. Berlin, Germany: Springer, 2019, pp. 128–157.

[All10] Z. Alliance. “Zigbee alliance”. In: *WPAN industry group* (2010).

[All22] C. S. Alliance. *Matter 1.0 Core Specification*. Tech. rep. Connectivity Standards Alliance, 2022.

[Bac19] G. Bacchiega. *Developing and Embedded Digital Twin*. June 2019.

[Bar14] R. Barnes. “Use cases and requirements for JSON object signing and encryption (JOSE)”. In: *Internet Eng. Task Force, Fremont, CA, USA, RFC* 7165 (2014).

[Bas+15] D. Basin et al. “Alice and Bob meet equational theories”. In: *Logic, Rewriting, and Concurrency*. Springer, 2015, pp. 160–180.

[Bas+17] D. Basin et al. “Symbolically Analyzing Security Protocols using Tamarin”. In: *ACM SIGLOG News* (Oct. 2017).

[Bas+22] D. Basin et al. “Tamarin: Verification of Large-Scale, Real-World, Cryptographic Protocols”. In: *IEEE Security & Privacy* 20.3 (2022), pp. 24–32.

[BBK17] K. Bhargavan, B. Blanchet, and N. Kobeissi. “Verified models and reference implementations for the TLS 1.3 standard candidate”. In: *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2017, pp. 483–502.

[BCC22] B. Blanchet, V. Cheval, and V. Cortier. “Proverif with lemmas, induction, fast subsumption, and much more”. In: *42nd IEEE Symposium on Security and Privacy (S&P’22)*. 2022.

[Ber+15] D. J. Bernstein et al. “TweetNaCl: A crypto library in 100 tweets”. In: *International Conference on Cryptology and Information Security in Latin America*. Springer. 2015, pp. 64–83.

[BGZ18] D. Becker, J. Guajardo, and K.-H. Zimmermann. "Revisiting Private Stream Aggregation: Lattice-Based PSA." In: *NDSS*. Reston, VA, USA: Internet Society, 2018.

[BH20] C. Bormann and P. E. Hoffman. *Concise Binary Object Representation (CBOR)*. RFC 8949. Dec. 2020.

[Bit+18b] R. Bitton et al. "Deriving a cost-effective digital twin of an ICS to facilitate security evaluation". In: *European Symposium on Research in Computer Security*. Springer. 2018, pp. 533–554.

[BJL16] F. Benhamouda, M. Joye, and B. Libert. "A new framework for privacy-preserving aggregation of time-series data". In: *ACM Transactions on Information and System Security (TISSEC)* 18.3 (2016), pp. 1–21.

[Bla+18] B. Blanchet et al. "ProVerif 2.00: automatic cryptographic protocol verifier, user manual and tutorial". In: *Version from* (2018), pp. 05–16.

[Bor+22] C. Bormann et al. *Terminology for Constrained-Node Networks*. Internet-Draft draft-ietf-lwig-7228bis-00. Work in Progress. Internet Engineering Task Force, June 2022. 27 pp.

[BR16] S. Boschert and R. Rosen. "Digital twin—the simulation aspect". In: *Mechatronic futures*. Springer, 2016, pp. 59–74.

[Bro+00] M. Brown et al. "PGP in Constrained Wireless Devices." In: *USENIX Security Symposium*. 2000.

[BSW11] D. Boneh, A. Sahai, and B. Waters. "Functional encryption: Definitions and challenges". In: *Theory of Cryptography Conference*. Berlin, Germany: Springer, 2011, pp. 253–273.

[CB13] D. Cadé and B. Blanchet. "From Computationally-Proved Protocol Specifications to Implementations and Application to SSH." In: *J. Wirel. Mob. Networks Ubiquitous Comput. Dependable Appl.* 4.1 (2013), pp. 4–31.

[Che+22] V. Cheval et al. *Sapic+: protocol verifiers of the world, unite!* Cryptology ePrint Archive, Paper 2022/741. `https://eprint.iacr.org/2022/741`. 2022.

[Cho+18] J. Chotard et al. "Decentralized multi-client functional encryption for inner product". In: *International Conference on the Theory and Application of Cryptology and Information Security*. Berlin, Germany: Springer, 2018, pp. 703–732.

[Cho+20] J. Chotard et al. "Dynamic decentralized functional encryption". In: *Annual International Cryptology Conference*. Berlin, Germany: Springer, 2020, pp. 747–775.

[CKR18] V. Cheval, S. Kremer, and I. Rakotonirina. “The DEEPSEC prover”. In: *International Conference on Computer Aided Verification*. Springer. 2018, pp. 28–36.

[Cor+22] V. Cortier et al. “Automatic generation of sources lemmas in TAMARIN: towards automatic proofs of security protocols”. In: *Journal of Computer Security* 30.4 (Aug. 2022), pp. 573–598.

[CPS10] B. Charron-Bost, F. Pedone, and A. Schiper. “Replication”. In: *LNCS* 5959 (2010), pp. 19–40.

[Cro+95] S. Crocker et al. *MIME Object Security Services*. RFC 1848. RFC Editor, Oct. 1995.

[CSS12] T.-H. H. Chan, E. Shi, and D. Song. “Privacy-preserving stream aggregation with fault tolerance”. In: *International Conference on Financial Cryptography and Data Security*. Berlin, Germany: Springer, 2012, pp. 200–214.

[De +08] G. De Meulenaer et al. “On the energy cost of communication and cryptography in wireless sensor networks”. In: *2008 IEEE International Conference on Wireless and Mobile Computing, Networking and Communications*. IEEE. 2008, pp. 580–585.

[DEP21] M. Dietz, L. Englbrecht, and G. Pernul. “Enhancing Industrial Control System Forensics using replication-based Digital Twins”. In: *IFIP International Conference on Digital Forensics*. Springer. 2021, pp. 21–38.

[DY83] D. Dolev and A. Yao. “On the Security of Public Key Protocols”. In: *IEEE Transactions on information theory* 29.2 (1983), pp. 198–208.

[EE18a] M. Eckhart and A. Ekelhart. “A Specification-based State Replication Approach for Digital Twins”. In: *Proceedings of the 2018 Workshop on Cyber-Physical Systems Security and PrivaCy*. CPS-SPC ’18. Toronto, Canada: ACM, 2018, pp. 36–47.

[EK21] J. Ernst and A. Koch. “Private Stream Aggregation with Labels in the Standard Model.” In: *Proc. Priv. Enhancing Technol.* 2021.4 (2021), pp. 117–138.

[El 18] A. El Saddik. “Digital Twins: The Convergence of Multimedia Technologies”. In: *IEEE MultiMedia* 25.2 (Apr. 2018), pp. 87–92.

[Emu17] K. Emura. “Privacy-preserving aggregation of time-series data with public verifiability from simple assumptions”. In: *Australasian Conference on Information Security and Privacy*. Berlin, Germany: Springer, 2017, pp. 193–213.

[For+17] F. Forsby et al. “Lightweight x. 509 digital certificates for the internet of things”. In: *Interoperability, Safety and Security in IoT*. Springer, 2017, pp. 123–133.

[GA16b] C. Gehrmann and M. A. Abdelraheem. “IoT protection through device to cloud synchronization”. In: *2016 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*. IEEE. 2016, pp. 527–532.

[Gid+20] M. Gidlund et al. “Guest Editorial: Security, privacy, and trust for industrial internet of things”. In: *IEEE Transactions on Industrial Informatics* 16.1 (2020), pp. 625–628.

[GK16] D. Gollmann and M. Krotofil. “Cyber-Physical Systems Security”. In: *The New Codebreakers: Essays Dedicated to David Kahn on the Occasion of His 85th Birthday*. Ed. by P. Y. A. Ryan, D. Naccache, and J.-J. Quisquater. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 195–204.

[GKS18] O. Garcia-Morchon, S. Kumar, and M. Sethi. *State-of-the-Art and Challenges for the Internet of Things Security*. Internet-Draft draft-irtf-t2trg-iot-seccons-16. `http://www.ietf.org/internet-drafts/draft-irtf-t2trg-iot-seccons-16.txt`. IETF Secretariat, Dec. 2018.

[Gri14b] M. Grieves. “Digital twin: manufacturing excellence through virtual factory replication”. In: *White paper* 1 (2014), pp. 1–7.

[Gri19] M. W. Grieves. “Virtually Intelligent Product Systems: Digital and Physical Twins”. In: *Complex Systems Engineering: Theory and Practice* (2019), pp. 175–200.

[Gun20] Gunnarsson, Martin. *Security Solutions for Constrained Devices in Cyber-Physical Systems*. eng. Licentiate Thesis. Mar. 2020.

[GV17] M. Grieves and J. Vickers. “Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems”. In: *Transdisciplinary perspectives on complex systems*. Springer, 2017, pp. 85–113.

[Haa00] J. C. Haartsen. “The Bluetooth radio system”. In: *IEEE personal communications* 7.1 (2000), pp. 28–36.

[HGM21] T. Hossen, M. Gursoy, and B. Mirafzal. “Digital twin for self-security of smart inverters”. In: *2021 IEEE Energy Conversion Congress and Exposition (ECCE)*. IEEE. 2021, pp. 713–718.

[HH12] R. Heydon and N. Hunn. “Bluetooth low energy”. In: *CSR Presentation, Bluetooth SIG https://www. bluetooth. org/DocMan/handlers/DownloadDoc. ashx* (2012).

[HTS08] 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.

[IEE11] IEEE Computer Society. *IEEE Standard for Local and Metropolitan Area Networks, Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)*. Oct. 2011.

[Ima+13] T. Imamura et al. "XML encryption syntax and processing version 1.1". In: *W3C, Recommendation* (2013).

[JBS15] M. Jones, J. Bradley, and N. Sakimura. "JSON web signature (JWS)". In: *Internet Requests for Comments, RFC* 7515 (2015).

[JL13] M. Joye and B. Libert. "A scalable scheme for privacy-preserving aggregation of time-series data". In: *International Conference on Financial Cryptography and Data Security*. Berlin, Germany: Springer, 2013, pp. 111–125.

[Jue06] A. Juels. "RFID security and privacy: A research survey". In: *IEEE journal on selected areas in communications* 24.2 (2006), pp. 381–394.

[Kag+13] H. Kagermann et al. *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*. Forschungsunion, 2013.

[Kal15] P. Kalvoda. "Implementace a evaluace protokolu CBOR". In: (2015).

[Kas+22] L. Kasper et al. "Toward a Practical Digital Twin Platform Tailored to the Requirements of Industrial Energy Systems". In: *Applied Sciences* 12.14 (2022), p. 6981.

[KKC19] H.-S. Kim, S. Kumar, and D. E. Culler. "Thread/OpenThread: A compromise in low-power wireless multihop network architecture for the Internet of Things". In: *IEEE Communications Magazine* 57.7 (2019), pp. 55–61.

[KKG19] M. Krotofil, K. Kursawe, and D. Gollmann. "Securing industrial control systems". In: *Security and Privacy Trends in the Industrial Internet of Things* (2019), pp. 3–27.

[KNT20] N. Kobeissi, G. Nicolas, and M. Tiwari. "Verifpal: Cryptographic protocol analysis for the real world". In: *International Conference on Cryptology in India*. Springer. 2020, pp. 151–202.

[Koo15] P. Koopman. *Embedded $ystemEngineering Economics*. Oct. 2015.

[Lam84] L. Lamport. “Using Time Instead of Timeout for Fault-Tolerant Distributed Systems”. In: *ACM Transactions on Programming Languages and Systems* (Apr. 1984), pp. 254–280.

[Lan+17] A. Langley et al. “The quic transport protocol: Design and internet-scale deployment”. In: *Proceedings of the conference of the ACM special interest group on data communication*. 2017, pp. 183–196.

[LBB19] B. Lipp, B. Blanchet, and K. Bhargavan. “A mechanised cryptographic proof of the WireGuard virtual private network protocol”. In: *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE. 2019, pp. 231–246.

[Lei+21] Z. Lei et al. “Toward a web-based digital twin thermal power plant”. In: *IEEE Transactions on Industrial Informatics* 18.3 (2021), pp. 1716–1725.

[LEM14] I. Leontiadis, K. Elkhiyaoui, and R. Molva. “Private and dynamic time-series data aggregation with trust relaxation”. In: *International Conference on Cryptology and Network Security*. Berlin, Germany: Springer, 2014, pp. 305–320.

[LF21] F. Lundberg and J. Feljan. “Julia: fast and secure key agreement for IoT devices”. In: *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks*. 2021, pp. 90–99.

[Li+21] X. Li et al. “Semantic-enhanced digital twin system for robot–environment interaction monitoring”. In: *IEEE Transactions on Instrumentation and Measurement* 70 (2021), pp. 1–13.

[Li22] D. L. Li. *Tamgram*. `https://darrenldl.github.io/tamgram/`. 2022.

[Liu+23] S. Liu et al. “A blockchain-based interactive approach between digital twin-based manufacturing systems”. In: *Computers & Industrial Engineering* 175 (2023), p. 108827.

[LJ19] F. Lundberg and S. Johansson. *Specification of Salt Channel v2*. Tech. rep. Stockholm, Sweden: ASSA ABLOY AB, 2019.

[Low95] G. Lowe. “An Attack on the Needham- Schroeder Public- Key Authentication Protocol”. In: *Information processing letters* 56.3 (1995).

[Low97] G. Lowe. “A hierarchy of authentication specifications”. In: *Proceedings 10th computer security foundations workshop*. IEEE. 1997, pp. 31–43.

[Mar+21] S. Marksteiner et al. "Using Cyber Digital Twins for Automated Automotive Cybersecurity Testing". In: *2021 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. IEEE. 2021, pp. 123–128.

[Mat+23] J. P. Mattsson et al. *CBOR Encoded X.509 Certificates (C509 Certificates)*. Internet-Draft draft-ietf-cose-cbor-encoded-cert-05. Work in Progress. Internet Engineering Task Force, Jan. 2023. 57 pp.

[Mea95] C. A. Meadows. "Formal verification of cryptographic protocols: A survey". In: *International Conference on the Theory and Application of Cryptology*. Springer. 1995, pp. 133–150.

[MF11] A. Melnikov and I. Fette. *The WebSocket Protocol*. RFC 6455. Dec. 2011.

[Mih+22] S. Mihai et al. "Digital twins: a survey on enabling technologies, challenges, trends and future prospects". In: *IEEE Communications Surveys & Tutorials* (2022).

[MM16] P. McLaughlin and R. McAdam. "The undiscovered Country: The future of industrial automation". In: *Honeywell Process Solutions. Honeywell* (2016).

[NS78] R. M. Needham and M. D. Schroeder. "Using encryption for authentication in large networks of computers". In: *Communications of the ACM* 21.12 (1978), pp. 993–999.

[Per08] C. Perrin. "The CIA triad". In: *Dostopno na* (2008).

[Per18] T. Perrin. "The Noise protocol framework". In: *PowerPoint Presentation* (2018).

[Rat+16] R. Ratasuk et al. "NB-IoT system for M2M communication". In: *2016 IEEE wireless communications and networking conference*. IEEE. 2016, pp. 1–5.

[Res18] E. Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.3*. RFC 8446. RFC Editor, Aug. 2018.

[RS11] M. D. Ryan and B. Smyth. "Applied pi calculus". In: *Formal Models and Techniques for Analyzing Security Protocols*. Ios Press, 2011, pp. 112–142.

[Sau+11] T. Sauter et al. "The evolution of factory and building automation". In: *IEEE Industrial Electronics Magazine* 5.3 (2011), pp. 35–48.

[SBC19] W. Saad, M. Bennis, and M. Chen. "A vision of 6G wireless systems: Applications, trends, technologies, and open research problems". In: *IEEE network* 34.3 (2019), pp. 134–142.

[Sch22a] J. Schaad. *CBOR Object Signing and Encryption (COSE): Initial Algorithms*. RFC 9053. Aug. 2022.

[Sch22b] J. Schaad. *CBOR Object Signing and Encryption (COSE): Structures and Process*. RFC 9052. Aug. 2022.

[Sel+19a] G. Selander et al. *Object Security for Constrained RESTful Environments (OSCORE)*. RFC 8613. RFC Editor, July 2019.

[SFS11] K. A. Stouffer, J. A. Falco, and K. A. Scarfone. *Sp 800-82. guide to industrial control systems (ics) security: Supervisory control and data acquisition (scada) systems, distributed control systems (dcs), and other control system configurations such as programmable logic controllers (plc)*. 2011.

[SHB14] Z. Shelby, K. Hartke, and C. Bormann. *The Constrained Application Protocol (CoAP)*. RFC 7252. June 2014.

[Shi+11] E. Shi et al. “Privacy-Preserving Aggregation of Time-Series Data”. In: *Network and Distributed System Security Symposium, NDSS* 1 (2011), p. 17.

[SIS05] J. Saito, K. Imamoto, and K. Sakurai. “Reassignment Scheme of an RFID Tag’s Key for Owner Transfer”. In: *International Conference on Embedded and Ubiquitous Computing*. Springer. 2005, pp. 1303–1312.

[Sor+15] N. Sornin et al. “LoRa Specification 1.0”. In: *Lora Alliance Standard specification., Jan* (2015).

[Sun+20] W. Sun et al. “Reducing offloading latency for digital twin edge networks in 6G”. In: *IEEE Transactions on Vehicular Technology* 69.10 (2020), pp. 12240–12251.

[SWW15] A.-R. Sadeghi, C. Wachsmann, and M. Waidner. “Security and privacy challenges in industrial internet of things”. In: *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*. IEEE. 2015, pp. 1–6.

[Tak+23] J. Takeshita et al. “TERSE: Tiny Encryptions and Really Speedy Execution for Post-Quantum Private Stream Aggregation”. In: *Security and Privacy in Communication Networks*. Ed. by F. Li et al. Cham: Springer Nature Switzerland, 2023, pp. 331–352.

[Tan+22] F. Tang et al. “Survey on digital twin edge networks (DITEN) toward 6G”. In: *IEEE Open Journal of the Communications Society* 3 (2022), pp. 1360–1381.

[Taq+18] E. Taqieddin et al. “Tag Ownership Transfer in Radio Frequency Identification Systems: A Survey of Existing Protocols and Open Challenges”. In: *IEEE Access* (2018).

[Tär+21] W. Tärneberg et al. “Prototyping intrusion detection in an industrial cloud-native digital twin”. In: *2021 22nd IEEE International Conference on Industrial Technology (ICIT)*. Vol. 1. IEEE. 2021, pp. 749–755.

[TB09] E. Tews and M. Beck. “Practical attacks against WEP and WPA”. In: *Proceedings of the second ACM conference on Wireless network security*. 2009, pp. 79–86.

[TN04] P. Tam and J. Newmarch. “Protocol for Ownership of Physical Objects in Ubiquitous Computing Environments”. In: *IADIS international conference E-Society*. Vol. 2004. 2004, pp. 614–621.

[Var22] S. A. Varghese. “Digital Twin-based Intrusion Detection for Industrial Control Systems”. In: *2022 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*. IEEE. 2022, pp. 611–617.

[Vie+21] M. Vielberth et al. “A digital twin-based cyber range for SOC analysts”. In: *IFIP Annual Conference on Data and Applications Security and Privacy*. Springer. 2021, pp. 293–311.

[Vig06] L. Vigano. “Automated security protocol analysis with the AVISPA tool”. In: *Electronic Notes in Theoretical Computer Science* 155 (2006), pp. 61–86.

[VP17] M. Vanhoef and F. Piessens. “Key reinstallation attacks: Forcing nonce reuse in WPA2”. In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 2017, pp. 1313–1328.

[Wal+21] H. Waldner et al. *Private Stream Aggregation from Labeled Secret Sharing Schemes*. Cryptology ePrint Archive, Paper 2021/081. `https://eprint.iacr.org/2021/081`. 2021.

[Wil92] T. J. Williams. *The Purdue enterprise reference architecture: a technical guide for CIM planning and implementation*. Instrument Society of America, 1992.

[WSJ17] M. Wollschlaeger, T. Sauter, and J. Jasperneite. “The future of industrial communication: Automation networks in the era of the internet of things and industry 4.0”. In: *IEEE industrial electronics magazine* 11.1 (2017), pp. 17–27.

[Yan+22] W. Yang et al. “Optimizing federated learning with deep reinforcement learning for digital twin empowered industrial IoT”. In: *IEEE Transactions on Industrial Informatics* 19.2 (2022), pp. 1884–1893.

[ZCZ21] K. Zhang, J. Cao, and Y. Zhang. “Adaptive digital twin and multiagent deep reinforcement learning for vehicular edge computing and networks”. In: *IEEE Transactions on Industrial Informatics* 18.2 (2021), pp. 1405–1413.

[ZFT22] T. Zhao, E. Foo, and H. Tian. “A Digital Twin Framework for Cyber Security in Cyber-Physical Systems”. In: *arXiv preprint arXiv:2204.13859* (2022).

[Zha+20] J. Zhang et al. “Formal analysis of QUIC handshake protocol using ProVerif”. In: *2020 7th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2020 6th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom)*. IEEE. 2020, pp. 132–138.

[Zha+21] D. Zhang et al. “Resilience dynamics modeling and control for a reconfigurable electronic assembly line under spatio-temporal disruptions”. In: *Journal of Manufacturing Systems* 60 (2021), pp. 852–863.

Included Publications

Evaluating the Performance of the OSCORE Security Protocol in Constrained IoT Environments

Abstract

The Constrained Application Protocol (CoAP) is a standard communication protocol for resource-constrained devices in the Internet of Things (IoT). Many IoT deployments require proxies to support asynchronous communication between edge devices and the back-end. This allows (non-trusted) proxies to access sensitive parts of CoAP messages. Object Security for Constrained RESTful Environments (OSCORE) is a recent standard protocol that provides end-to-end security for CoAP messages at the application layer. Unlike the commonly used standard Datagram Transport Layer Security (DTLS), OSCORE efficiently provides selective integrity protection and encryption on different parts of CoAP messages. Thus, OSCORE enables end-to-end security through intermediary (non-trusted) proxies, while still allowing them to perform their expected services, with considerable security and privacy improvements.

To assess whether these security features consume too much of the limited resources available on a constrained device, we have implemented OSCORE (the implementation is available as open-source), and evaluated its efficiency. This paper provides a comprehensive, comparative and experimental performance eval-

Martin Gunnarsson, Joakim Brorsson, Francesca Palombini,
Ludwig Seitz, Marco Tiloca "Evaluating the Performance of the OSCORE Security Protocol in Constrained IoT Environments". In *Internet of Things*, vol. 13, Mar. 2021, Elsevier.

uation of OSCORE on real resource-constrained IoT devices, using the operating system Contiki-NG as IoT software platform. In particular, we experimentally evaluated the efficiency of our OSCORE implementation on resource-constrained devices running Contiki-NG, in comparison with the DTLS implementation TinyDTLS maintained by the Eclipse Foundation. The evaluation results show that our OSCORE implementation displays moderately better performance than TinyDTLS, in terms of per-message network overhead, memory usage, message round-trip time and energy efficiency, thus providing the security improvements of OSCORE with no additional performance penalty.

1 Introduction

The Internet of Things (IoT) refers to a networked scenario where all connectable devices are reachable over the Internet and can communicate with each other. This has led to many new application scenarios, e.g. smart buildings, plant and home automation, smart electricity grids and smart transportation.

In such deployments, several IoT devices, also called *nodes*, are units with limited resources such as memory, computing power and energy (if they are battery-powered). Having constrained resources results in constrained network segments, e.g. due to lossy channels and limited bandwidth [BEK14b]. In order to cope with this, resource-constrained nodes tend to adopt an asynchronous and intermittent communication model, i.e. they handle network traffic according to sending/receiving timeslots. To save energy, nodes can go offline (*sleep*), between two active timeslots, considerably extending their battery lifetime.

To manage these limitations, *proxies* are used as intermediaries to enable access to server nodes that are not always online, by forwarding requests from client nodes and caching the associated responses. With this in mind, the Constrained Application Protocol (CoAP) [SHB14] has been developed with support for proxying functionality, and is now a de-facto standard application-layer protocol for IoT. CoAP is a RESTful protocol, REST being an acronym for Representational State Transfer [FT00]. The REST model considers a Client and a Server as communicating parties, where the Client sends a Request to the Server, which replies by sending a Response. Based on the intended operation to perform, CoAP Requests can be of different types, e.g. GET, PUT, POST, and DELETE.

Most applications require secure communications between client and server nodes. To this end, the CoAP specification [SHB14] considers Datagram Transport Layer Security (DTLS) [RM12] as the only method to achieve secure communication for CoAP. In particular, DTLS establishes a secure channel at the *transport layer* over unreliable datagram protocols such as UDP, and provides hop-by-hop security by protecting CoAP messages in their entirety. Due to proxies not being able to read encrypted CoAP messages, a DTLS channel must terminate at a proxy, so that the proxy can read the data needed for proxying functionality. Thus, a single DTLS channel cannot be established directly between the Client and the

Server. Instead, a first secure channel has to be established between a Client node and the proxy, and then a second secure channel has to be established between the proxy and the Server node. This in turn results in the two following issues and limitations.

First, it is necessary for the proxy to perform a double security processing of CoAP messages, as it has to decrypt a message received on the client DTLS channel, and then re-encrypt the same message for delivery on the server DTLS channel, which impacts performance. Second, the proxy is necessarily required to be trusted, as it is able to fully access the CoAP messages. Mandating such an extent of trust in proxies and the service providers operating them clearly results in unnecessary and excessive exposure of data, which in turn creates opportunities to tamper with them and easily raises privacy implications.

To efficiently overcome these issues, the protocol Object Security for Constrained RESTful Environments (OSCORE) has been recently standardized in the Internet Engineering Task Force (IETF) [Sel+19b]. OSCORE takes an *application-layer* approach for message protection based on *object security*, and selectively protects certain parts of the CoAP messages at the application layer, thus providing *end-to-end* secure communication between client and server nodes. In particular, some parts of CoAP messages can be encrypted, while other parts can be only authenticated and integrity-protected. This makes it possible to deploy non-trusted proxies, which are still able to perform their intended tasks. Furthermore, this mitigates privacy threats otherwise possible for proxies to exploit, thus preserving the personal sphere of human users related to the information exchanged and the operations performed by the communicating endpoints.

To investigate how these added security features impacts performance, a practical performance evaluation of the protocol is needed. In this paper, we therefore provide a comprehensive, experimental and comparative performance evaluation of the OSCORE protocol, considering a CoAP client and a resource-constrained CoAP server that securely communicate through a CoAP proxy. In our evaluation, we compared the performance of OSCORE against both an insecure baseline scenario using plain CoAP and an alternative secure scenario using CoAP over DTLS. To the best of our knowledge, this is the first experimental performance evaluation of a standard-compliant version of OSCORE on a real IoT device.

In order to carry out our experimental evaluation, we have implemented OSCORE (available as open-source []) for the Contiki-NG OS [1], and tested it on the resource-constrained platform Zolertia Firefly [2] equipped with the CC2538 system-on-a-chip [3]. To the best of our knowledge, ours is the first publicly available, standard-compliant implementation of OSCORE for Contiki-NG and more broadly for constrained IoT devices.

[1] `https://github.com/contiki-ng/contiki-ng/wiki` (Accessed 2020-08-13)

[2] `https://zolertia.io/product/firefly` (Accessed 2020-08-13)

[3] `http://www.ti.com/lit/ds/symlink/cc2538.pdf` (Accessed 2020-08-13)

During our experiments, we evaluated performance in terms of memory and CPU usage, as well as energy consumption on the server side, and Round Trip Time experienced on the Client side. Our results show that our OSCORE implementation displays moderately better performance than the evaluated DTLS implementation TinyDTLS, in terms of message overhead, round-trip time and energy efficiency, while still allowing a (non-trusted) proxy to perform its intended operations.

The rest of the paper is organized as follows. Section 2 presents the related work. Section 3 describes background technologies and concepts. Section 4 provides the motivation behind the OSCORE protocol and a description of how the protocol works. In Section 5, we present our experimental evaluation and discuss our results. Finally, in Section 6, we draw our conclusions.

2 Related work

With respect to channel security protocols, a number of approaches have been proposed as aimed at optimizing performance and enabling feasibility for constrained devices and networks. In particular, Raza *et al.* adapted DTLS to improve performance for resource constrained devices by using header compression mechanisms from 6LoWPAN [Raz+13]. This reduces message overhead, thus increasing energy efficiency and avoiding fragmentation. Raza *et al.* also proposed to use Next Header Compression, so that IP Security can be adapted to resource constrained devices [Raz+14]. Hummen *et al.* considered the viability of certificate-based DTLS and suggested to offload parts of the DTLS handshake to trusted gateways [Hum+13]. Sethi *et al.* proposed a similar approach, providing also end-to-end data integrity and with particular focus on performance of public-key cryptography for resource constrained devices [SAK12]. While these works target channel security protocols, the OSCORE protocol considered in this paper rather provides end-to-end security at the application layer (see Section 3) and is intended for constrained devices and networks by design (see Section 4).

In [Kot+13][Kot+12], Kothmayr *et al.* presented and evaluated an authentication security scheme for the Internet of Things, based on DTLS and intended to provide end-to-end security in the presence of an underlying network infrastructure only partially under the user's control. By doing so, the establishment and usage of a security association are entirely in the hands of the two communicating applications. This effectively works to protect application traffic end-to-end through routers, gateways and other entities operating at the network layer. However, as we explain in Section 3, this does not hold in the presence of application-layer intermediaries such as CoAP forward proxies at which DTLS has to terminate, thus also breaking end-to-end security between the two communicating applications.

All the approaches discussed above aim at reducing message overhead and ultimately improving performance of constrained devices and networks. However, none of them provide end-to-end secure communication at the application layer between client and server devices, in the presence of application-layer intermediate (untrusted) entities such as proxies. For example, one article has been presented by Van den Abeele *et al.* in [Van+17], where the authors identify the problem with DTLS and proxies. However, while they aim at offloading the work of the constrained servers, they do not achieve end-to-end security through application-layer intermediaries such as proxies. Instead, the OSCORE protocol presented in Section 4 provides end-to-end security at the application layer also in the presence of such intermediaries.

In order to achieve end-to-end security, other schemes based on object security have been proposed. One approach which is similar to OSCORE is OSCAR [Vuč+15], which also provides object security for the Internet of Things, but with a focus on access control. Besides, the object security format in OSCAR is designed for protection of *publish-subscribe* communications, rather than client-server *end-to-end* communications. That is, OSCAR considers a *many-requests-one-response* communication model, where many requests can be answered with the same response. Instead, OSCORE considers a *one-request-one-response* communication model, where request and response are strictly associated.

Work has also been done on how to protect specifically CoAP messages. In [NI15], the authors present an alternative scheme relying on object security, aimed at providing integrity-protection and authentication of CoAP messages. However, unlike OSCORE, the proposed scheme does not leverage standard efficient building blocks such as CBOR [BH13] and COSE [Sch17], and requires the addition of several new CoAP options, thus resulting in a considerable overhead for each secure message. Moreover, the usage of the HMAC-SHA256 algorithm for message integrity protection results in 32 byte Message Authentication Codes (MACs) for each protected message, which is a further significant overhead for constrained devices.

Another end-to-end security scheme for CoAP was proposed in [Uki+14]. This relies on a new CoAP option and uses AES-CCM-128 for encryption and authentication. However, unlike OSCORE, this scheme does not leverage CBOR and COSE either, with consequent overhead due to inefficient encoding. Moreover, unlike OSCORE, it protects only the message payload, without securing CoAP options and header fields.

In [R H19], the authors present an evaluation of OSCORE only, i.e. with no comparison against alternative approaches. In their work, they show how offloading AES-CCM encryption and decryption operations to hardware significantly improves performance, especially with regards to energy efficiency. However, the evaluation does not include a performance comparison against any alternative security solution, e.g. DTLS. Moreover, it is based on an implementation of an old version of OSCORE, well before its release as a standard protocol [Sel+19b]. Conversely, this paper provides a performance evaluation of a standard-compliant version of OSCORE, together with a comparison against DTLS (see Section 5.1).

In [Dur+19] the authors have implemented OSCORE within an Intel Software Guard Extensions (SGX) enclave for back-end-systems. The authors have evaluated the backend implementation against a constrained implementation with the Ephemeral Diffie-Hellman Over COSE (EDHOC) [SMP20] key-exchange protocol and keys stored in a Secure Element on the IoT device. The authors evaluation is aimed more at showing the feasibility of using hardware security mechanisms then thoroughly evaluating the OSCORE protocol. Instead, this paper focus on the quantitative evaluation of OSCORE performance, in terms of overall network performance and resource utilization on an IoT device acting as server (see Section 5.1).

Also, [Gün+20] presented an evaluation of OSCORE in the scope of Named Data Networking (NDN), hence considering a non-constrained network environment. Conversely, the performance evaluation provided by this paper consider a constrained IoT devices acting as OSCORE server, with a focus on its resource utilization in terms of CPU, memory and radio interface, as well as on its energy consumption.

The experimental evaluation of the OSCORE protocol provided in this paper uses our own standard-compliant implementation of OSCORE for the Contiki-NG operating system and intended for resource-constrained IoT devices. The evaluation includes a comparison against both an insecure baseline scenario using plain CoAP and a secure scenario using CoAP over DTLS as recommended by the CoAP standard. To the best of our knowledge, this is the first comprehensive and comparative performance evaluation of OSCORE on IoT devices, while our Contiki-NG implementation is the first one oriented to IoT devices to be publicly available and standard-compliant.

3 Technical background

This section introduces background concepts referred throughout the paper.

Channel security refers to the transmission of data over a secure channel [RK03]. This can be negotiated at the data link, network or transport layer in the protocol stack, through a specific establishment protocol. Note that a secure channel handles data agnostically, i.e. it has no knowledge of the conveyed secure data.

Object security refers to protection mechanisms for data objects, as an alternative to secure channels [RK03]. Instead of relying on a communication protocol at a lower layer to provide message protection, applications themselves handle protection and verification of their own generated messages.

The OSCORE protocol that we have implemented and evaluated in this paper is designed to secure the **Constrained Application Protocol (CoAP)** [SHB14]. CoAP is an application layer web transfer protocol, intended for resource constrained devices and networks. CoAP typically runs on top of UDP [Pos80], is not session-based and can handle loss or delayed delivery of messages. Further, it features an asynchronous messaging model and has native support for proxying.

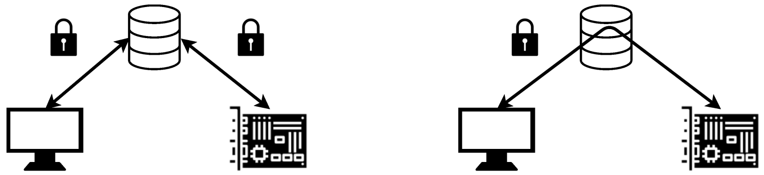
A CoAP message is divided into header and payload. The CoAP header always includes four initial bytes, followed by a *Token* value. The Token is used to bind a request with a response, and its size can vary between 0 and 8 bytes, as signaled in the first part of the CoAP header. Also, the CoAP header may further include a number of *options*, which follow a Type-Length-Value scheme and have a variable length depending on their type and specific content. These options are used to control various functions of the protocol. For example, options can be used to instruct a proxy on how to handle messages, specify for how long a message is valid, or even indicate message fragmentation at the application layer. Finally, if a payload is present, a single byte with value 0xFF acts as payload delimiter and is followed by the actual CoAP payload containing the conveyed data, whose size of course depends on the particular application message.

Concise Binary Object Representation (CBOR) [BH13] is a data encoding format designed to handle binary data, with the primary goal of achieving very small parser code size, and the secondary goal to achieve small message size. **CBOR Object Signing and Encryption (COSE)** [Sch17] specifies how to perform encryption, signing and Message Authentication Code (MAC) operations on CBOR data and to encode the result in CBOR.

DTLS [RM12] is an Internet standard providing channel security at the *transport layer* to protect communications over unreliable datagram protocols, such as UDP. That is, security is ensured hop-by-hop, i.e. between two nodes that are adjacent transport-layer hops. DTLS is a close copy of the TLS protocol [DR08b] and provides equivalent security guarantees, i.e. it prevents tampering, eavesdropping and message forgery. In particular, DTLS is adapted for use over UDP [Pos80] instead of TCP [Pos81], which is important for constrained devices and networks relying on UDP as a connectionless transport protocol. The original CoAP specification [SHB14] indicates DTLS as the only security mechanism to protect the exchange of CoAP messages.

Two communicating devices initially use the DTLS *Handshake* protocol to exchange network information and cryptographic key material for later message protection. In particular, one device acts as *client*, while the other acts as *server*. The default handshake relies on certificates, but extensions based on symmetric pre-shared keys [ET05] or on raw public keys [Wou+14] are often preferred in constrained applications. A completed handshake establishes a secure session, where the client and server can start exchanging data protected through the negotiated key material.

Secure communication is then carried out using the DTLS *Record* protocol, which provides security and reliability of message transfers. This works as an encapsulating protocol that transports data and connection state information among the two communicating parties. The record layer header conveys information including data type, sequence number, and length of the message content.



(a) Hop-by-hop, the message is decrypted to enable forwarding by the proxy.

(b) End-to-end, the message can be forwarded by the proxy without being decrypted.

Figure 1: Hop-by-hop vs. end-to-end security

4 The OSCORE protocol

A significant part of IoT devices are resource-constrained, with many even being battery powered. It is thus important to limit resource consumption, especially in terms of energy, to achieve a long device lifetime and acceptable performance. As introduced in Section 1, energy performance may rely on device *sleeping*, which in turn leads to an asynchronous communication model. In order to still provide a well functioning service, it is thus necessary to schedule requests to sleeping nodes with the help of proxies, used as intermediaries between clients and servers.

The original CoAP specification [SHB14] indicates DTLS [RM12] as the only method to achieve secure communication for CoAP. This in turn means that, when a proxy is deployed between a client and server, message protection is enforced *hop-by-hop* between client, proxies and server, as shown in Figure 1(a). Thus, in the presence of an intermediary proxy, DTLS cannot provide *end-to-end* secure communication between client and server nodes. Instead, a first secure channel has to be established between the client and the proxy, and a second secure channel has to be established between the proxy and the server. This in turn results in the security and privacy issues and limitations discussed in Section 1, i.e. the double security processing on the proxy, the unavoidable exposure of data to the proxy, as well as the need to fully trust it.

Figure 1(b) shows the alternative *end-to-end* security approach, where a client and a server rely on a two-way secure communication context. This approach essentially consists in tunneling channel security through the proxy, and hence successfully overcomes the two limitations discussed before. However, in order to be usable, a solution based on end-to-end security must allow proxies to correctly perform their intended functionalities, especially the forwarding of CoAP requests from clients and the scheduling of the associated CoAP responses from servers. Therefore it must be possible to *selectively* protect different parts of a same CoAP message in different ways, i.e. some encrypted, others only integrity protected and finally some parts fully accessible by the proxy.

This flexibility can be achieved by using *object security*, so that applications can choose which parts of an outgoing message have to be integrity-protected, encrypted, or both. Note that protecting only the CoAP payload is not sufficient against attacks such as changing the REST *Code* field in the CoAP header, e.g. from *GET* to *DELETE*, which tricks the server into deleting a resource instead of just returning its value.

The above motivates the need for lightweight end-to-end security that preserves proxying functionalities, and has led to the design of OSCORE, an application-layer protocol based on object security and fulfilling these requirements.

4.1 Functional Description of the OSCORE Protocol

This section describes OSCORE. For the reader's convenience and due to space constraints, we only present the main features, while a complete description is available in [Sel+19b]. OSCORE provides message confidentiality, integrity and reordering/replay protection, as well as a weak freshness protection through sequence numbers for CoAP messages. To this end, OSCORE transforms an *unprotected CoAP message* into a *protected CoAP message*. A protected CoAP message includes the newly defined *OSCORE option* [Sel+19b], which signals the usage of OSCORE to protect the message, as well as an encrypted COSE object [Sch17] in the CoAP payload.

OSCORE is designed for providing end-to-end security between two CoAP endpoints, while preventing intermediaries to alter or access any message field that is not related to their intended operations. The security concerns not only the actual payload of the original CoAP message, but also all the fully protected CoAP options, the original request and response REST code, as well as parts of the URI to resources targeted in request messages (see Section 4.1).

To be able to use OSCORE, the following two criteria must be fulfilled. First, the two CoAP endpoints are required to support CBOR and COSE (see Section 3), as well as the specific *HMAC-based Key Derivation Function* (HKDF) and *Authenticated Encryption with Associated Data* (AEAD) algorithms they want to use for key derivation and authenticated encryption. This assumption is fulfilled in the vast majority of IoT applications using CoAP. Second, the two CoAP endpoints are required to have an OSCORE security context (see Section 4.1), or the necessary information and keying material to derive it. While this has to happen in a secure and authenticated way, and some suitable approaches are proposed in [SMP20] and [Pal+20], OSCORE is not tied to any particular approach for context establishment, and further details are outside the scope of this paper.

The Security Context

OSCORE uses parameters and keying material included in an OSCORE *security context*, which is used to perform encryption and integrity protection operations. For this reason, every pair of communication endpoints, i.e. a CoAP client and a CoAP server, share the same security context.

The security context consists of three parts: a *Sender* part, a *Recipient* part and a *Common* part. The *Sender* part is used to protect outgoing messages (i.e. requests on the client and responses on the server). The *Recipient* part is used to verify incoming protected messages (i.e. requests on the server and responses on the client). Finally, the *Common* part contains shared data. This division is illustrated in Figure 2. An instance of a security context is present as a copy on the client and server, containing the same data values. However, as can be seen in Figure 2, the sender and recipient parts are mirrored, so that the sender part of the server corresponds to the recipient part of the client, and vice versa.

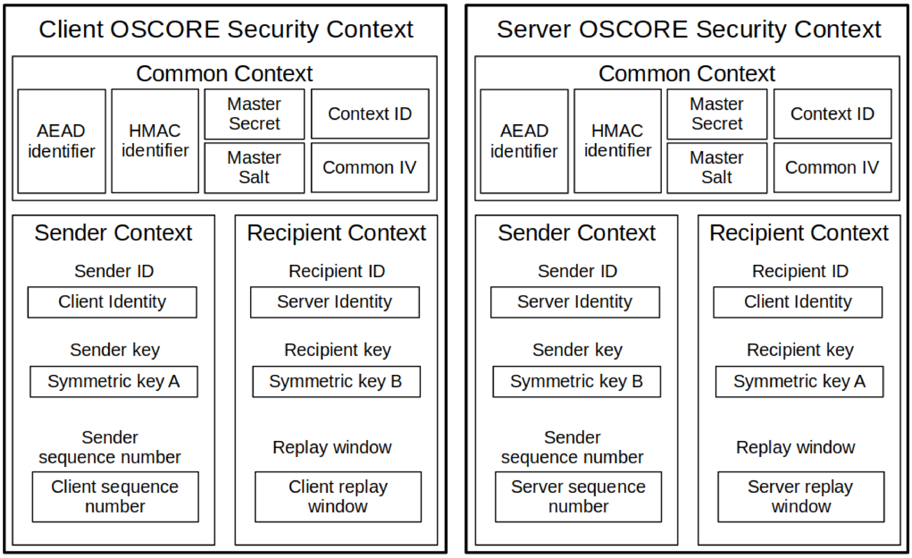


Figure 2: OSCORE Security Contexts for a Client and Server pair showing only the fields used during operation.

More in detail, the *Common* part includes: i) an identifier of the AEAD algorithm used to encrypt and authenticate exchanged messages; ii) an identifier of the HMAC-based key derivation function used to derive keys and initialization vectors (IVs); iii) the *Master Secret*, a random byte string used to derive keys and IVs; iv) the *Master Salt*, an optional byte string used with the Master Secret to derive the keys and IVs; v) a Context ID, used to identify the Common Context and to derive keys and IVs; vi) a Common IV to generate AEAD nonces.

The *Sender* part includes: i) a *Sender ID*, a byte string identifying the *Sender* part of the security context; ii) a *Sender Key*, the symmetric key to protect outgoing messages; iii) a *Sequence Number*, used for nonce generation to protect outgoing messages, and for replay protection of incoming messages.

The *Recipient* part includes: i) a *Recipient ID*, a byte string identifying the *Recipient* part of the security context; ii) a *Recipient Key*, the symmetric key to decrypt incoming messages; iii) a *Replay Window* to verify freshness of incoming messages on the CoAP server.

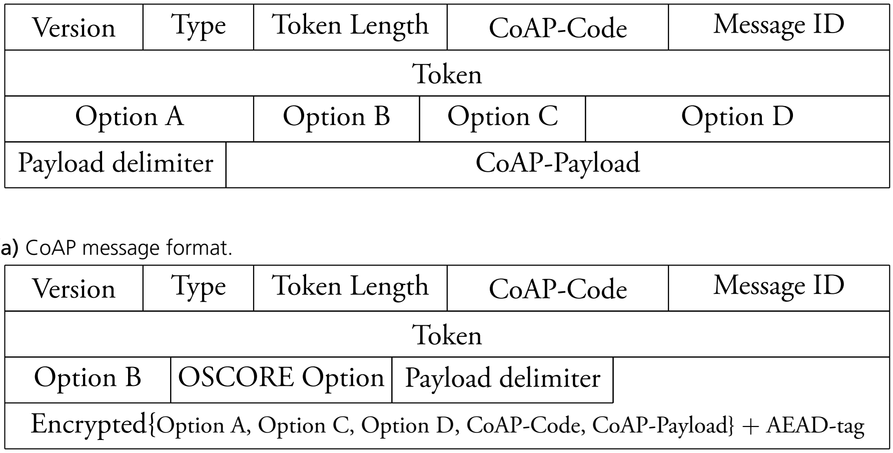
The combination of Context ID, Sender ID, Master Secret and Master Salt must be unique for each communicating pair of Client and Server. This ensures unique keys and nonces for the AEAD. Further details on establishing Sender/Recipient IDs and the ensuring their uniqueness are out scope for OSCORE and this paper.

Protecting the CoAP Message

Different parts in a CoAP message are protected in different ways. That is, *Confidential data*, which should neither be read or altered by a proxy, are both encrypted and integrity protected. *Static data*, which should be readable but not changed, are integrity protected but not encrypted. *Dynamic data*, which a proxy should be able to modify, are not protected. Finally, there are also *Mutually known data*, which the sender and receiver have agreed upon before exchanging messages. These data are part of the input to the integrity protection process, to ensure that the two communicating endpoints behave correctly and possibly detect anomalies. However, they are never sent as both parties already know them.

Figure 3 shows a comparison between an unprotected CoAP message and the resulting OSCORE-protected CoAP message. As discussed in Section 3, the first four bytes are followed by the variable-length Token, possible variable-length options, and a variable length payload. With particular reference to the OSCORE message format, we can see that sensitive parts of a message are encrypted, e.g. some options and the payload, while others are left unencrypted, e.g. some options and some fields of the CoAP header. The encrypted content is placed into the payload of the protected message.

The actual protection process takes as input an unprotected CoAP message and produces a protected OSCORE message as follows.
1) The confidential data are enclosed into a *COSE object* [Sch17]. These include the REST code of the original CoAP message, a subset of the CoAP options, and the CoAP payload (if present). The CoAP options considered at this step are the ones not relevant for operations of intermediary (proxy) units.
2) The static fields of the CoAP header and static proxy-readable CoAP options needs to be authenticated and integrity protected, but not to be encrypted. This set of data composes the *Additional Authenticated Data* (AAD).
3) The COSE object is finalized, by encrypting and integrity protecting the data it encloses, while only integrity protecting the AAD. To this end, the Sender Key and the Sender Sequence Number from the Sender Context are used. The resulting *ciphertext* and *AEAD-tag* is included in the *Message Content* field of the COSE Object.
4) The COSE object is used as payload of the protected CoAP message, and any encrypted options are removed from the CoAP message. The original REST code



Version | Type | Token Length | CoAP-Code | Message ID

Token

Option A | Option B | Option C | Option D

Payload delimiter | CoAP-Payload

(a) CoAP message format.

Version | Type | Token Length | CoAP-Code | Message ID

Token

Option B | OSCORE Option | Payload delimiter

Encrypted{Option A, Option C, Option D, CoAP-Code, CoAP-Payload} + AEAD-tag

(b) OSCORE message format.

Figure 3: Message layout, with named fields, for unprotected (a) and protected (b) CoAP messages.

is replaced with POST (2.04) for a CoAP request (response), or with FETCH (2.05) for a CoAP request (response) using the CoAP mechanism *Observe* [Har15].

An analogous reverse process is performed upon receiving a protected message, together with anti-replay checks. To decrypt the protected message, the recipient CoAP endpoint uses the Recipient Key from its own Recipient Context associated to the message originator.

Proxy Functionalities and Data Protection

Building on the previous sections, we can now describe how OSCORE handles proxying of encrypted messages. OSCORE is designed to uniquely bind each request to the corresponding response, thus preventing proxies from serving cached responses to clients different from the one originating the request.

As previously stated, OSCORE cannot encrypt entire CoAP messages. An example of static data in a CoAP message which can not be encrypted but should be integrity protected is the *Version* field of the CoAP header. This field has to remain readable, so that the receiver endpoint knows how to process an incoming message, but should be integrity protected to prevent future version-based attacks.

The *Token* field of the CoAP header also has to remain readable, as it is used for binding each request to the corresponding response. However, unlike the *Version* field, the *Token* field cannot be integrity protected, as it can be modified by proxies, when a message traverses the network.

5 Evaluating OSCORE in comparison with CoAP and CoAP over DTLS

In this section, we present our evaluation of OSCORE. In particular, we first describe our experimental setup in Section 5.1. Then, we analyze the overhead introduced by OSCORE in Sections 5.2 and 5.3. Finally, in Section 5.4, we present and discuss our experimental results.

5.1 Experimental Evaluation Method

To evaluate the feasibility and convenience of OSCORE, we developed a prototype implementation for resource-constrained CoAP servers. This section presents the conducted experiments which evaluates the performance of OSCORE. We chose to evaluate OSCORE against both plain CoAP and CoAP secured with DTLS, since CoAP recommends DTLS as a security mechanism.

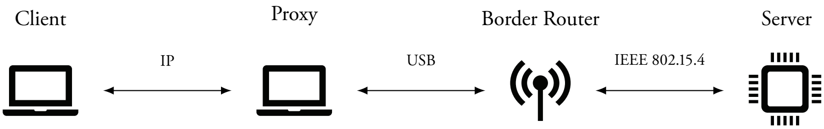


Figure 4: Experimental test scenario.

For our experiments, we considered the test scenario in Figure 4, which consists of a CoAP client, a CoAP proxy and a CoAP server. The client (C) and the proxy (P) were implemented using an extended version of the Java library Californium/Scandium [4], which provides both CoAP and DTLS. The client and the proxy ran as two distinct processes on a same commodity PC. To enable communication between P and the server (S), we relied on a dedicated border router (BR) device. In particular, both BR and S were resource-constrained Zolertia Firefly boards [5], and ran the Contiki-NG OS [6] together with an extended version of the Erbium library providing the communication stack. The Firefly boards are based on the CC2538 chipset and equipped with 512 KB of ROM, 32 KB of RAM, a 32 MHz ARM Cortex-M3 CPU, and an IEEE 802.15.4 [IEE11] radio interface. Based on these features, they can be categorized as Class 2 constrained nodes according to [BEK14b].

We considered and compared three different test cases: ”COAP”, ”COAPS” and ”OSCORE”. In all three test cases, P acts as CoAP proxy and relays CoAP requests from C to S, as well as corresponding CoAP responses from S to C. More specifically, the three test cases were defined as follows.

[4] `http://www.eclipse.org/californium` (Accessed 2020-08-13)

[5] `https://zolertia.io/product/firefly` (Accessed 2020-08-13)

[6] `https://github.com/contiki-ng/contiki-ng/wiki` (Accessed 2020-08-13)

• **”COAP”**. The ”COAP” test case, acting as a baseline for comparison, considered plain CoAP communication with no security provided. In order to carry out the performance evaluation for this test case, we used the constrained implementation of CoAP in the Contiki-NG adaptation[7] of Erbium [KDD14].

• **”COAPS”**. The ”COAPS” test case considered CoAP communication, with the addition of DTLS 1.2, providing hop-by-hop secure communication. DTLS was configured with a first secure channel between C and P, and a second secure channel between P and S. The DTLS cipher suite used is TLS_PSK_WITH_AES_128_CCM_8 [MB12]. In order to carry out the performance evaluation for this test case, we used the constrained implementation of DTLS from TinyDTLS[8] for Contiki-NG[9].

• **”OSCORE”**. The ”OSCORE” test case considered CoAP communication with the addition of OSCORE, providing end-to-end secure communication between C and S, as described in Section 4.1. In order to carry out the performance evaluation for this test case, we have made our own constrained implementation[] of OSCORE, based on the Contiki-NG version of Erbium. To the best of our knowledge, this is the first implementation of OSCORE for Contiki-NG, and more broadly for constrained IoT devices, to be publicly available and fully aligned with the standard specification [Sel+19b].

Note that neither ”COAPS” nor ”OSCORE” relied on hardware acceleration for cryptographic operations, and that ”COAP” does not make use of cryptography.

We have made a theoretical analysis of how the total message size relates to the total transmitted size and the impact this has on the energy consumption of the constrained nodes.

In all three test cases, the client sent POST requests addressed to a dedicated target resource at the server S. Also, S was configured to reply to each such request by sending a response with the same payload size. The client was pre-configured in order to skip the resource discovery process.

In the considered setup, the server did not use Radio Duty Cycling, a mechanism provided by Contiki-NG for periodically switching off the node’s radio interface, in order to conserve energy[10]. That is, with Radio Duty Cycling turned off, the radio interface is always active. We chose to adopt this configuration since

[7] `https://github.com/contiki-ng/contiki-ng/wiki/Documentation:-CoAP` (Accessed 2020-08-13)

[8] `https://projects.eclipse.org/projects/iot.tinydtls` (Accessed 2020-08-13)

[9] `https://github.com/contiki-ng/contiki-ng/wiki/Documentation:-CoAP` (Accessed 2020-08-13)

[10] `https://github.com/contiki-os/contiki/wiki/Radio-duty-cycling` (Accessed 2020-08-13)

using Radio Duty Cycling would unnecessarily perturb our measurements, by essentially adding noise to the collected data and being counterproductive for the intended performance evaluation. In particular, Round Trip Times would be affected if the server is ”sleeping” when a message arrives, for reasons not strictly related to the considered (secure) communication protocols under evaluation.

At the time of writing, OSCORE does not have a standardized equivalent to the DTLS handshake for establishing a security context between the two peers. Therefore, to ensure a consistent and meaningful performance comparison, we consider the DTLS handshake out of scope, and instead focus on evaluating the (secure) exchange of messages. In particular, the DTLS handshake, as well as the establishment of OSCORE security contexts, are assumed to be completed before any messages are sent. Therefore, the ”COAPS” and ”OSCORE” test cases omit DTLS handshakes and OSCORE key provisioning.

For each test case, we performed separate experiments. During a given experiment, the payload size of every CoAP message was either 1, 16, 32, 48, 64, 80, 96, 112 or 128 bytes. For each listed payload size, we performed 500 message exchanges between the client and the server, through the proxy. The messages were sent at a constant rate of 10 messages per second. In all the test cases, we performed the following measurements: i) Responsiveness as experienced by C for a message exchange with S; ii) CPU usage by the server; iii) Memory usage by the server, i.e. RAM and ROM; iv) Radio usage by the server; and v) Energy usage by the server. In particular, we performed the measurements as follows.

Responsiveness. We evaluated the Round Trip Time (RTT), as experienced by the client when performing a full Request-Response exchange. The statistical significance of these results were verified with the method of paired t-tests.

CPU usage. We measured the execution time needed to process both incoming messages from a client and outgoing response messages. For the ”COAPS” and ”OSCORE” test cases, this includes decryption and integrity verification of incoming messages, as well as encryption of outgoing messages. Paired t-tests were used for verification of statistical significance.

Memory usage. The static memory utilization was determined by using the GNU utility *size*. Since Contiki-NG does not use dynamic memory allocation on the heap, dynamic memory utilization is limited to the stack. Hence, in our experiments, we used the *stack painting* painting technique, where a known value is written to all addresses of the stack part of the memory. After the experiments, the number of bytes that had been overwritten by the program execution were counted.

Radio usage. We have measured the time needed by the server to receive and transmit messages using *Energest*, a Contiki-NG utility for monitoring system utilization. Energest makes it possible to measure the time intervals where the CPU has been active, or where the radio interface has been active either in reception mode (RX) or transmission mode (TX). Energest has been proven to enable an accurate estimation of energy consumption, while increasing the computing time by only 0.7% [A D07].

Energy usage. We measured the energy consumed at the server, both by the CPU, and by the radio interface in transmission and reception mode. Each measurement was computed as the product between the overall related time collected by Energest, and the power consumption of the related hardware component as documented in the respective manuals [11] [12].

5.2 Payload Size

To aid reasoning and facilitate further discussion in the next sections, we have analyzed overhead introduced by DTLS and OSCORE, compared to plain CoAP. To ensure a fair comparison, we have considered the same AEAD cipher for both DTLS and OSCORE, namely AES-128-CCM-8. Tables 1a and 1b show the overhead of the two different protocols. The entry ”AEAD Tag” refers to the resulting Integrity Check Value produced by the AEAD cipher.

Table 1: Payload overhead for DTLS 1.2 and OSCORE.

Type	1
Version	2
Epoch	2
Sequence Number	6
Length	2
AEAD Tag	8
Total overhead	21

(a) Overhead of a DTLS-record layer message (bytes).

	Request	Response
OSCORE Option Byte	1	1
OSCORE Flag Byte	1	-
Partial IV	0-5	-
Kid	0-7	-
CoAP Code	1	1
Payload Marker	1	1
AEAD Tag	8	8
Total overhead	12-24	11

(b) Overhead of an OSCORE message (bytes).

As we can see in Table 1a, DTLS displays a fixed payload overhead of 21 bytes, which is equal for both requests and responses. This results in a total overhead of 42 bytes for a full message exchange. Note that, as defined in the DTLS profile for IoT in [TF16] (Appendix B), devices using DTLS are actually expected to further include an additional 8-byte explicit nonce to the DTLS header. This would result in an overhead of 29 bytes per message, i.e. of 58 bytes for each two-way message exchange.

In OSCORE, the overhead can vary, due to the following reasons. First, the ”Partial IV” field in the OSCORE option includes the message sequence number, whose value is incremented and size grows over time as Requests are transmitted, up to a maximum size of 5 bytes. Second, the ”Kid” field (Key Id) in the OSCORE

[11] `https://zolertia.io/product/firefly` (Accessed 2020-08-13)

[12] `http://www.ti.com/lit/ds/symlink/cc2538.pdf` (Accessed 2020-08-13)

option is immutably set by the user during early configuration, with possible sizes ranging between 0 (empty Key Id) and 7 bytes.

With reference to Table 1b, we can see that, as long as the Key Id is chosen to have a length of maximum 4 bytes, OSCORE will display the same or lower overhead for all Requests. Note that a Key Id of maximum 2 bytes is expected to be the practical choice for most applications using OSCORE. Furthermore, OSCORE Responses omit a number of implicit fields in the OSCORE option, thus showing a smaller fixed overhead of 11 bytes. Note that, unlike DTLS, OSCORE has a (much) smaller overhead for responses than requests. Assuming a high-value Partial IV of 5 bytes and a Key Id of 2 bytes, this would result in an overhead of 19 bytes per request message and of 11 bytes per response message, i.e. of 30 bytes for a two-way message exchange.

5.3 Network Energy Overhead

An application relying on IEEE 802.15.4 typically displays an effective data rate (i.e. excluding headers, CRCs and control packets) of about 8.4 kbit/s (out of 250 kbit/s). However, as shown by Latré *et al.*, IEEE 802.15.4 networks can actually achieve a throughput of about 140 kbit/s, even if acknowledgement frames are transmitted [B L05]. Using these numbers together with the energy consumption numbers stated in the CC2538 datasheet [13], we get the numbers shown in Table 2.

Table 2: Overhead in transmission time and energy consumption for a CC2538 server receiving and sending DTLS and OSCORE messages.

	Time (ms)		Energy (μJ)	
	DTLS	OSCORE	DTLS	OSCORE
Request	1.2	1.086	95.04	71.676
Response	1.2	0.628	79.2	49.738
Exchange	2.4	1.714	174.24	121.414

5.4 Results and Discussion

This section presents and discusses the results of our experiments, with reference to the test cases and scenario described in Section 5.1. The data presented here represent the average for a single message from a sample set of 500 messages.

Responsiveness

The top graph in Figure 5 shows the average round-trip time (RTT) experienced by the client for different payload sizes, with the different curves showing the three

[13] `http://www.ti.com/lit/ds/symlink/cc2538.pdf` (Accessed 2020-08-13)

different test cases. The bottom graph shows the calculated difference in mean response time between OSCORE and COAPS, with error-bars showing the standard deviation.

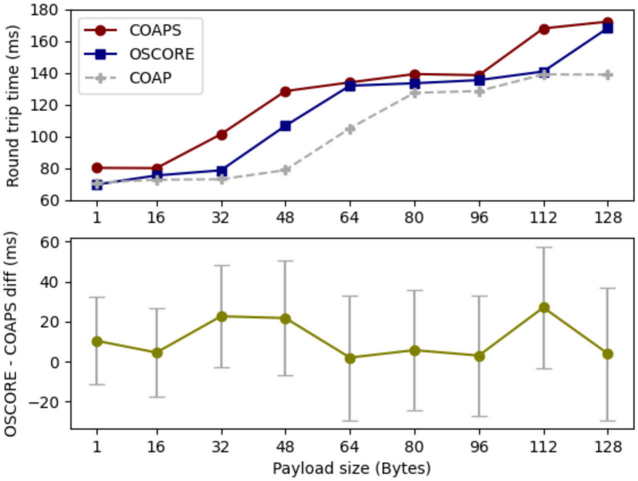


Figure 5: Measurement of responsiveness comparing RTT between COAP, COAPS and OSCORE.

Table 3 shows the statistical significance (t-statistics and p-values) for the values in the bottom graph in Figure 5. The statistics in Table 3 have been derived using paired t-tests, by comparing the response time sample populations for each payload size.

Table 3: Statistical significance for RTT, (t-statistics and p-values)

Payload (Bytes)	1	16	32	48	64	80	96	112	128
t	7.8	3.2	14.5	12.2	1.0	3.0	1.7	13.8	1.9
p	0.0	0.0	0.0	0.0	0.30	0.0	0.09	0.0	0.06

As shown in Figure 5, we can see that there is a notable difference in the mean response time between the OSCORE and DTLS protocols. In particular, we observe that OSCORE is more efficient than COAPS for all the considered payload sizes, as it results in an always smaller RTT experienced on the client side. Consistently, we also observe that, with respect to COAPS, the RTT experienced with OSCORE is always closer (and often very close) to the one measured for the baseline COAP case.

As per Table 3, the statistical significance of the difference between the OSCORE and COAP RTT is strong for most packet sizes, achieving a 99% confidence interval. However, for 64, 96 and 128 bytes payload, statistical significance is not achieved. This is mostly due to a large variance in the transfer time on the IEEE 802.15.4 network. Overall, the lines in the top graph of Figure 5 resemble a staircase. Further investigations showed that this is especially due to package fragmentation, as especially involved for the payload sizes mentioned above, hence especially experiencing variance in the transfer time.

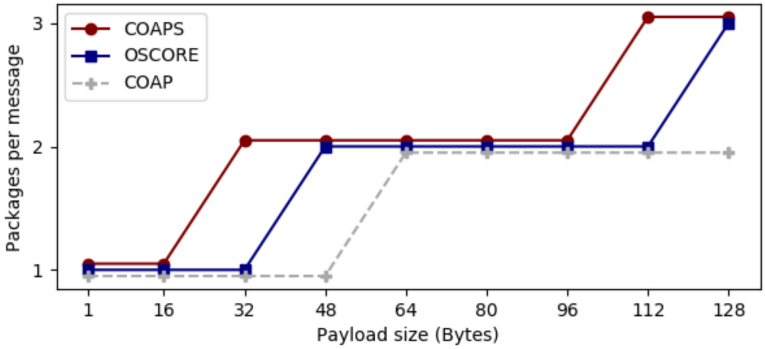
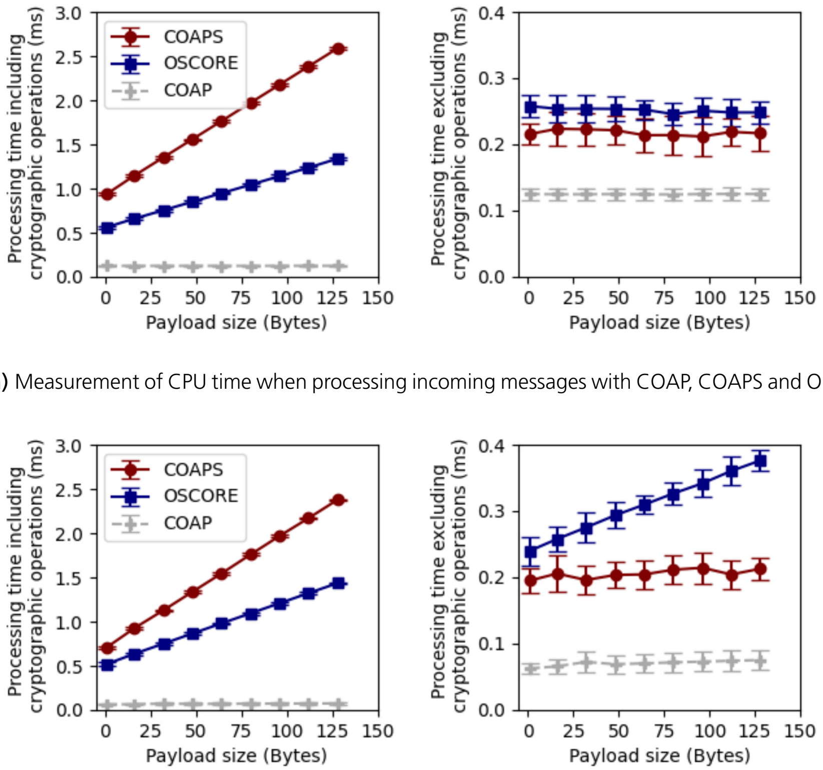


Figure 6: Number of packages needed to transport a single message.

Figure 6 shows the number of packages needed to transport one CoAP message, considering the different protocols and payload sizes. In particular, we can see that the biggest possible payload that can be carried in a single frame has a different size in the three scenarios, with COAP being able to fit the most data into a single frame, and COAPS the least. That is, one packet is sufficient for COAP, OSCORE and COAPS to transport a 16-byte payload, but only for COAP and OSCORE to transport a 32-byte payload. Similarly, two packets are sufficient for COAP, OSCORE and COAPS to transport a 96-byte payload, but only for COAP and OSCORE to transport a 112-byte payload. This is because the total header sizes due to the different used protocols, and with them the resulting overhead, vary between the COAP, OSCORE and COAPS scenarios. That is, it amounts to 76, 90 and 105 bytes, respectively, while the Maximum Transmission Unit for a IEEE 802.15.4 network is 127 bytes. Therefore, for a number of payload sizes, OSCORE enables the transmission of more data using less packets, with respect to COAPS. For COAPS and OSCORE messages with payloads of sizes: 48, 64, 80, 96, and 128 bytes, the same number of packets are needed to send the message. That only messages with 48 and 80-byte payloads achieve statistical significance, indicating that the RTT difference is minimal. Exactly why these payload sizes achieve statistical significance is to the authors, at this point, unknown. Further experiments with more payload sizes might give additional information and insights about this phenomenon.

CPU Usage

Figure 7a and Figure 7b show the CPU time for processing incoming and outgoing COAP, COAPS and OSCORE messages. The left graphs show the total CPU time for the different protocols, including cryptographic operations. The right graphs show the CPU time excluding cryptographic operations.



(a) Measurement of CPU time when processing incoming messages with COAP, COAPS and OSCORE.

(b) Measurement of CPU time when processing outgoing messages with COAP, COAPS and OSCORE.

Figure 7: Measurements of CPU time when processing messages with COAP, COAPS and OSCORE. The error bars show standard deviations.

In particular, Figure 7a shows the CPU-time for processing incoming messages. That is, the left graph shows the total CPU-time for processing incoming messages with COAPS, OSCORE and COAP. The graph shows that OSCORE is faster than COAPS when processing incoming messages, for all message sizes. In the right graph of Figure 7a, we show the CPU-time for processing incoming messages excluding the CPU-time spent for decryption. Here we see that OSCORE takes longer time than COAPS for all message sizes.

Similarly, Figure 7b shows the CPU-time for processing outgoing messages. The left graph shows the total CPU-time for processing outgoing messages with COAPS, OSCORE and COAP. Where cryptographic operations are taken into account, OSCORE is faster than COAPS when processing outgoing messages, for all messages sizes. In the right graph of Figure 7b, we show the CPU-time for processing outgoing messages excluding the CPU-time spent for encryption. Again, here we see that OSCORE takes longer time than COAPS for all message sizes.

The observed difference between the right and left side of the graphs is affected by the following points. Other than the actual encryption/decryption processing of messages, OSCORE is slower than DTLS due to: i) a more complex handling of OSCORE security contexts (i.e. retrieval and update), compared to the handling of DTLS sessions; ii) a more complex preparation of a protected OSCORE message from an original CoAP message and vice versa (see Section 4.1), compared to the preparation of a protected DTLS record from an original CoAP message and vice versa. However, when cryptographic operations are taken into account, OSCORE outperforms DTLS by being more efficient in protecting/unprotecting handled messages. Ultimately, OSCORE achieves this result by leveraging a more efficient implementation of the AES-CCM algorithm.

The results in both Figure 7a and Figure 7b are verified for statistical significance with 99% confidence interval using a paired t-test. The results of these experiments show that the CPU performance for both protocols hinges on a fast cipher implementation. In these experiments we used software implementations of AES128-CCM. Hardware acceleration of the encryption will increase the performance of both OSCORE and COAPS.

Memory Usage

Figure 8 shows the memory usage results. In particular, the left bar chart shows the RAM usage, including the maximum stack usage, while the right bar chart shows the ROM usage. In order to be independent from the particular used cryptographic primitives, e.g. cipher and hash functions, the shown memory results do not include memory usage due to such primitives. Furthermore, since the DTLS Handshake protocol is not comparable against anything analogous in OSCORE, the memory usage due to the DTLS Handshake is also excluded from the shown results. Nevertheless, for the sake of information completeness, the right bar chart highlights the memory utilization due to such contributions with faded color areas at the top of the bars.

We can see that OSCORE uses less RAM and ROM than DTLS. Furthermore, OSCORE only uses 2% more RAM than COAP, while COAPS uses 17% more RAM than COAP. When comparing the ROM usage excluding cryptographic primitives, OSCORE uses 12% more ROM than COAP, while COAPS uses 27% more ROM than COAP when excluding the DTLS Handshake proto-

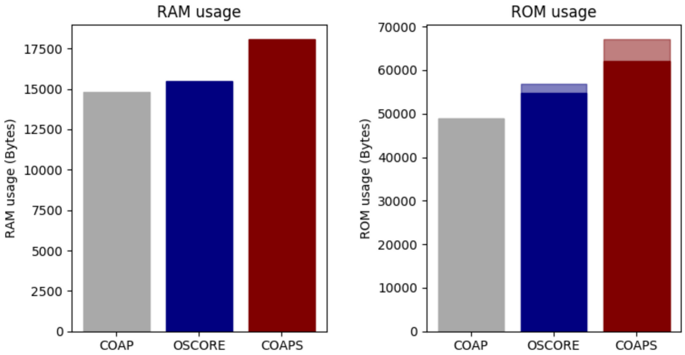


Figure 8: Memory utilization. The lighter parts in the ROM usage graph on the right indicates the memory used for the DTLS Handshake (in the "COAPS" test case) and for cryptographic primitives (in the "COAPS" and "OSCORE" test cases).

col and cryptographic primitives. In total COAP uses 45% of the available RAM and 9 % of the available ROM on the Zolertia Firefly board. OSCORE uses 47 % of the available RAM and 10 % of the ROM, the same numbers for DTLS are 55 % of the RAM and 12 % of the ROM on the boards. So while OSCORE uses more memory compared to COAP, and DTLS uses more than OSCORE, the total memory usage for all the protocols is manageable on the Zolertia Firefly platform.

Radio Usage

Figure 9 shows the radio utilization rates for the tested protocols. The left graph shows the percentage of time spent in transmitter mode while the right graph shows the percentage of time spent in receiver mode.

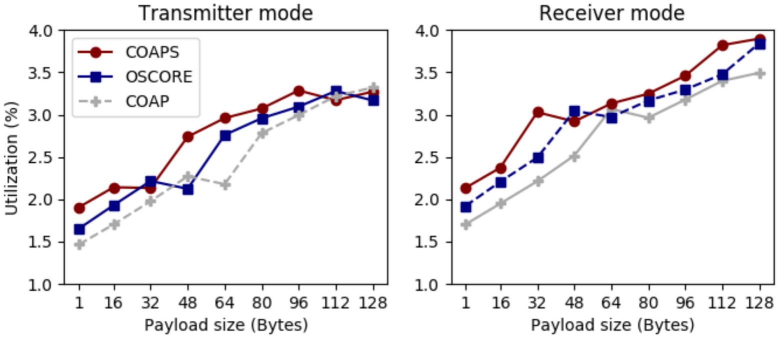


Figure 9: Measurements of radio time occupancy for transmitting and receiving for the three protocols.

Compared to COAPS, OSCORE displays less radio usage when transmitting messages, for most payload sizes. The time spent in receiver mode is also less for OSCORE, for most payload sizes. For messages whose payload size results in packet fragmentation at the link layer, there are observable deviations from the main trend in the graphs. Looking at the graph showing the transmitter mode utilization, this can be observed as a decrease of utilization, compared to other data points. This applies to COAPS with 32 and 112 byte payloads, OSCORE with 48 and 128 byte payloads and COAP with 64 byte payloads. Based on our experiments, we believe that this is related again to getting close to a fragmentation threshold enforced at the link layer. Following each of such decline points, we can observe a return to the main trend. To better understand this phenomenon, experiments with more payload sizes would probably give further insights. We conclude that OSCORE generally requires fewer network resources than COAPS, but more experiments will show this in better detail.

These results were acquired using Energest, which uses timers to measure the time the radio has spent in either transmitter mode or receiver mode. Note that switching one timer off and the other on, e.g. going from transmitter mode to receiver mode cannot be done instantly. Therefore, the sum of the time in transmitter mode and the time in receiver mode does not always add up to the total elapsed time. However, the difference between these times and the error percentage is negligible.

Energy Usage

We used Energest to measure the energy used for CPU, radio transmission and radio reception by the different protocols for a message transaction. These results, along with a summation of total energy usage, can be seen in Figure 10.

We can see that the impact of the CPU power consumption is larger than the impact of the radio power consumption. A contributor to this is the fact that these CPU measurements include power consumed when the CPU idles in between messages (which are sent at a rate of 10 messages per second). This factor can be reduced by letting the CPU sleep instead of idling in between messages.

Most important, we can see that OSCORE uses less energy in total with respect to COAPS, for all payload sizes. In particular, OSCORE results in a per-exchange energy consumption about 8-28% higher than in COAP. This shows that OSCORE is more energy efficient than COAPS, which results in an energy consumption about 17-59% higher than in COAP.

6 Conclusion

In this paper, we have considered OSCORE, a recently standardized security protocol for the IoT that efficiently protects CoAP messages and provides end-to-end security at the application layer, also in the presence of application-layer inter-

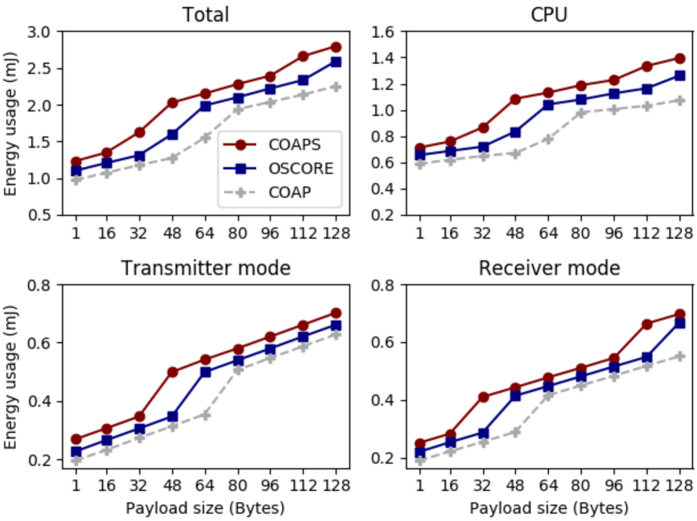


Figure 10: Energy consumption per message exchange. Note that scales are different for the y-axes.

mediaries such as proxies. In particular, we have provided a comprehensive, experimental and comparative performance evaluation of OSCORE, considering a CoAP client and a resource-constrained CoAP server that securely communicate through a CoAP proxy.

In our evaluation, we compared the performance of OSCORE against both an insecure baseline scenario using plain CoAP and an alternative secure scenario using CoAP over DTLS. Our experimental results show that OSCORE displays moderately better performance than DTLS in important metrics, namely radio transmission overhead, round trip time as experienced by CoAP clients, and memory usage as well as energy efficiency for constrained servers, while still allowing a (non-trusted) proxy to perform its intended operations. To the best of our knowledge, this paper provides the first experimental performance evaluation of a standard-compliant version of OSCORE on a real IoT device.

Our evaluation relied on our own standard-compliant implementation of OSCORE for Contiki-NG, and especially considered the resource-constrained platform Zolertia Firefly equipped with the CC2538 system-on-a-chip. To the best of our knowledge, ours is the first publicly available, standard-compliant implementation of OSCORE for Contiki-NG and more broadly for constrained IoT devices.

As future work, we will focus on using OSCORE to secure CoAP messages sent over IP multicast in use cases relying on group communication, for which we also plan to carry out an experimental performance evaluation in a group of real IoT devices. Experiments with a more extensive test envelope will also be interesting, using more and payload sizes in smaller size increments than the ones tested here.

Acknowledgements

The authors sincerely thank the anonymous reviewers and the Editor-in-Chief for their insightful comments and suggestions, that have helped to improve the technical and editorial quality of the manuscript. This work received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no. 607109. This work was also supported by the EIT-Digital High Impact Initiative ACTIVE; VINNOVA for the Celtic-Plus project CyberWI and the Celtic-Next project CRITISEC; the H2020 project SIFIS-Home (Grant agreement 952652); the SSF project SEC4Factory under the grant RIT17-0032; and the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. We thank Rikard Höglund and Jiye Park for their help.

References

[] *OSCORE implementation for Contiki-NG.* `https://github.com/Gunzter/contiki-ng/tree/master` (Accessed 2020-11-27).

[A D07] A. Dunkels, F. Österlind, N. Tsiftes and Z. He. "Software-based On-line Energy Estimation for Sensor Nodes". In: *Proceedings of the 4th Workshop on Embedded Networked Sensors.* EmNets '07. Cork, Ireland: ACM, 2007, 28–32.

[B L05] B. Latré, P. De Mil, I. Moerman, N. Van Dierdonck, B. Dhoedt, and P. Demeester. "Maximum throughput and minimum delay in IEEE 802.15.4". In: *The 1st International Conference on Mobile Ad-Hoc and Sensor Networks.* Springer, 2005, 866–876.

[BEK14b] C. Bormann, M. Ersue, and A. Keränen. *Terminology for Constrained-Node Networks.* RFC 7228. May 2014.

[BH13] C. Bormann and P. Hoffman. *Concise Binary Object Representation (CBOR).* RFC 7049. Fremont, CA, USA: RFC Editor, Oct. 2013, pp. 1–54.

[DR08b] T. Dierks and E. Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.2.* RFC 5246. Fremont, CA, USA: RFC Editor, Aug. 2008, pp. 1–104.

[Dur+19] A. Durand et al. "Trusted Lightweight Communication for IoT Systems Using Hardware Security". In: *Proceedings of the 9th International Conference on the Internet of Things.* 2019, pp. 1–4.

[ET05] P. Eronen and H. Tschofenig. *Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)*. RFC 4279. Fremont, CA, USA: RFC Editor, Dec. 2005, pp. 1–15.

[FT00] R. T. Fielding and R. N. Taylor. *Architectural styles and the design of network-based software architectures*. Vol. 7. University of California, Irvine Doctoral dissertation, 2000.

[Gün+20] C. Gündogan et al. *IoT Content Object Security with OSCORE and NDN: A First Experimental Comparison*. Technical Report. Open Archive: arXiv.org, 2020.

[Har15] K. Hartke. *Observing Resources in the Constrained Application Protocol (CoAP)*. RFC 7641. Fremont, CA, USA: RFC Editor, Sept. 2015, pp. 1–30.

[Hum+13] R. Hummen et al. "Towards Viable Certificate-based Authentication for the Internet of Things". In: *Proceedings of the 2Nd ACM Workshop on Hot Topics on Wireless Network Security and Privacy*. HotWiSec '13. Budapest, Hungary: ACM, 2013, pp. 37–42.

[IEE11] IEEE Computer Society. *IEEE Standard for Local and Metropolitan Area Networks, Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)*. Oct. 2011.

[KDD14] M. Kovatsch, S. Duquennoy, and A. Dunkels. *Erbium (Er) REST Engine and CoAP Implementation for Contiki*. 2014.

[Kot+12] T. Kothmayr et al. "A DTLS based end-to-end security architecture for the Internet of Things with two-way authentication". In: *Local Computer Networks Workshops (LCN Workshops), 2012 IEEE 37th Conference on*. IEEE. 2012, pp. 956–963.

[Kot+13] T. Kothmayr et al. "DTLS based security and two-way authentication for the Internet of Things". In: *Ad Hoc Networks* 11.8 (2013), pp. 2710–2723.

[MB12] D. McGrew and D. Bailey. *AES-CCM Cipher Suites for Transport Layer Security (TLS)*. RFC 6655. Fremont, CA, USA: RFC Editor, July 2012, pp. 1–8.

[NI15] H. V. Nguyen and L. L. Iacono. "REST-ful CoAP Message Authentication". In: *2015 International Workshop on Secure Internet of Things (SIoT)*. Oct. 2015, pp. 35–43.

[Pal+20] F. Palombini et al. *OSCORE profile of the Authentication and Authorization for Constrained Environments Framework*. Internet-Draft draft-ietf-ace-oscore-profile-13. Work in progress. IETF Secretariat, Sept. 2020.

[Pos80] J. Postel. *User Datagram Protocol.* RFC 768. Fremont, CA, USA: RFC Editor, Aug. 1980, pp. 1–3.

[Pos81] J. Postel. *Transmission Control Protocol.* RFC 793. Fremont, CA, USA: RFC Editor, Sept. 1981, pp. 1–91.

[R H19] R. H. Randhawa, A. Hameed A. N. Mian. “Energy efficient cross-layer approach for object security of CoAP for IoT devices”. In: *Ad Hoc Networks* 92 (2019), p. 101761.

[Raz+13] S. Raza et al. “Lithe: Lightweight Secure CoAP for the Internet of Things”. In: *IEEE Sensors Journal* 13.10 (Oct. 2013), pp. 3711–3720.

[Raz+14] S. Raza et al. “Secure communication for the Internet of Things - a comparison of link-layer security and IPsec for 6LoWPAN”. In: *Security and Communication Networks* 7.12 (2014), pp. 2654–2668.

[RK03] E. Rescorla and B. Korver. *Guidelines for Writing RFC Text on Security Considerations*. RFC 3552. Fremont, CA, USA: RFC Editor, July 2003, pp. 1–44.

[RM12] E. Rescorla and N. Modadugu. *Datagram Transport Layer Security Version 1.2*. RFC 6347. Jan. 2012.

[SAK12] M. Sethi, J. Arkko, and A. Keränen. “End-to-end security for sleepy smart object networks”. In: *Local Computer Networks Workshops (LCN Workshops), 2012 IEEE 37th Conference on*. Oct. 2012, pp. 964–972.

[Sch17] J. Schaad. *CBOR Object Signing and Encryption (COSE)*. RFC 8152. RFC Editor, July 2017.

[Sel+19b] G. Selander et al. *Object Security for Constrained RESTful Environments (OSCORE)*. RFC8613 (Proposed Standard), Internet Engineering Task Force. RFC Editor, July 2019.

[SHB14] Z. Shelby, K. Hartke, and C. Bormann. *The Constrained Application Protocol (CoAP)*. RFC 7252. June 2014.

[SMP20] G. Selander, J. Mattsson, and F. Palombini. *Ephemeral Diffie-Hellman Over COSE (EDHOC)*. Internet-Draft draft-ietf-lake-edhoc-02. Work in progress. IETF Secretariat, Oct. 2020.

[TF16] H. Tschofenig and T. Fossati. “Transport Layer Security (TLS)/Datagram Transport Layer Security (DTLS) Profiles for the Internet of Things”. In: *RFC 7925*. Internet Engineering Task Force, 2016.

[Uki+14] A. Ukil et al. “Lightweight security scheme for IoT applications using CoAP”. In: *International Journal of Pervasive Computing and Communications* 10.4 (2014), pp. 372–392.

[Van+17] F. Van den Abeele et al. “Secure Service Proxy: A CoAP(s) Intermediary for a Securer and Smarter Web of Things”. In: *Sensors* 17.7 (2017), pp. 1–30.

[Vuč+15] M. Vučinić et al. “OSCAR: Object Security Architecture for the Internet of Things”. In: *Ad Hoc Networks* 32 (June 2015), pp. 3–16.

[Wou+14] P. Wouters et al. *Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)*. RFC 7250. Fremont, CA, USA: RFC Editor, June 2014, pp. 1–18.

Performance Evaluation of Group OSCORE for Secure Group Communication in the Internet of Things

Abstract

The Constrained Application Protocol (CoAP) is a major application-layer protocol for the Internet of Things (IoT). The recently standardized security protocol OSCORE efficiently provides end-to-end security of CoAP messages at the application layer, also in the presence of untrusted intermediaries. At the same time, CoAP supports one-to-many communication, targeting use cases such as smart lighting and building automation, firmware update, or emergency broadcast. Securing group communication for CoAP has additional challenges. It can be done using the novel Group OSCORE security protocol, which fulfills the same security requirements of OSCORE in group communication environments. While evaluations of OSCORE are available, no studies exist on the performance of Group OSCORE on resource-constrained IoT devices.

This paper presents the results of our extensive performance evaluation of Group OSCORE over two popular constrained IoT platforms, namely Zolertia Zoul and TI Simplelink. We have implemented Group OSCORE for the Contiki-NG operating system and made our implementation available as open-source software. We compared Group OSCORE against unprotected CoAP as

Martin Gunnarsson, Krzysztof Mateusz Malarski, Rikard Höglund,
Marco Tiloca "Performance Evaluation of Group OSCORE for Secure
Group Communication in the Internet of Things". In *Transactions on Internet of Things*, vol. 3,
issue 3, Aug. 2022, ACM.

well as OSCORE. To the best of our knowledge, this is the first comprehensive and experimental evaluation of Group OSCORE over real constrained IoT devices.

1 Introduction

In the recent years, we have been witnessing a massive Internet of Things (IoT) rollover, which is further accelerated under the umbrella of 5G deployment and is expected to result in 55.7 billions of connected devices in 2025 [IDC20]. Thousands of resource-constrained sensors and actuators are being deployed in farms, factories, smart homes and buildings, as well as public spaces. The range of relevant applications encompasses environmental monitoring, automated building or infrastructure control, remote metering, telemedicine and many more. It goes without saying that ensuring efficient performance as well as fulfilling security requirements are of paramount importance.

For several applications, it is additionally convenient to rely on a group communication model, where a single transmitted message targets multiple devices, with all of them as intended recipients. This communication model especially suits some typical and renowned use cases, including but not limited to device and appliance control in smart buildings (e.g., smart lighting and door locks), discovery of resources and services in the network environment, distribution of software and firmware updates (e.g., functionality upgrades and vulnerability patches), as well as emergency broadcast. Evidently, interaction and message exchange among IoT devices have to be lightweight, robust and secure also for applications taking advantage of such a group communication model.

As of today, most notable application-layer transfer protocols used for the IoT are MQTT [Ban+17] and the Constrained Application Protocol (CoAP) [SHB14], both of which are available as open standards.

MQTT relies on a publish-subscribe models, where publisher clients can disseminate messages organized into topics among subscriber clients. This is mediated by a Broker server, as responsible for dispatching messages of a certain topic towards subscribers to that topic. MQTT mainly works over the Transport Layer Protocol (TCP) and it can enjoy secure communication between the clients and the broker by means of the Transport Layer Security (TLS) protocol suite [Res18].

CoAP builds on the REpresentational State Transfer (REST) model [Fie00a], and thus focuses on client-server interactions for manipulating and trasferring the state of server resources. As a consequence, CoAP enables seamless cooperation with the ubiquitous and also REST-based HyperText Transfer Protocol (HTTP) [FR14] used in the Internet, while introducing a small and limited message overhead. CoAP runs mainly over User Datagram Protocol (UDP) and provides optional features such as the event-based resource Observation [Har15]. Also, CoAP natively supports deployed intermediaries such as proxies, which typically provide additional services such as caching of responses and translation of underlying transport protocol. Hereafter, we focus on CoAP as the specific target of the security protocol evaluated in this paper.

As to secure communication, CoAP initially considered only the Datagram Transport Layer Security (DTLS) [RM12] protocol suite to protect exchanged messages at the transport layer. Recently, the protocol Object Security for Constrained RESTful Environments (OSCORE) has been published as RFC 8613 [Sel+19a], and it enables the end-to-end protection of CoAP messages at the application layer, independent of the underlying transport protocol and preserving messages mostly opaque to possible (untrusted) intermediaries. OSCORE has attracted attention from the industry [u-b20; Eri19] and is the application-layer security solution used in the IoT device management standard OMA LwM2M [All20a][All20b].

Furthermore, CoAP natively supports one-to-many group communication scenarios, where a single instance of a message intended to multiple recipients is dispatched to all of them at once, e.g., using Internet Protocol (IP) over multicast [RD14][DWT21]. This makes CoAP particularly suitable to serve the use cases mentioned above. However, with respect to security, group communication introduces additional challenges and especially DTLS cannot be used to provide message protection in group communication scenarios. In order to fill this gap, the new security protocol Group OSCORE [Til+21a] has been proposed and is under ongoing standardization in the Internet Engineering Task Force (IETF). In particular, Group OSCORE extends and adapts OSCORE to work in group communication scenarios, while fulfilling the same security requirements. That is, Group OSCORE protects CoAP messages end-to-end at the application layer, and provides replay detection, confidentiality, integrity and source authentication of exchanged messages. In particular, the group mode of Group OSCORE that this paper focuses on provides source authentication of messages by means of digital signatures. The communication model considers protected request messages sent to multiple recipients at once, as well as the corresponding protected, cryptographically bound multiple responses. The design and development of Group OSCORE are also actively supported by the industry [Eri19].

Even though Group OSCORE is suitable for applications using CoAP in group communication environments, it clearly requires to be tested on commercially available IoT hardware platforms, on which its performance and impact ought to be assessed. However, while evaluations of OSCORE have been carried out and made available [Gun+21], no studies have been made so far on the feasibility and performance of Group OSCORE on resource-constrained IoT devices. Note that, while the Group OSCORE specification at [Til+21a] focuses on the theoretical protocol design and practical considerations, it does not refer to any particular implementation nor does it provide an experimental evaluation.

This raises the following questions: *How does an implementation of Group OSCORE perform in a setup of real constrained IoT devices relying on CoAP group communication over IP multicast? How does it quantitatively impact their performance,* *especially due to the use of digital signatures? How is this in turn affected by the use of different signature algorithms? Ultimately, is it a feasible choice to implement and use in a scenario with constrained devices relying on group communication?*

In this paper, we address these questions and especially fill the lack of performance assessments by providing an experimental performance evaluation of the Group OSCORE security protocol on two popular constrained IoT platforms. Specifically, we implemented Group OSCORE in the Contiki-NG operating system [Con21], and made our implementation available as open-source software[1]. Then, we used our implementation to perform experiments on real low-power IoT devices, considering the two platforms Zolertia Firefly Rev.A board (Zoul) [Zol20] with the CC2538 System on a Chip (SoC) and TI Simplelink with the CC1352R1 SoC [Ins21].

We considered a setup consisting of a constrained CoAP client exchanging messages with three constrained CoAP servers. Both one-to-one and one-to many unprotected communications, as well as one-to-one communications protected with OSCORE and one-to-many communications protected with Group OSCORE were evaluated. Similarly, when OSCORE or Group OSCORE were evaluated, cryptographic operations were performed either in software or through hardware acceleration. Specifically for Group OSCORE, we considered the two different signature algorithms ECDSA P-256 and EdDSA25519. We experimentally evaluated performance and resource utilization on the server side for both IoT platforms, in terms of Random Access Memory (RAM)/Read-Only Memory (ROM), Central Processing Unit (CPU) and energy consumption. Furthermore, we investigated the Round Trip Time (RTT) as perceived by the client during the message exchanges.

As key takeaways from our evaluation, we can show that the memory utilization in terms of RAM and ROM is manageable for constrained IoT devices. The use of Elliptic Curve Cryptography (ECC) signatures adds a non-negligible delay to a full message exchange and significantly contributes to the energy consumption. However, the EdDSA signature algorithm displays considerably better performance than ECDSA P-256. Even though the complexity of ECC cryptography adds such penalties, Group-OSCORE remains feasible for low-power applications.

We can summarize the key contributions of this paper as follows.

- Our implementation of Group OSCORE in Contiki-NG, which is to the best of our knowledge the first open source and publicly available implementation of Group OSCORE suitable for constrained IoT platforms.

- An extensive performance evaluation of Group OSCORE on real IoT devices of two commercially-available hardware platforms, based on our open-source implementation for Contiki-NG. To the best of our knowledge, this

[1] `https://github.com/Gunzter/Contiki-ng/tree/group_oscore` — Accessed: 2022-01-21

is the first publicly available experimental evaluation of Group OSCORE over commercial IoT devices.

The rest of this paper is organized as follows. Section 2 discusses the related work. Section 3 overviews relevant use cases relying on group communication and related security requirements. Section 4 introduces relevant background technologies and concepts, while Section 5 provides a high-level description of the Group OSCORE protocol. Section 6 describes the methodology and the setup considered in our performance evaluation, while in Section 7 we present and discuss our experimental results. Finally, Section 8 draws our conclusive remarks and highlights possible future work.

2 Related work

Previous works have considered secure group communication, enforced at different layers in the communication stack. In 2017, [GFK17] characterized typical security properties desired in secure group communication, covering message protection and main aspects of group management, and surveyed some known approaches. This section focuses on message protection in group communication and overviews the main current approaches in a bottom-up fashion. The specific layer enforcing security may depend on the device capabilities and the requirements of the application scenario.

At the data link layer, the IEEE 802.15.4 Standard [IEE20] can also be used to provide secure group communication, by using a single group cryptographic key commonly shared among all network nodes. For most of the resource-constrained platforms commercially available, the actual cryptographic operations are typically available in hardware. Securing communications at the data link layer presents a number of limitations. First, link layer frames are protected hop-by-hop, thus they have to be decrypted and re-encrypted at each network node on the path between the original sender and the final recipient. This considerably affects network performance and message delivery time, especially in multi-hop setups, and yields a non-negligible overhead in terms of energy consumption [DDT11]. Second, like for other link layer protocols, IEEE 802.15.4 does not consider the establishment and management of cryptographic keys, and blindly entrusts this to the higher layers in the stack [IEE20]. Third, as using only a common group key to protect group communication, it is not possible to ensure source authentication of messages exchanged in the group.

At the network layer, security can be provided by the protocol suite IPsec [SK05][Bri10], with security associations established through the IKEv2 protocol [Kau+14]. The IPsec suite has been adapted to work in group communication scenarios [GWI08], and an analogous adaptation of IKEv2 has been defined [SW21][Riz+19]. In order to foster the use of IPsec also for constrained IoT devices, additional work has been done. This includes an extension of 6Low-

PAN to provide header compression also for IPsec [Raz+11], a further adaptation of the IKEv2 protocol, i.e., Minimal-IKEv2 [Kiv16], and Diet-ESP [Mig+17], i.e., an adaptation of the IPsec protocol Encapsulated Security Payload (ESP). In particular, by leveraging an ESP Header Compression (EHC) strategy, Diet-ESP achieves an overhead which is limited and smaller than the one of ESP. As noted in [GFK17], such approaches used in group communication scenarios do not provide source authentication of messages, but only confidentiality and group authentication. More generally, source authentication of exchanged IPsec packets can be enforced again by embedding digital signatures computed with the RSA algorithm, as defined in [Wei06]. However, this has not been defined for more recent, efficient and secure signature algorithms (e.g., ECDSA and EdDSA), and the responsible Working Group MSEC[2] of the IETF terminated its activities.

The work in [TNR17][Til+15] proposed an approach to secure group communication for the IoT at the transport layer, through an adaptation of the Datagram Transport Layer Security (DTLS) protocol suite [RM12], which originally supports only one-to-one communication among two peers. This approach relies on a pre-established Group Security Association (GSA), including common key material shared among the group members. Building on that, sender nodes use the common group key material to protect one-to-many multicast request messages addressed to the other group members, which can individually reply back. Similarly to other techniques above, this approach relies only on shared cryptographic material, hence it does not provide source authentication of exchanged messages.

In [CNN20], the authors proposed a similar DTLS-based approach for group communication, with additional focus on key provisioning. This relies on a centralized Group Controller that, upon the joining of new group members, provides those with a common group key for protecting request messages sent over IP multicast. When receiving a multicast request from a certain sender in the group for the first time, a listener asks the Group Controller for a pairwise key to share with that sender and to protect unicast responses sent to that sender. When receiving a response from a certain listener for the first time, a sender asks the Group Controller for the same pairwise key. Also in this case, since requests sent over multicast are protected with the same common group key, their source authentication is not ensured.

In the presence of intermediaries such as proxies, these DTLS-based approaches do not provide end-to-end security between the original data producer and the final data consumer(s) in the group. In fact, in order to enable proxy operations on CoAP messages, a DTLS session at a sender node would have to terminate at the proxy [Sel+19a][SPH17], which has to use a separate DTLS session with the final recipient(s). This both worsens performance due to the double security processing at the proxy [Gun+21], and requires to trust the proxy beyond what minimally required in the application scenario.

[2] `https://datatracker.ietf.org/wg/msec/about/` — Accessed: 2022-01-21

The Group OSCORE protocol [Til+21a] evaluated in this paper provides an efficient and lightweight solution for secure group communication. In particular, Group OSCORE operates at the application layer, although not within the actual application. As explained more in detail in Section 5, Group OSCORE transforms an unprotected CoAP message into an equivalent protected CoAP message, which allows to use Group OSCORE over any one-to-many transport where CoAP works. As a result, Group OSCORE fills the gaps and limitations of the approaches above, i.e.: it ensures end-to-end security between the original data producer and the final data consumer(s), also in the presence of (untrusted) intermediaries; it ensures source authentication of exchanged CoAP messages, through either digital signatures or pairwise symmetric keys derived from pairwise Diffie-Hellman secrets; it ensures crypto agility and extensibility in the space of encryption and signature algorithms, by using the standard CBOR Object Signing and Encryption (COSE) [Sch21][Sch20]; it can seamlessly rely on the off-the-shelf approach for group joining and key provisioning defined in [TPP21], in turn based on the ACE framework for authentication and authorization in constrained environments [Sei+21].

Alternative approaches for group communication leverage multi-party content dissemination, rather than message delivery on actual one-to-many links over multicast. This notably includes the open standard MQTT [Ban+17] based on the publish-subscribe model, where publisher clients can send messages for a certain topic to a Broker server, which in turn forwards those messages to the subscriber clients that have registered their interest for that topic. MQTT mainly works over TCP, and communications between the Broker and each client can be secured with the TLS protocol suite [Res18]. The work in [CLC20] proposed an extension to provide end-to-end protection of content exchanged between the clients. That is, the Broker provides each registered client with a pairwise session key, as well with a per-topic group key encrypted with that session key. As all the clients interested in a topic share the same group key, a publisher (subscriber) client can encrypt (decrypt) published content by using that group key, thus the Broker does not perform a decryption & re-encryption process.

It is possible to use CoAP to set up an application based on the publish-subscribe model, as defined in [KKJ19]. In such a case, a CoAP server acts as Broker, while the CoAP clients acting as subscribers use CoAP Observe [Har15] on the Broker's resources associated to their topics of interest. By doing so, the subscribers receive unsolicited notifications from the Broker when the topic resources change their content, following a new publication on such topics from the CoAP clients acting as publishers. Secure communications can be provided between the Broker and each client as usually in CoAP, i.e., by using OSCORE and/or DTLS. Besides, [Til+21b] defines how to further extend this model, by enabling the Broker to send a single notification to all the subscribers over multicast. These notifications can be protected using Group OSCORE, having all the clients and the Broker configured as members of the same group.

Lizardo *et al.*, proposed Sharelock [Liz+21], a security protocol that provides end-to-end security and confidentiality of messages exchanged by groups of communicating nodes. In particular, the group members include different clusters of IoT devices behind untrusted Edge Servers used as communication intermediaries, as well as Cloud Servers as intended recipients and consumers of data sent by the IoT devices. Also, the group members are directly responsible for managing and establishing keys in the group, as well as for managing group memberships and consequent update of group key material.

In [Gün+21], the authors considered multi-party dissemination of content based on the information-centric Named-Data Networking (NDN) architecture [Jac+09]. In particular, the study used CoAP in an NDN-based setup, where clients retrieve the content of interest from the origin server only as a last resort, while greatly relying on that content being available at a closer intermediary, that has cached previous responses conveying that content. If secure communication is ”naively” introduced with OSCORE, the nature of OSCORE limits the benefits of caching to exact request-response pairs, thus helping only in case of request retransmission. However, [Gün+21] has additionally considered the particular use of Group OSCORE proposed in [AT21], and shown that it effectively enables full-fledged caching of OSCORE-protected responses to a same *deterministic* request for the same content, thus considerably reducing content retrieval time.

Finally, regardless the underlying message transport, a number of stand-alone cryptographic schemes may also be supportive of group communication, although they fulfill only some of the expected security requirements. One of these schemes is μTesla [Per+02], i.e., an adaptation for constrained devices of the Tesla scheme [Per+00] originally designed for streaming applications. While it relies on symmetric keys to provide authentication and thus displays very efficient performance, μTesla provides neither source authentication nor confidentiality. Further schemes include Identity-Based Signature (IBS) [Bae+13] and Attribute-Based Encryption (ABE) [YCT15], which however provide only source authentication and integrity without confidentiality, and only confidentiality without authentication and integrity, respectively.

3 Relevant use cases

This section provides a high-level overview of most relevant use cases that benefit of one-to-many, group communication. In addition, Section 3.1 lists a number of security requirements that such use cases expect to be fulfilled by the adopted security solutions and protocols. Most relevant use cases can be roughly organized into the following different categories, for which we provide some of the most representative examples.

- **Device control in group settings**. A first simple, yet effective and recurring use case concerns the control of lighting appliances. For example, a group can include lighting devices (acting as servers) and switch devices (acting as clients) deployed in a same room, corridor, floor, or open environment. Switch devices can then be used to control the lighting devices, by sending a single on/off/dimming command at once to all the lighting devices in the group, e.g., over IP multicast or other one-to-many technologies. Connectivity between lighting devices may be achieved, for instance, by means of IPv6 and (border) routers supporting 6LoWPAN.

A more advanced use case concerns the integrated control of smart buildings. While leveraging the same principles, this makes it possible to efficiently check and control the operational status of multiple types of appliances, such as physical sensors, heating, ventilation and air-conditioning units, or locks of doors and windows. Furthermore, groups can be configured in order to reflect not only the devices' physical positioning, but also their types and capabilities. Controlled devices may respond providing the result of the operation as well as their current operational status. As additional support, border routers connected to an IP network backbone (which is also multicast-enabled) can be used to interconnect routers in the building with each other, hence covering larger environments spanning over multiple floors.

In [Pol18], the Fairhair Alliance described its Security Architecture for IoT-based building automation, which especially relies on the Group-OSCORE protocol considered in this paper for securing CoAP group communication.

- **Discovery of resources and services**. Group communication allows to conveniently reach out multiple devices at once, to discover their precise physical positioning as well as their hosted particular resources or services, possibly based on filter criteria, such as ID, name, protocol, version and type. To this end, groups can be configured to specifically reflect that their members are devices with similar capabilities and features, or a common physical location. Queried devices may reply back, to notify about their presence, provide the requested set of information as well as their current operational status.

- **Software update**. Instead of sending software updates separately to each individual device, group communication enables the efficient delivery of common updated data to a large set of devices at once, with benefits in terms of performance. That is, it yields a reduced network load and overall time latency for providing the data to all the intended recipient devices. The software update can consist in an application component, a firmware image or patch, or a set of parameter values for a single common configuration update. Devices receiving software updates typically reply, to provide feedback on the result of the update operation and their current operational status.

- **Emergency multicast**. Particular situations, such as a safety crisis or a natural disaster, require a notifier to quickly broadcast emergency related information to multiple devices as part of a wide audience. The recipient devices may reply back, providing feedback and local information concerning the ongoing emergency.

3.1 Security Requirements

Typical applications relying on group communication have the following security requirements, and expect them to be fulfilled by the used security solutions and protocols. As per Section 4, Group OSCORE fulfills all these requirements.

- **Data confidentiality**. Group-level data confidentiality must be ensured for all messages exchanged in a given group. This is achieved by encrypting messages at a group level, through security material shared by the members of that group. As a result, a message can be decrypted by any member of the group, but not by an external adversary or other external entities. This must apply both to one-to-many request messages as well as to the corresponding multiple response messages. Group OSCORE provides this when used in Group Mode.

  Some security protocol may additionally provide pairwise data confidentiality for messages exchanged in a one-to-one fashion within the group, i.e., as sent to a single group member rather than to the whole group. This can be achieved by encrypting such messages using pairwise security material, which is shared only between two group members. As a consequence, only the intended single recipient is able to correctly decrypt the message. Group OSCORE provides this when used in Pairwise Mode.

- **Data replay protection**. It must be possible for a group member to detect whether an incoming message exchanged in the group has been replayed.

- **Source message authentication**. It must be possible for a group member to verify that an incoming message was indeed originated by the specific group member identified as alleged sender. Group OSCORE ensures this by using digital signatures appended to messages, when used in Group Mode; or by encrypting messages with a pairwise key derived from two group members' asymmetric key material, when used in Pairwise Mode.

- **Message integrity**. It must be possible to ensure that a message sent in the group has not been tampered with while in transit, either by another group member, an external adversary or any other external entity which is not a group member. This is practically achieved by the same means used to ensure message authentication.

- **Message ordering**. A group member must be able to determine the ordering of incoming messages from each different sender in the group. In this respect, Group OSCORE especially ensures absolute freshness of response messages that are not notifications [Har15], and relative freshness of request messages and notification responses. On the other hand, it is typically not required to determine the ordering of messages from different senders.

Fulfilling the requirements above concerns the actual secure communication protocol used to protect messages exchanged in the group, such as Group OSCORE. Furthermore, practical applications also rely on additional mechanisms for provisioning cryptographic keys and other security material to the group members, both upon their joining the group or later on if needed during their operation. Typically, this can be achieved through a logically centralized Key Distribution Center, which maintains a dedicated secure communication association with each group member, for each of the groups it is responsible for. A Key Distribution Center suitable for groups where the Group OSCORE protocol is used has been proposed in [TPP21] and is under ongoing standardization.

Finally, applications may require that the security material used in the group is revoked, and new one is distributed, upon a change in the group membership. That is, renewing the security material upon a new member's joining will ensure backward secrecy, since the new member would not be able to access messages exchanged in the group before its joining (even if it recorded them). Also, renewing the security material upon a member's leaving (e.g., if compromised or suspected so) will ensure forward secrecy, since the leaving member would not be able to access messages exchanged in the group after its leaving.

The Group OSCORE protocol specifically requires that the security material in the group is renewed in case of a member's leaving. Nevertheless, Group OSCORE is agnostic of the particular approaches and key management scheme used to renew the group security material and fulfill the related requirements discussed above. Further details on the secure, efficient revocation and re-distribution of security material in the group is out of the scope of this paper.

4 Technical background

In this section, we introduce background technical concepts referred throughout the rest of the paper.

4.1 CoAP

CoAP is a lightweight application-layer protocol, intended to support applications for constrained networks and with the aim of integrating massive IoT in the existing Internet infrastructure [SHB14]. Since the contemporary Internet services (web-browsing, online trade and banking, to name a few) are based on

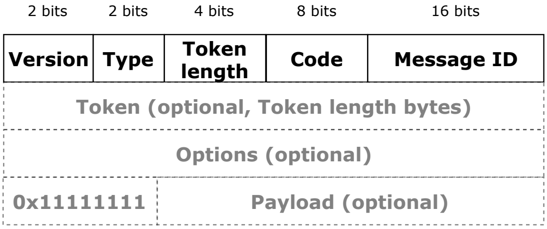


Figure 1: Format of a CoAP message.

the HTTP/REST architecture [FR14; Fie00a], the design of CoAP follows the same principles, being HTTP/REST-compliant and providing analogous operation methods such as GET, PUT, POST and DELETE.

Furthermore, CoAP was firstly designed to operate over unreliable transport, especially the UDP protocol [Pos80], and was later extended to possibly operate over reliable transports [Bor+18], such as the TCP protocol [Pos81]. CoAP enables synchronous and asynchronous communications, and considers low-power IoT deployments with multiple years on battery expected. In particular, it supports messages as small as 4 bytes[3] and can work seamlessly with network proxies or caches, which can be used to offload the constrained devices or schedule the message delivery so that devices can remain in sleep, power-saving mode for longer time.

As shown in Figure 1, a CoAP message consists of some mandatory header fields (coloured black) and optional additional elements (coloured gray). The header specifies the release of CoAP in use (Version); the type of the message (i.e., Confirmable, Non-Confirmable, Acknowledgement or Reset); the length of the Token, which correlates a response to a request; the CoAP Method Code indicating the Request Method in request messages (e.g., GET, PUT, etc.) or the Response Code in response messages; and the Message ID, which correlates an Acknowledgment to a Confirmable request/response and allows perform message deduplication. None or more CoAP options with different sizes may follow, to provide additional information regarding the communication as well as the message encoding and expected handling. Finally, should application data be present as message payload, it needs to be prepended by a 1-byte marker with all bits set to one.

Group communications

As CoAP is suitable for the IP stack, it is possible to transmit the requests over UDP [Pos80] and IP multicast [DWT21], thus decreasing the amount of data to be transmitted in order to deliver the same payload to multiple recipients (CoAP

[3] For the sake of comparison: only identifying the protocol version (e.g., "HTTP/1.1") already takes 8 bytes in a HTTP message.

servers). Instead of addressing the request to a specific endpoint, the CoAP client sends the request message towards the multicast IP address of the group. The server devices that registered for listening to the group address can respond to the client in a unicast fashion[4]. All CoAP request messages sent over IP multicast can only be Non-Confirmable (i.e., with no Acknowledgement expected). The servers must delay their responses to the client by a randomized value, namely *leisure time*, which must be equal to or longer than:

$$leisure_{min} = S * G/R \tag{1}$$

where S corresponds to the message size, G denotes the group size and R symbolises the transmission data rate. Such an approach reduces the chances of simultaneous transmissions from the servers in the group, which would result in colliding responses and possible network congestion. The risk of network congestion may also be controlled by setting the servers as silent nodes that never reply with a response and/or by suppressing certain responses (e.g., error responses of a certain error class).

Rather than IP multicast, applications that want to use CoAP for group communication can rely on alternative technologies providing one-to-many message delivery. For example, a CoAP request message addressed to a group of devices can be directly transported as payload of an IEEE 802.15.4 broadcast frame [IEE20]. In the rest of this paper, we focus on group communication for CoAP using UDP and IP multicast as per [DWT21].

4.2 OSCORE

OSCORE [Sel+19a] is a recently standardized application-layer security protocol that, unlike DTLS, provides end-to-end security between the original data producer and the final data consumer. Instead of protecting the whole communication channel between the client and the server, the protocol is ”CoAP-aware” and encrypts only the parts of the CoAP message that require confidentiality, while the fields meant to be used by proxies are left unprotected, or only integrity protected. In particular, OSCORE provides *end-to-end* encryption, integrity protection, source authentication and replay protection of messages, while displaying smaller power consumption and memory burden on constrained devices when compared to DTLS [Gun+21].

At a high-level, OSCORE takes a CoAP message as input and produces as output a new protected CoAP message, namely an OSCORE message. The reverse process occurs when an OSCORE message is received, and the original CoAP message is recomputed. Furthermore, OSCORE is independent of the specific transport layer, and it works wherever CoAP works. Also, it is possible to combine

[4] Note that the server’s IP address is the source address of the response.

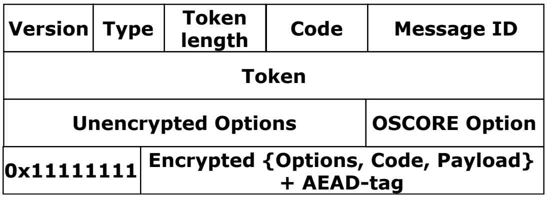


Figure 2: Format of an OSCORE-protected CoAP message.

OSCORE with communication security on other layers, e.g., to further protect an OSCORE-protected message using DTLS.

The lightweight design of OSCORE leverages the efficient and small-size encoding scheme Concise Binary Object Representation (CBOR) [BH20]. That is, the data to be protected composes a CBOR structure, which is then encrypted and authenticated by using COSE [Sch21][Sch20], thus yielding a COSE object which is transported in the protected CoAP message. Therefore, OSCORE follows the *object security* paradigm, where each data chunk is secured separately.

Before they can exchange secure data, two CoAP nodes need to establish an OSCORE Security Context. However, as focused only on message protection, OSCORE itself does not provide a mechanism to do so, i.e., one equivalent to the Handshake protocol of the DTLS suite. On the other hand, a number of approaches have been developed to let two CoAP nodes establish an OSCORE Security Context. These include, for instance, the lightweight key establishment protocol Ephemeral Diffie-Hellman Over COSE (EDHOC) [SMP21], which is currently an IETF standardization proposal.

An OSCORE Security Context contains three parts, i.e., a Common Context, a Sender Context and a Recipient Context. The Common Context is identical for both nodes, and specifies the HMAC-based Key Derivation Function (HKDF) and Authenticated Encryption with Associated Data (AEAD) algorithms, as well as information used to identify the Security Context and to derive the cryptographic keys and nonces (i.e., Context ID, Common IV, Master Secret and Master Salt). Each of the two nodes uses its Sender Context to protect outgoing messages addressed to the other peer, and its Recipient Context to decrypt incoming messages from the other peer. Given a certain node, its Sender (Recipient) Context contains the Sender ID of that node (the Recipient ID that the other node uses as its own Sender ID), the symmetric key used to encrypt (decrypt) outgoing (incoming) messages to (from) the other peer, and a sender sequence number (a replay window). Evidently, the Sender Context of one node mirrors the Recipient Context of the other node. Furthermore, a pair of nodes must ensure that their Context ID, Sender ID, Master Secret and Master Salt (thus also the derived symmetric keys and AEAD nonces) are unique; otherwise, the security features of OSCORE cannot be guaranteed. Note that each of the two nodes can act as only CoAP client, only CoAP server, or both.

Figure 2 shows the format of an OSCORE-protected CoAP message. The procedure that produces such a message as output can be summarized as follows. A more detailed description is available in the OSCORE specification [Sel+19a].

1. All the CoAP options that are not needed for proxying, the original CoAP Method Code and the original CoAP payload (if any) are considered as plaintext to be encrypted.

2. The CoAP options to be only integrity protected, the ”Version” field of the CoAP header and additional OSCORE-related information compose the Additional Authenticated Data (Additional Authenticated Data (AAD)), to be integrity protected only.

3. The plaintext and the AAD are included in a new COSE object.

4. The agreed cryptographic algorithm is used to encrypt and authenticate the COSE object, by taking as input: the plaintext and AAD; the Sender Key; and a nonce derived from the Sender Sequence Number and other information from the Security Context. Note that a response may be protected by reusing the same nonce of the request; this spares the server from using its own fresh Sender Sequence Number and from including it in the response, thus yielding a smaller message as result.

5. The resulting ciphertext of the COSE object and its AEAD-tag become the payload of the OSCORE-protected message.

6. The OSCORE-protected messages is finalized, by: i) adding an OSCORE option, with information that enables the recipient to decrypt the message; and ii) replacing the original CoAP Method Code in the CoAP header to indicate the POST method (or FETCH, if CoAP Observe is used [Har15]) in request messages, and the Response Code 2.04 ”Changed” (or 2.05 ”Content”, if CoAP Observe is used) in response messages.

The recipient of the OSCORE-protected message decrypts and authenticates the ciphertext using the Recipient Key from its Recipient Context. Also, for request messages, it checks the Sequence Number conveyed in the OSCORE option against its Replay Window, in order to enforce replay protection.

5 Group OSCORE

In our previous work [Gun+21], we showed that the OSCORE protocol has performance advantages over DTLS in terms of energy consumption and communication latency, on top of its pivotal end-to-end communication security in the presence of intermediaries. However, the applicability and security guarantees of

OSCORE are limited only to unicast communications, where messages are exchanged strictly between two CoAP nodes.

As a step forward to provide security in use cases relying on group communication (see Section 3), new work started for adapting OSCORE to group communication environments (see Section 4.1). This resulted in the ongoing standardization activities around the Group OSCORE protocol [Til+21a]. This provides the same security properties of OSCORE, while protecting a one-to-many CoAP requests addressed to multiple servers at once, as well as the multiple corresponding CoAP responses. At the time of writing, Group OSCORE is the only available method to protect group communication based on CoAP, and is the mandatory security solution for CoAP over UDP and IP multicast [DWT21].

The rest of this section overviews Group OSCORE, presenting its main features as much as possible in comparison with OSCORE. A more detailed description is available in the Group OSCORE specification draft [Til+21a].

5.1 General Properties

Group Object Security for Constrained RESTful Environments (Group OSCORE) extends and adapts OSCORE to work also in group communication scenarios. In particular, Group OSCORE provides end-to-end security of CoAP messages exchanged between members of a group, e.g. using IP multicast.

Group OSCORE ensures cryptographic binding between a CoAP group request, sent by a client to multiple servers, and the corresponding CoAP responses individually sent by the servers in the group. Since message protection builds on commonly shared, group keying material, source authentication of messages exchanged in the group is achieved by using asymmetric keying material. As explained below, this relies on either digital signatures when using the *group mode*, or on pairwise keys derived from asymmetric, individual keying material when using the *pairwise mode*.

Like OSCORE, Group OSCORE is independent of the specific transport layer, and it works wherever CoAP works. Also, like with OSCORE, it is possible to combine Group OSCORE with communication security on other layers. One example is the additional use of DTLS, between one client and one proxy (and vice versa), or between one proxy and one server (and vice versa), to secure and hide from external observers also information left unprotected by OSCORE, such as CoAP options intended for intermediary proxies to perform message forwarding. Note that DTLS cannot be used to secure messages sent over IP multicast or other one-to-many message delivery technologies.

Group OSCORE has two different modes of operation, as different ways to protect CoAP messages. It is up to the application to decide in which particular mode a message has to be protected, possibly on a per-message basis.

- In the *group mode*, a message is encrypted with symmetric keying material available to all group members, and includes an additional signature computed by using the private key of the sender CoAP endpoint. The group mode supports signature verification by intermediaries external to the group, e.g. gateways. In the rest of this paper, we focus on the *group mode*, especially when providing a performance evaluation of Group OSCORE (see Section 7). A more detailed description of how the group mode works is provided in Section 5.4.

- In the *pairwise mode*, two group members can exchange unicast messages, as protected only with pairwise symmetric keys and not including a signature. These symmetric keys are derived from Diffie-Hellman shared secrets, calculated with the asymmetric keys of the two group members. Since a signature is not included, this results in a smaller message overhead. This method is applicable to one-to-one messages sent in the group, i.e., responses as individually sent by servers, and requests addressed to one single server.

  The pairwise mode has minimal differences from the original OSCORE processing, i.e.: the used symmetric keys are derived from the asymmetric keys of the two group members; it uses the same extended AAD defined for the group mode of Group OSCORE (see Section 5.3); the inclusion of the 'kid' and 'kid_context' parameters in the OSCORE option works as in the group mode of Group OSCORE (see Section 5.3). The pairwise mode allows a large number of endpoints in the same group to perform OSCORE-protected pairwise communication with one another, while keeping the related key provisioning effort and overhead small, as it is limited to interactions with the responsible Group Manager and it mostly occurs when joining the OSCORE group (see Section 5.2).

Like OSCORE, Group OSCORE provides message binding of responses to requests, which in turn provides relative freshness of responses, and replay protection of requests. In particular, Group OSCORE fulfills the following security objectives: data replay protection; source authentication; message integrity; group-level data confidentiality (in group mode) or pairwise data confidentiality (in pairwise mode); proof of group membership with respect to a message sender.

5.2 The Group Manager

Group OSCORE relies on the presence of an additional trusted entity acting as *Group Manager*. This is responsible for one or more OSCORE groups, for the respective Group Identifier (Gid) used as OSCORE ID Context, and for the Sender ID and Recipient ID of the respective group members.

The Group Manager has exclusive control over the Gid values uniquely assigned to the different groups under its control, and over the Sender ID and Recipient ID values uniquely assigned to the members of each of those groups. A CoAP endpoint receives the Gid and other OSCORE input parameters, including its Sender ID, from the Group Manager upon joining the OSCORE group. That Sender ID is valid only within that group, and is unique within the group.

Furthermore, the Group Manager stores and maintains the public keys of endpoints joining a group, and provides information about the group and its members to other current group members. At any time, a group member can retrieve from the Group Manager the public key and other information associated to other group members.

It is recommended that the Group Manager takes care of the group joining by using the approach defined in [TPP21], as based on the ACE framework for authentication and authorization in constrained environments [Sei+21]. Further details on the method used for joining an OSCORE group are out of the scope of this paper.

Renewal of Group Keying Material

Due to a number of reasons, the Group Manager may force the members of an OSCORE group to establish a new Security Context, by revoking the current group key material and distributing new one (rekeying). To this end, a new Group Identifier (Gid) for the OSCORE group and a new value for the Master Secret parameter is distributed to the group members. When doing so, the Group Manager may additionally distribute also a new value for the Master Salt parameter, while it should preserve the same current value of the Sender ID of each group member.

Then, each group member re-derives the key material in its own Sender Context and Recipient Contexts (see Section 5.3), using the newly distributed Gid and Master Secret parameters. The Master Salt used for the re-derivations is the newly distributed Master Salt if provided by the Group Manager, or an empty byte string otherwise. Thereafter, each group member uses its latest installed Sender Context to protect its own outgoing messages.

When a current endpoint leaves the group, the Group Manager renews the group key material and informs the remaining members about the leaving endpoint. This keeps the group members able to correctly assert the group membership of a message sender, and additionally preserves forward secrecy in the group. Furthermore, in accordance with the specific application requirements, it is recommended to rekey the group also when a new joining endpoint is added to the group, thus preserving backward secrecy as well.

Group OSCORE is not devoted to a particular key management scheme for rekeying the OSCORE group. However, the Group Manager should support the distribution of the new Gid and Master Secret parameter to the OSCORE group according to the Group Rekeying Process defined in [TPP21]. Alternatively, different more advanced and efficient methods for group rekeying can be used from the many available in the literature, such as [WGL00][WHA99][DS11][TD13][TD16] [Til+20].

5.3 Main Differences from OSCORE

This section introduces in which respects Group OSCORE differs from OSCORE, with a focus on the data structure and key material stored by group members, as well as the COSE object and compressed encoding of OSCORE messages.

As a particular case, a group member can assume the special role of *silent server*. This kind of endpoint is interested in receiving request messages, but never replies to them. An endpoint can implement both a silent server and a client, as the two roles are independent. However, an endpoint implementing only a silent server processes only incoming requests, maintains less keying material, and especially does not have a Sender Context for the OSCORE group.

The Security Context

Each member of an OSCORE group stores a Security Context (see Section 4.2), which is extended as follows with the respect to the original format considered in OSCORE.

- The Common Context, shared by all the group members, specifies also: i) the Signature Encryption Algorithm used to encrypt messages when using the group mode; ii) the Signature Algorithm used to compute the message signature when using the group mode; iii) the Pairwise Key Agreement Algorithm used to derive pairwise symmetric keys, when using the pairwise mode. Possible parameters associated to these algorithms are embedded in the stored public keys of group members. The AEAD Algorithm parameter inherited from the original format of the Security Context specifies the encryption algorithm used to protect messages with the pairwise mode.

  Furthermore, the ID Context parameter contains the Group Identifier (Gid) of the OSCORE group, which is thus used as Context ID for that group. The choice of the Gid is specific to the application running at the Group Manager. It is up to the application running at the group members how to handle possible collisions between Gids, as used for OSCORE groups managed by different, non-synchronized Group Managers.

- The Sender Context includes also the endpoint's private key, unless the endpoint is configured exclusively as silent server. When using the group mode, the private key is used to compute the message signature. When using the pairwise mode, the private key is used to derive a pairwise key between the endpoint and another member of the OSCORE group. It is out of scope for Group OSCORE how the private key has been established.

- Multiple Recipient Contexts are included, i.e., one for each endpoint from which messages are received. No Recipient Contexts are maintained as associated to endpoints from which messages are not (expected to be) received.

The Recipient Context is extended with the public key of the associated endpoint. When using the group mode, the public key is used to verify the message signature. When using the pairwise mode, the public key is used to derive a pairwise key shared with the associated endpoint.

The public key of the associated endpoint and the input parameters for deriving the Recipient Context may be provided to the recipient endpoint upon joining the OSCORE group. Alternatively, these parameters can be acquired at a later time, for example the first time a message is received from this particular associated endpoint in the OSCORE group. The received message, together with the Common Context, includes everything necessary to derive a Security Context for verifying a message, except for the public key of the associated endpoint.

For particularly constrained devices, it can be not feasible to simultaneously handle the ongoing processing of a recently received message and the retrieval of the associated endpoint's public key. Such devices may instead be configured to drop a received message for which there is currently no Recipient Context, and retrieve the public key of the sender endpoint in order to have it available to verify subsequent messages from that endpoint.

Group OSCORE uses the same derivation process of OSCORE to derive Sender Context and Recipient Context - and specifically symmetric Sender/Recipient Keys - from a set of input parameters.

The COSE Object

Compared to OSCORE (see Section 4.2), the following differences apply to the COSE Object.

- When using the group mode, the COSE Object includes an additional digital signature. Its value is set to the countersignature of the encrypted COSE Object, computed by the sender CoAP endpoint by using its own private key and according to the Signature Algorithm specified in the Security Context.

  The literature traditionally considers a countersignature as applied over another signature (i.e., not over any other security structure), and by a principal different from the one that produced what is being countersigned. However, COSE defines a countersignature as applicable also to other security structures. Furthermore, in a group communication scenario and especially building on the design choices of Group OSCORE, it is also appropriate and correct that the same principal as a group member both encrypts a message and then countersigns the result, thus proving to be the actual message sender and ensuring message source authentication.

- The 'kid' parameter is present in all messages, i.e., both requests and responses, specifying the Sender ID of the endpoint transmitting the message. An exception is possible only for response messages, if sent as a reply to a request protected with the pairwise mode.
- The 'kid context' parameter is present in every request message, specifying the Gid value of the group's Security Context. This parameter remains optional to include in response messages.
- The AAD is extended to include additional information, i.e., the algorithms specified in the Common Context; the 'request_kid_context' parameter specifying the Gid used when protecting a request message; and the binary serialization of the OSCORE Option. Like in OSCORE, the AAD is not transmitted, but only takes part in the secure message processing.

Compared to OSCORE (see Section 4.2), the following differences apply to the encoding of a Group OSCORE message.

- When using the group mode, the ciphertext of the COSE Object included as payload of the protected message is further concatenated with the value of the countersignature of the COSE Object.
- In the first byte of the OSCORE option, the sixth least significant bit, namely the Group Flag bit, is used to signal the usage of the group mode. In particular, the Group Flag bit is set to 1 if the message is protected using the group mode. In any other case, including when using the pairwise mode, this bit is set to 0.

5.4 The Group Mode

This section describes how Group OSCORE protects messages in group mode, as main differences from OSCORE (see Section 4.2). In particular, source authentication of messages is achieved by appending a signature to the message payload, computed by using the private key of the message sender. Message confidentiality is achieved at a group level, i.e., every member of the OSCORE group is able to decrypt a message protected in group mode.

Protection of Requests

A CoAP client protects a request in group mode as in OSCORE, with the following differences.

- The extended Additional Authenticated Data (AAD) discussed in Section 5.3 is used for encrypting and signing the request, while the encryption and encoding of the COSE object are as defined in Section 5.3.

- A countersignature of the encrypted COSE object is computed and added at the end of the payload of the protected request message.

Verification of Requests

A server verifies a request protected in group mode as in OSCORE, with the following differences.

- The decoding of the compressed COSE object follows the updates discussed in Section 5.3, while the extended Additional Authenticated Data (AAD) discussed in Section 5.3 is used for decrypting the request and verifying the countersignature of the encrypted COSE object.
- If the received Recipient ID ('kid') does not match with any Recipient Context for the retrieved Gid ('kid context'), then the server may create a new Recipient Context and initialize it at that point in time, also retrieving the client's public key. Such a configuration is application specific. If the application does not specify dynamic derivation of new Recipient Contexts, the server stops processing the request.
- Before decrypting the request, the server verifies the message signature using the public key of the client from the associated Recipient Context. If the verification fails, the server may reply with a 4.00 (Bad Request) response.
- If the used Recipient Context was created upon receiving this group request and the message is not decrypted and verified successfully, the server may delete that Recipient Context. Although application specific, this configuration prevents attacks aimed at overloading the server's storage and creating processing overhead.

Protection of Responses

A server protects a response in group mode as in OSCORE, with the following differences.

- The encoding of the compressed COSE object follows the updates in Section 5.3, while the extended Additional Authenticated Data (AAD) discussed in Section 5.3 is used for encrypting and signing the response.
- A countersignature of the encrypted COSE object is computed and added at the end of the payload of the protected response message.

Since a group rekeying can occur, with consequent re-establishment of the Security Context, the server must always protect a response by using its Sender Context from the latest owned Security Context. As a consequence, right after a group rekeying has been completed, the server may end up protecting a response by using a Security Context different from the one used to protect the group request. In such a case, the server must: i) use its current Sender Sequence Number value to build the nonce for the encryption process; ii) include in the response the 'Partial IV' field of the OSCORE Option, and set it to the Sender Sequence Number above; iii) increment its Sender Sequence Number by one.

Verification of Responses

A client decrypts and verifies a response in group mode as in OSCORE, with the following differences. Note that, as discussed in Section 5.4, a client may receive a response protected with a Security Context different from the one used to protect the corresponding group request.

- The decoding of the compressed COSE object follows the updates in Section 5.3, while the extended Additional Authenticated Data (AAD) discussed in Section 5.3 is used for decrypting the response and verifying its signature.
- If the received Recipient ID ('kid') does not match with any Recipient Context for the retrieved Gid ('kid context'), then the client may create a new Recipient Context and initialize it at that point in time, also retrieving the server's public key. Such a configuration is application specific. If the application does not specify dynamic derivation of new Recipient Contexts, the client stops processing the response.
- Before decrypting the response, the client verifies the signature using the public key of the server from the associated Recipient Context.
- If the used Recipient Context was created upon receiving this response and the message is not decrypted and verified successfully, the client may delete that Recipient Context. Although application specific, this configuration prevents attacks aimed at overloading the client's storage and creating processing overhead.

6 Experimental evaluation

In this section, we present our chosen approach and the experimental setup used to evaluate the performance of the Group OSCORE protocol and to compare it with unicast OSCORE, Group CoAP and unicast CoAP.

6.1 Methodology

In this work, we aim to investigate the performance of Group OSCORE in scenarios where (most) devices are CPU-constrained and most likely battery-powered devices. For example, in a secure firmware/configuration update scenario or a building or lighting control use-case (see Section 3), a single client communicates with multiple servers (sensors, controllers), yielding changes in their state or requesting sensitive data.

Figure 3 depicts the network topology and network protocol stack considered for all our experiments, one constrained client communicate with three constrained servers. No proxies or any other kinds of intermediary nodes were considered in our experimental scenarios.

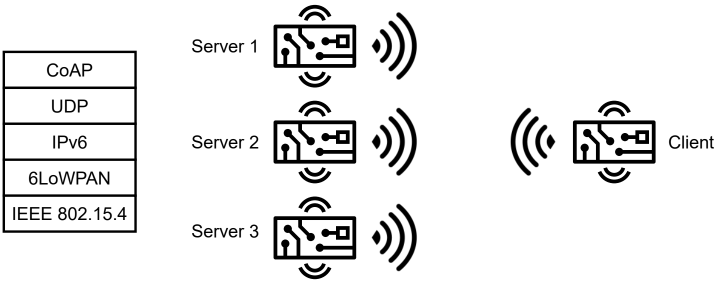


Figure 3: Network topology and network stack for the evaluation experiments.

For our tests, we have tested the four different scenarios defined below.

1. Unprotected CoAP, with no security mechanism applied and all messages sent unicast.

2. Unprotected CoAP with group communication functionality, with no security mechanism applied and with the client sending request messages over multicast and expecting response messages as unicast.

3. Protected CoAP, i.e., using OSCORE to protect all messages, each of which is sent unicast.

4. Protected CoAP with group communication functionality, using Group OSCORE to protect all messages, with the client sending request messages over multicast and expecting response messages as unicast.

In particular, in scenarios (2) and (4), the servers listened to a multicast address for group requests from the client. Furthermore, we evaluated scenarios (3) *twice*, once using security mechanisms implemented entirely in software and once with cryptographic operations executed in a dedicated cryptoprocessor.

Scenario (4) was evaluated *thrice* with different configurations. All messages are protected using the group mode of Group OSCORE, using as Signature Algorithm either ECDSA with curve P-256 or EdDSA with curve Ed25519. We chose to test both algorithms also since the draft of Group OSCORE specifies that either ECDSA P-256 or EdDSA25519 is mandatory to support for constrained devices.

Both the platforms that we have used for our tests have an hardware implementation of ECDSA P-256. Hence, we have tested ECDSA P-256 using both a software implementation and the hardware implementation. Neither platform supports EdDSA with Ed25519 in hardware, which thus was tested only using a software implementation.

6.2 Experimental Setup

All experiments were run with a constrained server of two different hardware platforms, namely Zolertia Firefly Rev. A [Zol20] and TI CC1352R1 Launchpad [Ins21]. We show an overview of the most relevant hardware platform parameters in Table 1. Note that we used the 2.4 GHz frequency band in the experiments.

Table 1: Summary of the hardware platforms used in the experiments.

Platform	MCU	Clock Freq.	RAM	ROM /Flash	Hardware Features	RF Bands
Zolertia Firefly Rev. A (**Zoul**)	ARM Cortex M3	32MHz	32kB (16kB in low-power mode)	512kB	AES128/256, SHA2, ECC128/256	sub-GHz, 2.4 GHz
TI CC1352R1 Simplelink (**Simplelink**)	ARM Cortex M4F	48MHz	80kB + 8kB cache	352kB flash +256kB ROM	AES128/256, SHA2, ECC128/256.	sub-GHz, 2.4 GHz

They are compatible with the Contiki-NG operating system [Con21], our publicly available implementation of Group OSCORE can be found here[5].

We have used version 4.0 of Contiki-NG[6]. The network stack used in these experiments are 2.4 GHz IEEE 802.15.4 LR-WPAN for physical and Medium Access Control (MAC) layer with 6LoWPAN and IPv6 as the network layer. UDP is used at the transport layer. The implementations and settings used for these protocols are the default from the Contiki-NG source tree. Stateless Multicast RPL Forwarding (SMRF) was used for IPv6 multicast communication [OPT13].

The cryptographic functions have been taken from a variety of sources. The software implementation of AES-CCM-128 used in OSCORE is taken from the Contiki-NG source-tree. The SHA256 software implementation used in both OSCORE and Group OSCORE is taken from Tinydtls[7]. The hardware implementation of AES-CCM-128, SHA256 and ECDSA P-256 for the Zoul platform is also taken from the Contiki-NG source-tree. The same functions for the Simplelink platform was taken from the Simplelink SDK provided by Texas Instruments. The software implementations of asymmetric cryptography used is uECC (micro-ecc)

[5] `https://github.com/Gunzter/Contiki-ng/tree/group_oscore` — Accessed: 2022-01-21

[6] `https://github.com/contiki-ng/contiki-ng/releases/tag/release%2Fv4.0`

[7] `https://projects.eclipse.org/projects/iot.tinydtls`

for ECDSA P-256[8] and Monocypher for EdDSA25519[9]. These were chosen because they are written in portable C and have permissible licenses. Both libraries have shown good performance in previous evaluations [Zan+19].

The tested software configurations can be found below:

1. In the *CoAP* case, no security schemes were used.
2. In the *Group-CoAP* case, no security schemes were used.
3. In the *OSCORE-SW* case, relevant cryptographic operations were performed on the primary CPU.
4. In the *OSCORE-HW* case, the cryptographic operations were performed on the dedicated crypto accelerators of the IoT platforms.
5. In the *Group-OSCORE-SW* case, the cryptographic operations were performed on the primary CPU. The asymmetric cryptography algorithm used is ECDSA, with curve P-256.
6. In the *Group-OSCORE-HW* case, the cryptographic operations were performed on the dedicated crypto accelerators of the IoT platforms.
7. In the *Group-OSCORE-EDDSA* case, the cryptographic operations were performed on the primary CPU. The asymmetric cryptography algorithm used is EdDSA, with curve Ed25519.

6.3 Experimental Scenarios

For each of the hardware and software configurations, **14 in total**, the following experiments were conducted:

1. Memory occupancy (RAM and ROM) on the server.
2. Time elapsed on the client from the start of request processing to the end of response processing, or RTT.
3. Time spent by the CPU on the server, to process incoming/outgoing messages.
4. Energy consumption on the server, specific to message processing.

The measurement of memory occupancy (1) relied on the fact that Contiki-NG only dynamically allocates stack memory, while the heap is a static memory block. We determined heap usage with the GNU command *size*, while we measured the stack utilization with *stack painting*. We *painted the stack* by writing

[8] `http://kmackay.ca/micro-ecc/` — Accessed: 2022-01-21

[9] `https://monocypher.org/`

a known value to all the bytes of the stack at device boot, then we ran the experiments. After we finished the experiments, we counted the bytes with values different from the initial values.

We measured the Round Trip Time (RTT) experienced by the client, as the time interval from the start of the processing of an outgoing request until the completion of the processing of the received response in (2).

As for the CPU time for message processing (3), the measured time interval included all the cryptographic operations considered by the used security protocol, i.e., encryption/decryption, integrity verification, and message signing/verifying.

In order to obtain the energy consumption measurements of message processing (5), we used a DC energy analyzer to measure the voltage and current supplied to the tested device (see Section 7.4).

During the RTT, CPU and energy consumption tests, a constrained Contiki-NG client generated and sent requests of varying payload sizes (1 to 128 bytes). This number (1 to 128) indicates the CoAP payload size, before any protocol overhead. The data transmitted was dummy bytes, that were changed on the server to indicate that the responses had actually passed the server. These data were transmitted by the client and received by the servers, which in turn responded with messages transmitted to the client.

7 Results and discussion

In this section, we present the results of our experiments. We start with memory utilization on the servers, comparing the RAM and ROM requirements in the different scenarios. Then, we present the Round Trip Time (RTT) on the observed client, followed by the overall processing time and the relative impact due to the performed cryptographic operations on the server-side. Finally, we provide the measured energy consumption on the server-side in the different scenarios as a vital element to consider for constrained devices.

7.1 Memory utilization

We calculated the memory utilization by combining data from compiled ELF files with stack usage numbers from run-time. We used the GNU tool *objdump* [10] to extract the size of the .Text, .BSS, and .Data segments from the ELF files. Furthermore, we wrote a known bit pattern to the stack at system boot, ran the experiments, and afterwards, we counted the number of bytes on the stack that the tested program had overwritten. We then calculated the RAM utilization as the size of .Data + .BSS + stack, while the ROM utilization was calculated as the size of .BSS + .Text.

Figures 4 and 5 present RAM and ROM utilization, additionally highlighting the stack usage and the memory overhead corresponding to the cryptographic

[10] `https://man7.org/linux/man-pages/man1/objdump.1.html` — Accessed: 2022-01-21

operations. Noteworthy, the use of the stack depends on the message size, while implementation issues influences the code size for cryptography. Regarding the scenarios defined in Section 6.1 and summarized in Table 3, the bars shown in the figures use the following labels: *COAP* for scenario (1); *Group-COAP* for scenario (2); *OSCORE-SW* for scenario (3) and *OSCORE-HW* for scenario (4), with cryptographic operations performed in software or with hardware acceleration, respectively; and *Group-OSCORE-SW* for scenario (5), *Group-OSCORE-HW* for scenario (6), with cryptographic operations performed in software or with hardware acceleration, respectively. Finally, *Group-OSCORE-EDDSA* for scenario (7) covers also the use of the EdDSA signature algorithm in software.

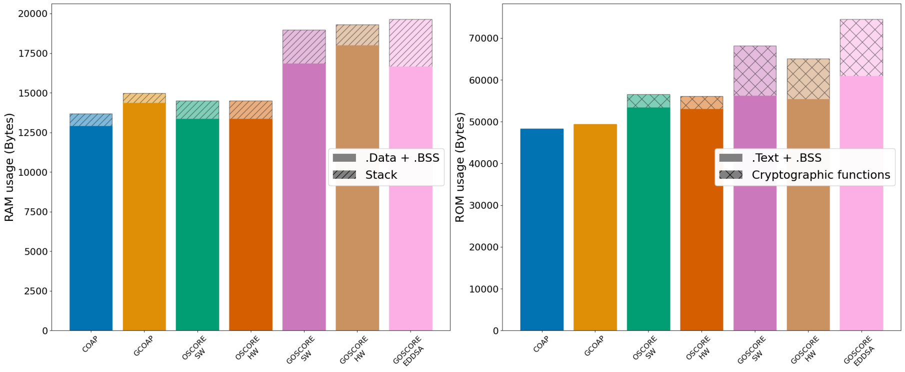


Figure 4: Memory utilization for the Zoul platform, out of 32KB of RAM and 512KB of ROM available.

Figure 4 shows the memory utilization for the Zoul platform. The bars represent memory utilization for the whole system i.e., the Contiki-NG operating system, network stack, default libraries, drivers and a CoAP server possibly including OSCORE or Group-OSCORE.

The slight increase in both RAM and ROM for *Group-COAP* compared to *COAP* is caused by the inclusion of IPv6 Multicast routing functionality. Additional routines for processing *OSCORE* messages add 2 KB of extra ROM and another 1 KB for the AES-CCM-128 and SHA256 algorithms. Although hardware-accelerated cryptography used in *Group-OSCORE* requires less ROM than the cryptographic software libraries, it requires more RAM. The code used to interface with the hardware accelerator on the Zoul platform implements ECDSA P-256 without any optimizations. This negatively impacts performance in terms of memory utilization. The software library used for ECDSA P-256, micro-ecc, is optimized to conserve memory. This explains how the hardware implementation of ECDSA P-256 surprisingly utilize more memory. Finally, in the case of *Group-OSCORE-EDDSA* we observe a larger RAM and ROM utilization that we attribute to the used cryptographic library, which has been shown to require significant memory resources [Zan+19].

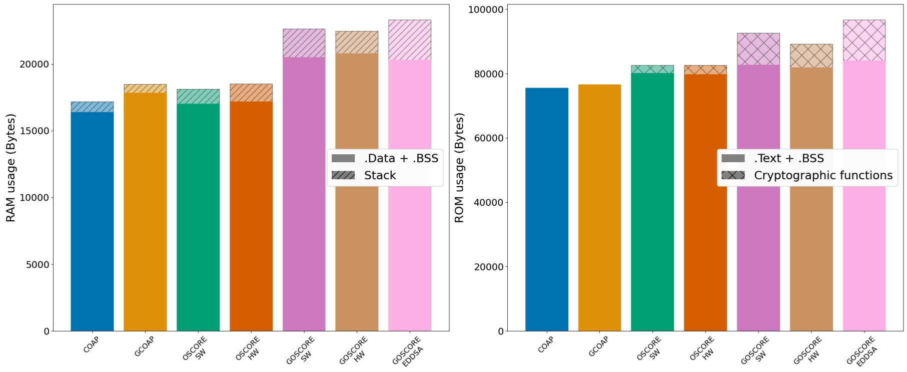


Figure 5: Memory utilization for the Simplelink platform, out of 88KB of RAM and 608KB of ROM available.

Figure 5 shows the memory utilization for the Simplelink platform. Note that the comparison between software and hardware-accelerated cryptography solutions looks slightly different than in the case of the Zoul platform (see Figure 4), which results from hardware discrepancies and the nuances of particular device driver implementations.

Considering that the Zoul platform has 32 KB of RAM and 512 KB of ROM, the increased amount of memory needed for *Group-OSCORE* processing seems reasonable. We can conclude that fitting Contiki-NG with a network stack and our *Group-OSCORE* implementation in addition to application code is feasible on the Zoul platform. Moreover, low-power operation, resulting in only half of Zoul RAM memory available, could be achieved with *Group-OSCORE* on board. The Simplelink platform has even more memory, i.e., 88 KB of RAM and 608 KB of ROM. Thus, the increased memory utilization due to *Group-OSCORE* is not significant in terms of the total memory.

7.2 Message Round Trip Time (RTT)

We measured the RTT as the time elapsed from the start of processing the outgoing request in the constrained client until the moment when the received response has been processed by the client. In particular, we considered different sizes for the application payload, excluding any message overhead from headers, AEAD-encryption tags, or signatures.

Because of the response delay randomization required by CoAP group communication (see Section 4.1), we considered both the time elapsed until the first response arrived and the time elapsed until the last response arrived. As used together with group communication, Group-OSCORE also relies on the response delay randomization mechanism. When evaluating the test cases relying on group communication, the servers chose the delay of each response as a random value between 0 and 8 s, in order to conservatively prevent collisions and ensure a feasible collection of results. It follows that, on average, a large share of RTT values when group communication is used consists of the added randomized delay.

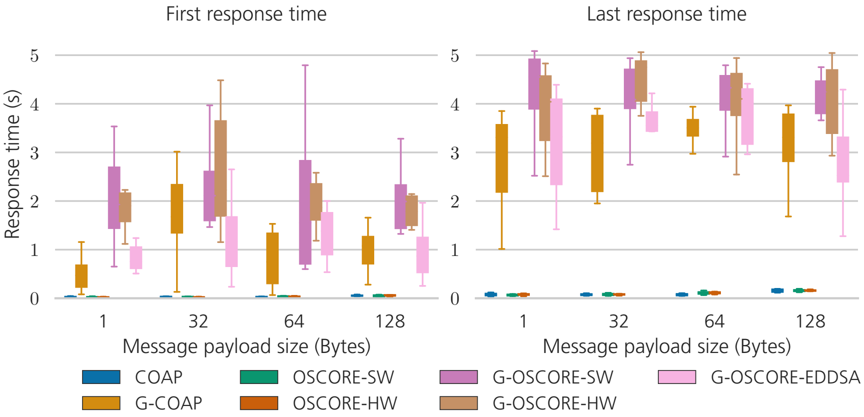


Figure 6: Round Trip Time (RTT) measurements for the Zoul platform.

Figure 6 shows the RTT measured when considering the Zoul platform, where we can see two sets of response times. The first set comprises response times for the three test cases *COAP*, *OSCORE-SW* and *OSCORE-HW*. Consistently with a quick response delivery, they are all low for both the first and the last response, with no notable influence from the considered message payload sizes.

The second set comprises response times for the four test cases relying on group communication, i.e, *Group-COAP*, *Group-OSCORE-SW*, *Group-OSCORE-HW* and *Group-OSCORE-EDDSA*. In these cases, the response times are considerably greater and fall within a large range of possible values. This is mainly due to the random delay necessitated when using group communication for CoAP, as introduced by each responding server (see Section 4.1). The extent of such an impact may largely vary on a per-response basis, depending on the exact delay randomly selected and introduced by each server. Also, its contribution to the overall response time might be negligible, or rather comparable to or bigger than that from possible cryptographic operations, again on a per-message basis. Thus, the contribution of security operations is later analyzed in Section 7.3, when discussing CPU utilization. At the same time, the variability introduced by the random delays overshadows the possible role played by the message payload sizes. More in general, the range of values where the random delay is selected from can be determined according to the characteristics of the system and network deployment, hence its width could be limited in particular settings, in turn limiting the variability of response times.

Figure 7 shows the RTT measured for the Simplelink platform. We can make the considerations discussed above for the Zoul platform and the related results

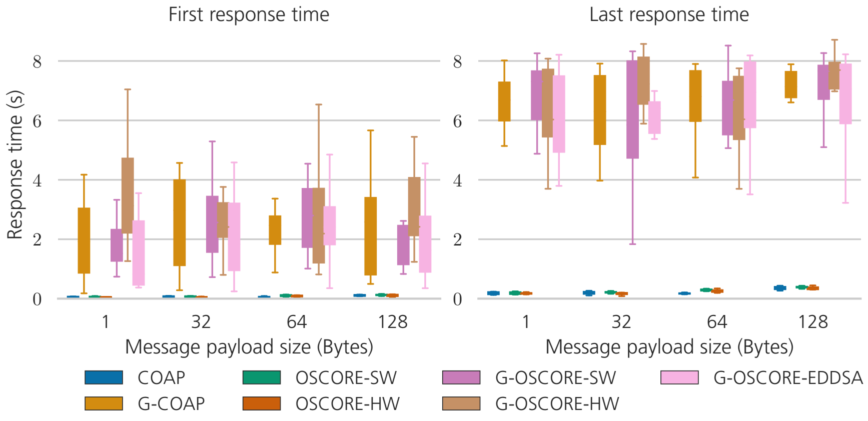


Figure 7: Round Trip Time (RTT) measurements for the Simplelink platform.

shown in Figure 6. In particular, the three purely-unicast test cases display small, consistent and similar response times, while the four test cases relying on group communication display a larger response time with a large range of values due to the random delays introduced by the servers.

7.3 CPU utilization

To evaluate the impact of the cryptographic operations on the overall message processing, we have also measured the time required to perform such functions and compared it against the total message processing time. For *OSCORE*, we have measured the time needed to encrypt (decrypt) a message and presented the results as the percentage of the total time required to serialize an outgoing (parse an incoming) message. For *Group-OSCORE*, we have measured the time needed to encrypt and sign (or decrypt and verify the signature of) an outgoing (incoming) message and presented the results as the percentage of the total time needed to serialize an outgoing (parse an incoming) message.

Table 2 shows the processing times for the Zoul platform in milliseconds. The table reveals that hardware acceleration for cryptographic operations improves *OSCORE* performance, as hardware accelerated cryptography takes up a smaller fraction of the total processing time, i.e., less than 50%. For *Group-OSCORE*, cryptographic functions are responsible for the absolute majority of time taken to serialize and parse messages. Particularly, over 99% of the processing time is spent for producing or verifying the signature of an outgoing or incoming message, respectively. Note that using hardware acceleration for cryptographic operations reduces the overall time needed to *serialize* messages. Instead, signature verification and the total message parsing time do not show the same performance improvement when hardware acceleration for cryptographic operations is applied. The reason is the relative efficiency displayed by the software implementation of ECDSA P-256. The ECDSA software implementation used in the *Group-OSCORE-SW* code was µECC [11], which has outstanding performance compared to both other ECDSA software implementations [Mös+16]. This performance has been achieved through careful optimizations of Elliptic-Curve operations in Assembly and the use of Shamir's Trick [Sma16], which reduces the number of Elliptic-Curve scalar multiplications needed to verify a signature from two to one. In particular, relying on Shamir's Trick reduces the time to verify a signature by approximately 40%.

EdDSA25519 shows even better performance and beats both the software and hardware implementation of ECDSA P-256. This implementation shows consistently superior performance for both signing and verifying messages.

Table 2: CPU usage of the Zoul platform in milliseconds, for serializing (S) and parsing (P) of messages with cryptographic operations as a percentage of total.

Payload (Bytes)	1		32		64		128	
	S	P	S	P	S	P	S	P
COAP	0.05	0.12	0.07	0.12	0.09	0.12	0.11	0.12
Group-COAP	0.05	0.12	0.05	0.12	0.06	0.12	0.06	0.12
OSCORE-SW	0.51	0.56	0.71	0.76	0.92	0.95	1.35	1.34
Encypt/Decrypt	54.2%	53.0%	65.0%	64.3%	72.1%	71.1%	77.1%	79.2%
OSCORE-HW	0.32	0.38	0.35	0.39	0.38	0.40	0.45	0.44
Encypt/Decrypt	26.7%	27.6%	29.3%	28.1%	28.8%	31.8%	30.4%	34.5%
G-OSCORE-SW	576.97	624.99	577.68	626.98	577.65	625.98	578.36	627.65
Encypt/Decrypt	99.9%	99.9%	99.8%	99.9%	99.9%	99.9%	99.9%	99.9%
G-OSCORE-HW	348.84	706.55	349.08	711.96	349.29	710.74	349.74	707.45
Encypt/Decrypt	99.9%	99.9%	99.8%	99.9%	99.8%	99.9%	99.8%	99.9%
G-OSCORE-EDDSA	107.50	269.03	109.13	268.09	109.81	269.15	111.88	271.39
Encypt/Decrypt	99.2%	99.8%	99.4%	99.8%	99.4%	99.8%	99.3%	99.8%

We present the results for the Simplelink platform in Table 3. The reported values follow the same trends as in Table 2, but they are generally lower compared to the values for the Zoul platform. One notable exception is *OSCORE-SW*, where the times are longer in the Simplelink case. This is due to the less efficient AES-CCM Simplelink implementation, rendered by the increased delay share of the *OSCORE-SW* crypto functions in Table 3.

The processing time results for Zoul and Simplelink platforms show similar trends, albeit shorter total times for the Simplelink platform can be observed. *OSCORE* cryptography takes between 25% and 90% when implemented in software and between 10% and 55% when implemented in hardware. This is a significant part of the processing time, but it is noticeably smaller compared to the 99% of the total time that cryptography takes up when *Group-OSCORE* is used. As learned from the *OSCORE* experiments, 1-2 ms were needed to encrypt and decrypt a message, while the further increase in the processing delay is due to verifying a signature or generating one. Hardware acceleration shortens the total processing time for *Group-OSCORE*. Ed25519 using the Monocypher library shows excellent

[11] `https://github.com/kmackay/micro-ecc` — Accessed: 2022-01-21

Table 3: CPU usage of the SimpleLink platform in milliseconds, for serializing (S) and parsing (P) of messages with cryptographic operations as a percentage of total.

Payload (Bytes)	1		32		64		128	
	S	P	S	P	S	P	S	P
COAP	0.00	0.18	0.06	0.16	0.06	0.16	0.06	0.17
Group-COAP	0.03	0.20	0.05	0.20	0.06	0.20	0.07	0.21
OSCORE-SW	1.85	1.98	2.93	3.06	4.03	4.14	6.18	6.28
Encypt/Decrypt	90.8%	85.2%	93.8%	90.0%	94.5%	92.2%	96.2%	94.8%
OSCORE-HW	0.31	0.46	0.37	0.49	0.42	0.51	0.61	0.68
Encypt/Decrypt	41.2%	34.2%	50.0%	37.5%	43.5%	36.9%	51.0%	49.5%
G-OSCORE-SW	296.91	317.10	298.40	319.06	299.23	319.76	301.76	322.94
Encypt/Decrypt	99.8%	99.8%	99.8%	99.8%	99.8%	99.8%	99.8%	99.8%
G-OSCORE-HW	233.89	467.10	233.95	466.39	234.01	466.50	234.25	466.08
Encypt/Decrypt	99.8%	99.9%	99.8%	99.9%	99.7%	99.9%	99.7%	99.9%
G-OSCORE-EDDSA	50.58	112.17	52.63	112.92	53.92	114.14	57.23	117.58
Encypt/Decrypt	99.1%	99.5%	99.0%	99.4%	98.9%	99.4%	98.8%	99.4%

performance in terms of speed on both platforms, outperforming ECDSA P-256 in both hardware and software on both platforms. This speed comes at the penalty of a larger memory footprint in ROM and the largest utilization of RAM of all the tested asymmetric cryptography implementations on both platforms. This leads to a remark that asymmetric cryptography is more time-consuming compared to symmetric cryptography, when measured on these types of constrained devices.

7.4 Per-message energy consumption

For resource-constrained devices, energy consumption is an important metric to assess when evaluating protocol performance. In this paper, we have measured the energy consumption by using the following method.

We have used a DC energy analyzer, an instrument that measures voltage and current with a high precision at a very high sampling frequency. In particular, we used a Joulescope as the DC energy analyzer to measure the power consumption of the tested software configurations on both hardware platforms. Power was supplied to the device by a fixed voltage from a small power supply. The Joulescope measures voltage and current supplied to the device, as evaluated at 2 million samples per second with nanoampere precision[12].

We opted to measure the actual energy consumed during message processing. As in the case of CPU-time measurements, we considered the period between the start of application-layer processing of the incoming message until the message is delivered to the application. We also measured the total time spent by the application-layer to fully prepare and process an outgoing message, including possible security operations (see Section 7.3).

We did our experiments with 10 message exchanges per payload size, since the times measured in Section 7.3 had small deviations from the mean.

[12] `https://www.joulescope.com/` — Accessed: 2022-01-21

Figure 8 shows the measured energy consumption for the Zoul platform. Note that the Y-scale is logarithmic since it was impossible to adequately show the large difference in energy consumption for both the *Group-OSCORE* and *COAP* cases in the same plot using a linear scale.

Looking at the bars for *OSCORE-SW*, *OSCORE-HW*, *Group-OSCORE-SW* and *Group-OSCORE-HW*, it becomes clear that the per-message energy consumption increases with growing message sizes for *OSCORE-SW* and *OSCORE-HW*, but remains virtually constant for *COAP*, *Group-COAP* and the three *Group-OSCORE* variants. This can be explained by the time needed to generate and verify ECC signatures, which is almost constant, as discussed in Section 7.2. Furthermore, the energy consumption for *Group-OSCORE-EDDSA* is lower than that for the *Group-OSCORE-SW* and *Group-OSCORE-HW* results. This is due to the efficient (i.e., fast) cryptographic library for EdDSA signing used. When it comes to *COAP* and *Group-COAP*, the roughly constant energy consumption can be explained by the very short CPU time observed in Section 7.3. For *OSCORE-SW* and *OSCORE-HW*, a linear growth trend can be observed, because of the increased processing time needed to encrypt the payload of the messages.

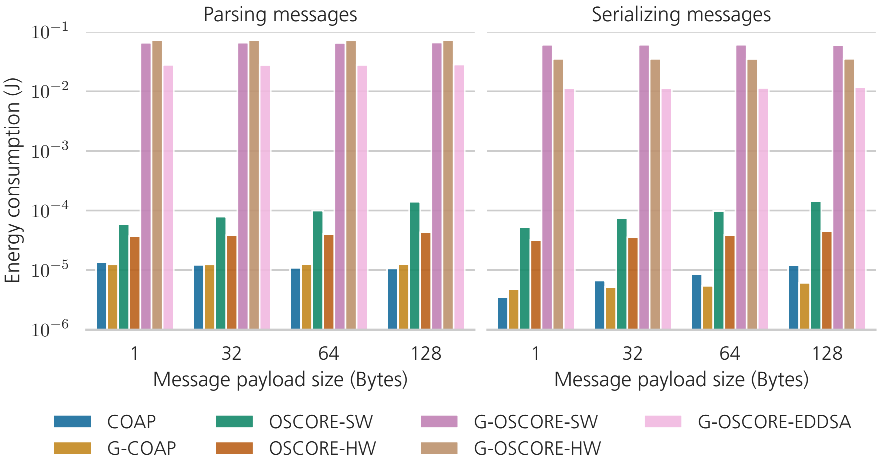


Figure 8: Energy measurements for the Zoul platform.

In Figure 9, we show the results for the Simplelink platform. As for the previous case, one can see that where *COAP*, *Group-COAP*, *Group-OSCORE-SW*, *Group-OSCORE-HW* and *Group-OSCORE-EDDSA* show a roughly constant energy consumption across message payload sizes. Instead, *OSCORE-SW* and *OSCORE-HW* show a linear relation between message payload size and energy consumption. This is due to the encryption and decryption of the message payload taking more CPU-time, and thus energy, to process a larger payload.

Among the three test cases with Group OSCORE, *Group-OSCORE-SW* displays the highest energy consumption for both parsing and serializing messages,

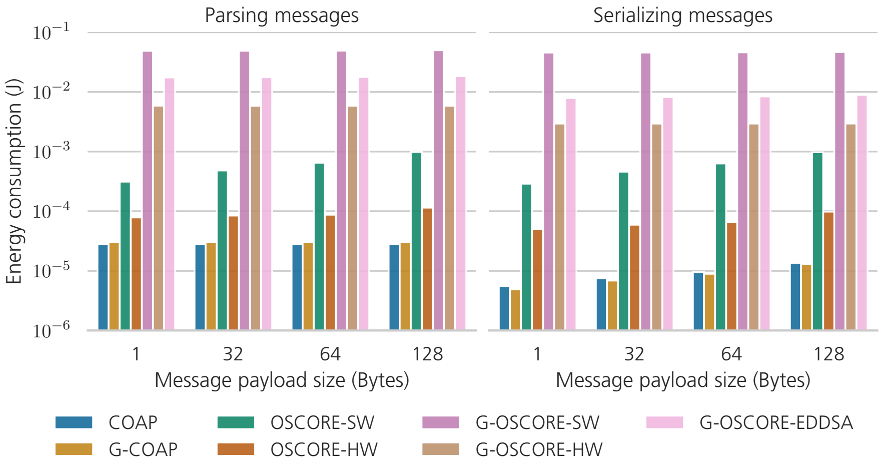


Figure 9: Energy measurements for the Simplelink platform.

while *Group-OSCORE-HW* and *Group-OSCORE-EDDSA* display the lowest energy consumption and a value in between, respectively.

For the Zoul platform, the per-exchange energy consumption is 0.3 mJ for *OSCORE-SW*, 125 mJ for *Group-OSCORE-SW*, 107 mJ for *Group-OSCORE-HW*, and 39 mJ for *Group-OSCORE-EDDSA*. Using *OSCORE-SW* as a baseline, *Group-OSCORE-SW* consumes 416 times more energy per message exchange. For *Group-OSCORE-HW*, the corresponding number is 356 times the energy of *OSCORE-SW*. Lastly *Group-OSCORE-EDDSA* consumes 130 times more energy per message exchange.

For the Simplelink platform, the processing energy consumption per exchange is 2 mJ for *OSCORE-SW*, 96 mJ for *Group-OSCORE-SW*, 9 mJ for *Group-OSCORE-HW*, and 27 mJ for *Group-OSCORE-EDDSA*. This results in a 48 times increase in energy consumption for *Group-OSCORE-SW*, a 4.5 times increase for *Group-OSCORE-HW*, and a 13.5 times increase for *Group-OSCORE-EDDSA*.

Considering a 5000 mAh battery, operating at 3.3 V (equivalent to 16.5 Wh or 59400 J of energy), one can realise that it becomes possible to perform the following amount of message exchanges:

- Zoul platform: 198 million *OSCORE-SW* transactions, 475 thousand *Group-OSCORE-SW* transactions and 555 thousand *Group-OSCORE-HW* transactions.

- Simplelink platform: 30 million *OSCORE-SW* transactions, 619 thousand *Group-OSCORE-SW* transactions, 6.6 million *Group-OSCORE-HW* transactions, and 2.2 million *Group-OSCORE-EDDSA* transactions.

As a matter of fact, the above calculations are rough and do not take into account physical phenomena (e.g., leakage currents or battery degradation) as well as application overhead, which occur when a device is deployed in a real-life setting. Nevertheless, the amount of possible client-server transactions in Group-OSCORE variants is in our opinion sufficiently high to claim that this protocol can be applied in low-power, low-data IoT applications.

8 Conclusions

In this paper, we have presented and experimentally evaluated the Group-OSCORE security protocol. This can be used to protect CoAP messages end-to-end at the application level in group communication scenarios, where a CoAP client sends a request intended to multiple CoAP servers, e.g., over IP multicast. In particular, source authentication of messages is achieved by means of digital signatures embedded in the protected payload.

To best of our knowledge, this work is the first publicly available Group OSCORE performance evaluation using an implementation on real, constrained IoT hardware. For our study, we developed a Contiki-NG implementation (published as open-source) and tested the protocol on two platforms: Zolertia Firefly Rev. A CC2538 and TI Simplelink CC1352R1 Launchpad. Furthermore we considered multiple signing algorithms, implementing and evaluating the Group OSCORE solution both for EdDSA and ECDSA P-256.

Our experiments encompassed memory consumption, RTT and energy consumption overhead of the novel protocol. The results showed that incorporating Group OSCORE does not introduce significant RAM/ROM penalty on the two tested platforms. On the other hand, Group OSCORE operations (most notably message signing and signature verification) contribute to a considerably larger RTT, when compared to CoAP, Group CoAP and OSCORE. Group OSCORE also consumes more energy per message sent if compared to OSCORE, which might be a limiting factor for constrained, battery powered IoT nodes communicating frequently. Nevertheless, we have showed that the number of possible message exchanges is high, also for battery-powered devices. Finally we observed that using the EdDSA signature algorithm resulted in better performance than ECDSA P-256, especially when considering the ECDSA P-256 software implementation.

We believe that, in order to fully understand the potential and limitations of Group OSCORE, more trials are appropriate to be conducted in the future. Specifically, a follow-up evaluation can take into account additional hardware platforms. Furthermore, it can also consider the pairwise mode, which uses only symmetric encryption to protect the unicast responses. This is expected to substantially reduce the computing time and the corresponding energy consumption, as well as the Round Trip Time experienced by a client.

Acknowledgements

The authors sincerely thank the anonymous reviewers and the Editor-in-Chief for their insightful comments and suggestions, that have helped to improve the technical and editorial quality of the manuscript. This work was partially funded by Innovation Fund Denmark through Eureka Turbo project IoT Watch4Life and Nordic University Hub for Industrial IoT (HI2OT); the SSF project SEC4Factory under the grant RIT17-0032; VINNOVA through the Celtic-Next project CRITISEC; and the H2020 project SIFIS-Home (Grant agreement 952652). The authors would like to thank Matthew Bradbury for his valuable help with the implementation and Pegah Nikbakht Bideh for her valuable help with the energy measurement equipment.

References

[All20a] O. M. Alliance. *Lightweight Machine to Machine Technical Specification: Core - Approved Specification 1.2*. `http://www.openmobilealliance.org/release/LightweightM2M/V1_2-20201110-A/OMA-TS-LightweightM2M_Core-V1_2-20201110-A.pdf`. Accessed: 2022-01-21. Nov. 2020.

[All20b] O. M. Alliance. *Lightweight Machine to Machine Technical Specification: Transport Bindings - Approved Specification 1.2*. `http://www.openmobilealliance.org/release/LightweightM2M/V1_2-20201110-A/OMA-TS-LightweightM2M_Transport-V1_2-20201110-A.pdf`. Accessed: 2022-01-21. Nov. 2020.

[AT21] C. Amsüss and M. Tiloca. *Cacheable OSCORE*. Internet-Draft draft-amsuess-core-cachable-oscore. (Work in progress). Internet Engineering Task Force, 2021.

[Bae+13] J. Baek et al. "An Authentication Framework for Automatic Dependent Surveillance-Broadcast Based on Online/Offline Identity-Based Signature". In: *2013 Eighth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PG-CIC)*. USA: IEEE Computer Society, 2013, pp. 358–363.

[Ban+17] A. Banks et al. *OASIS Standard MQTT Version 5.0*. Tech. rep. Accessed: 2022-01-21. OASIS, Mar. 2017.

[BH20] C. Bormann and P. E. Hoffman. *Concise Binary Object Representation (CBOR)*. RFC 8949. Dec. 2020.

[Bor+18] C. Bormann et al. *CoAP (Constrained Application Protocol) over TCP, TLS, and WebSockets*. RFC 8323. Feb. 2018.

[Bri10] B. Briscoe. *Tunnelling of Explicit Congestion Notification*. RFC 6040. Nov. 2010.

[CLC20] H.-Y. Chien, P.-C. Lin, and M.-L. Chiang. "Efficient MQTT Platform Facilitating Secure Group Communication". In: *Journal of Internet Technology* 21 (7 Dec. 2020), pp. 1929–1940.

[CNN20] B. Choudhury, A. Nag, and S. Nandi. "DTLS based secure group communication scheme for Internet of Things". In: *2020 IEEE 17th International Conference on Mobile Ad Hoc and Sensor Systems (MASS 2020)*. New York, NY, USA: IEEE, 2020, pp. 156–164.

[Con21] Contiki-NG. *Contiki-NG: The OS for Next Generation IoT Device*. `https://www.contiki-ng.org/`. Accessed: 2022-01-21. 2021.

[DDT11] R. Daidone, G. Dini, and M. Tiloca. "On Experimentally Evaluating the Impact of Security on IEEE 802.15.4 Networks". In: *In Proceedings of International Conference on Distributed Computing in Sensor Systems and Workshops, (PWSN 2011)*. Barcelona, Spain: IEEE Computer Society, June 2011, pp. 20–25.

[DS11] G. Dini and I. M. Savino. "LARK: A Lightweight Authenticated ReKeying Scheme for Clustered Wireless Sensor Networks". In: *ACM Transactions on Embedded Computing Systems* 10.4 (Nov. 2011), 41:1–41:35.

[DWT21] E. Dijk, C. Wang, and M. Tiloca. *Group Communication for the Constrained Application Protocol (CoAP)*. Internet-Draft draft-ietf-core-groupcomm-bis. (Work in progress). Internet Engineering Task Force, 2021.

[Eri19] Ericsson. *OSCORE: A look at the new IoT security protocol*. `https://www.ericsson.com/en/blog/2019/11/oscore-iot-security-protocol`. Accessed: 2022-01-21. Nov. 2019.

[Fie00a] R. Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. `https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm`. Accessed: 2022-01-21. Irvine, California, USA, 2000.

[FR14] R. Fielding and J. Reschke. *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content*. RFC 7231. June 2014.

[GFK17] T. Guggemos, N. G. Felde, and D. Kranzlmüller. "Secure group communication in constrained networks—A gap analysis". In: *2017 Global Internet of Things Summit (GIoTS)*. IEEE. 2017, pp. 1–4.

[Gun+21] M. Gunnarsson et al. "Evaluating the performance of the OSCORE security protocol in constrained IoT environments". In: *Internet of Things* 13 (Mar. 2021). Accessed: 2022-01-21, p. 100333.

[Gün+21] C. Gündoğan et al. "Group Communication with OSCORE: RESTful Multiparty Access to a Data-Centric Web of Things". In: *2021 IEEE 46th Conference on Local Computer Networks (LCN 2021)*. New York, NY, USA: IEEE, Oct. 2021, pp. 399–402.

[GWI08] G. Gross, B. Weis, and D. Ignjatic. *Multicast Extensions to the Security Architecture for the Internet Protocol*. RFC 5374. Nov. 2008.

[Har15] K. Hartke. *Observing Resources in the Constrained Application Protocol (CoAP)*. RFC 7641. Fremont, CA, USA: RFC Editor, Sept. 2015, pp. 1–30.

[IDC20] IDC. *IoT Growth Demands Rethink of Long-Term Storage Strategies, says IDC*. `https://www.idc.com/getdoc.jsp?containerId=prAP46737220`. Accessed: 2022-01-21. July 2020.

[IEE20] IEEE. *IEEE 802.15.4-2020 - IEEE Standard for Low-Rate Wireless Networks*. IEEE. July 2020.

[Ins21] T. Instruments. *LAUNCHXL-CC1352R1 SimpleLink™ Multi-Band CC1352R Wireless MCU LaunchPad™ Development Kit*. `https://www.ti.com/tool/LAUNCHXL-CC1352R1`. Accessed: 2022-01-21. 2021.

[Jac+09] V. Jacobson et al. "Networking named content". In: *Proceedings of the 5th international conference on Emerging networking experiments and technologies*. 2009, pp. 1–12.

[Kau+14] C. Kaufman et al. *Internet Key Exchange Protocol Version 2 (IKEv2)*. RFC 7296. Oct. 2014.

[Kiv16] T. Kivinen. *Minimal Internet Key Exchange Version 2 (IKEv2) Initiator Implementation*. RFC 7815. Mar. 2016.

[KKJ19] M. Koster, A. Keränen, and J. Jiménez. *Publish-Subscribe Broker for the Constrained Application Protocol (CoAP)*. Internet-Draft draft-ietf-core-coap-pubsub. (Work in progress). Internet Engineering Task Force, 2019.

[Liz+21] A. Lizardo et al. "End-to-end secure group communication for the Internet of Things". In: *Journal of Information Security and Applications* 58 (May 2021), p. 102772.

[Mig+17] D. Migault et al. "Diet-ESP: IP layer security for IoT". In: *Journal of Computer Security* 25 (2 May 2017), pp. 173–203.

[Mös+16] M. Mössinger et al. “Towards quantifying the cost of a secure IoT: Overhead and energy consumption of ECC signatures on an ARM-based device”. In: *2016 IEEE 17th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM 2016)*. New York, NY, USA: IEEE, June 2016, pp. 1–6.

[OPT13] G. Oikonomou, I. Phillips, and T. Tryfonas. “IPv6 multicast forwarding in RPL-based wireless sensor networks”. In: *Wireless personal communications* 73.3 (2013), pp. 1089–1116.

[Per+00] A. Perrig et al. “Efficient authentication and signing of multicast streams over lossy channels”. In: *Proceeding 2000 IEEE Symposium on Security and Privacy. S P 2000*. New York, NY, USA: IEEE, May 2000, pp. 56–73.

[Per+02] A. Perrig et al. “SPINS: Security Protocols for Sensor Networks”. In: *Wireless Networks* 8 (5 Sept. 2002), pp. 521–534.

[Pol18] P. Polak. *Security Architecture for the Internet of Things (IoT) in Commercial Buildings*. Accessed: 2022-01-21. Mar. 2018.

[Pos80] J. Postel. *User Datagram Protocol*. RFC 768. Fremont, CA, USA: RFC Editor, Aug. 1980, pp. 1–3.

[Pos81] J. Postel. *Transmission Control Protocol*. RFC 793. Fremont, CA, USA: RFC Editor, Sept. 1981, pp. 1–91.

[Raz+11] S. Raza et al. “Securing communication in 6LoWPAN with compressed IPsec”. In: *2011 International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS)*. IEEE. 2011, pp. 1–8.

[RD14] A. Rahman and E. Dijk. *Group Communication for the Constrained Application Protocol (CoAP)*. RFC 7390. Oct. 2014.

[Res18] E. Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.3*. RFC 8446. RFC Editor, Aug. 2018.

[Riz+19] K. Rizki et al. “Group-IKEv2 for multicast IPsec in the internet of things”. In: *International Journal of Security and Networks* 14.1 (Apr. 2019), pp. 10–22.

[RM12] E. Rescorla and N. Modadugu. *Datagram Transport Layer Security Version 1.2*. RFC 6347. Jan. 2012.

[Sch20] J. Schaad. *CBOR Object Signing and Encryption (COSE): Initial Algorithms*. Internet-Draft draft-ietf-cose-rfc8152bis-algs. (Work in progress). Internet Engineering Task Force, 2020.

[Sch21] J. Schaad. *CBOR Object Signing and Encryption (COSE): Structures and Process*. Internet-Draft draft-ietf-cose-rfc8152bis-struct. (Work in progress). Internet Engineering Task Force, 2021.

[Sei+21] L. Seitz et al. *Authentication and Authorization for Constrained Environments (ACE) using the OAuth 2.0 Framework (ACE-OAuth)*. Internet-Draft draft-ietf-ace-oauth-authz. (Work in progress). Internet Engineering Task Force, 2021.

[Sel+19a] G. Selander et al. *Object Security for Constrained RESTful Environments (OSCORE)*. RFC 8613. RFC Editor, July 2019.

[SHB14] Z. Shelby, K. Hartke, and C. Bormann. *The Constrained Application Protocol (CoAP)*. RFC 7252. June 2014.

[SK05] K. Seo and S. Kent. *Security Architecture for the Internet Protocol*. RFC 4301. Dec. 2005.

[Sma16] N. P. Smart. *Cryptography made simple*. London, UK: Springer Nature, 2016.

[SMP21] G. Selander, J. Mattsson, and F. Palombini. *Ephemeral Diffie-Hellman Over COSE (EDHOC)*. Internet-Draft draft-ietf-lake-edhoc. (Work in progress). Internet Engineering Task Force, 2021.

[SPH17] G. Selander, F. Palombini, and K. Hartke. *Requirements for CoAP End-To-End Security*. Internet-Draft draft-hartke-core-e2e-security-reqs. (Work in progress). Internet Engineering Task Force, 2017.

[SW21] V. Smyslov and B. Weis. *Group Key Management using IKEv2*. Internet-Draft draft-ietf-ipsecme-g-ikev2. (Work in progress). Internet Engineering Task Force, 2021.

[TD13] M. Tiloca and G. Dini. “HISS: A HIghly Scalable Scheme for Group Rekeying”. In: *The Computer Journal* 56.4 (Apr. 2013), pp. 508–525.

[TD16] M. Tiloca and G. Dini. “GREP: a Group REkeying Protocol based on member join history”. In: *IEEE ISCC 2016*. Washington, DC, USA: IEEE, June 2016, pp. 326–333.

[Til+15] M. Tiloca et al. *Secure Two-Way DTLS-Based Group Communication in the IoT*. Internet-Draft draft-tiloca-dice-secure-groupcomm. (Work in progress). Internet Engineering Task Force, 2015.

[Til+20] M. Tiloca et al. “Group rekeying protocol based on member join history”. In: *International Journal of Information Security* 19 (Aug. 2020), pp. 343–381.

[Til+21a] M. Tiloca et al. *Group OSCORE - Secure Group Communication for CoAP*. Internet-Draft draft-ietf-core-oscore-groupcomm. (Work in progress). Internet Engineering Task Force, 2021.

[Til+21b] M. Tiloca et al. *Observe Notifications as CoAP Multicast Responses*. Internet-Draft draft-ietf-core-observe-multicast-notifications. (Work in progress). Internet Engineering Task Force, 2021.

[TNR17] M. Tiloca, K. Nikitin, and S. Raza. In: *ACM Transactions on Embedded Computing Systems* 16.3 (Apr. 2017).

[TPP21] M. Tiloca, J. Park, and F. Palombini. *Key Management for OSCORE Groups in ACE*. Internet-Draft draft-ietf-ace-key-groupcomm-oscore. (Work in progress). Internet Engineering Task Force, 2021.

[u-b20] u-blox. *Trialing OSCORE for end-to-end IoT security in resource constrained devices*. `https://www.u-blox.com/en/blogs/tech/trialing-oscore-end-end-iot-security-resource-constrained-devices`. Accessed: 2022-01-21. Feb. 2020.

[Wei06] B. Weis. *The Use of RSA/SHA-1 Signatures within Encapsulating Security Payload (ESP) and Authentication Header (AH)*. RFC 4359. Jan. 2006.

[WGL00] C. K. Wong, M. Gouda, and S. S. Lam. "Secure group communications using key graphs". In: *IEEE/ACM Transactions on Networking* 8.1 (Feb. 2000), pp. 16–30.

[WHA99] D. Wallner, E. Harder, and R. Agee. *Key Management for Multicast: Issues and Architectures*. RFC 2627. June 1999.

[YCT15] X. Yao, Z. Chen, and Y. Tian. "A Lightweight Attribute-Based Encryption Scheme for the Internet of Things". In: *Future Generation Computer Systems* 49.C (Aug. 2015), pp. 104–112.

[Zan+19] K. Zandberg et al. "Secure firmware updates for constrained iot devices using open standards: A reality check". In: *IEEE Access* 7 (May 2019), pp. 71907–71920.

[Zol20] Zolertia. *Firefly*. `https://zolertia.io/product/firefly/`. Accessed: 2022-01-21. 2020.

Formal Verification of the WirelessHART Protocol - Verifying Old and Finding New Attacks

Abstract

Connected industrial control systems open up many possibilities for increased efficiency and control but also bring drawbacks. Attacks against connected industrial control systems have highlighted the need for rigorous security analysis of every part of the systems.

In this paper, we report a careful scrutiny of the security of the WirelessHART protocol. A literature study points at the need for a formal verification of the protocol. We have modeled WirelessHART using the state-of-the-art formal verification tool Tamarin and tested the claimed security properties. Our analysis has verified the existence of all previously published attacks. In addition we have shown a new type of attack that enable an insider attacker to maliciously re-key devices in the system, effectively performing a denial of service attack on large parts of the system. Our analysis has demonstrated that WirelessHART is secure as long as no device in the network has been compromised. If an attacker can get a foothold in the network it can launch several types of attacks to compromise the network further.

Martin Gunnarsson "Formal Verification of the WirelessHART Protocol - Verifying Old and Finding New Attacks". In *Eight Annual Industrial Control System Security (ICSS) Workshop* in conjunction with Annual Computer Security Applications Conference (ACSAC) 2022.

1 Introduction

Industrial Control Systems (ICS) are becoming increasingly dependent on networking technologies and the interconnection of the system's parts to operate efficiently. Cheaper and more powerful networked devices enable data collection from industrial processes in ways that have not been previously possible. ICS-specific protocols have been developed to communicate with these new types of devices, one of them being WirelessHART [IEC16]. Its precursor, Highway Addressable Remote Transducer Protocol (HART) [Gro20], was developed in the 1980s for process automation. The HART protocol is wired and operates in point-to-point and multi-drop modes. The HART protocol can transmit both analog and digital signals in an ICS. In 2007, WirelessHART was standardized as IEC 62591. WirelessHART traffic is transmitted over IEEE 802.15.4 radio [Che+14], like ordinary WiFi. Concepts like Smart Manufacturing and Industry 4.0 [Sen+13; Neu+16] emphasize the inclusion of connected devices, sometimes called the Internet of Things (IoT), into manufacturing systems. Because WirelessHART interfaces with legacy HART technologies, it is a protocol suitable for wireless sensors connected to legacy ICS.

Security in ICS has become a pressing concern after a series of published attacks such as STUXNET [Lan11], Black Energy [Lia+16], and Triton [DDC18]. Studies have been published on attacks against cyber-physical systems and ICS [Hum+17b; Slo19] attacks against oil and gas infrastructures have been studied [SGL20]. Future challenges for ICS security have been identified [GS19] and in the realm of secure communications for ICS [Ray+16].

WirelessHART has been integrated into oil refineries [Ram+18] and ICS equipment produced by major manufacturers [The+22; LSH08]. When new technologies are integrated into ICS, they become new attack surfaces for malicious actors wanting access to the ICS. WirelessHART must be analyzed to verify the stated security goals of the protocol.

This paper will present an extensive *formal verification* of the WirelessHART security protocol. The WirelessHART standard, IEC 62591, is not openly available, making studies and security analyses of the standard more complex.

The security properties of WirelessHART have been studied since 2009 [Raz+09], and several works have been published [Bay+16; AL10]. Published attacks include Sybil-Attacks, denial-of-service attacks, and device impersonation attacks. The security of the physical and medium access control layer of WirelessHART has been studied. A jamming attack was published in 2021 [Che+21]. In 2021 the first try at a formal analysis of WirelessHART was published [LFZ21]. The authors modeled the end-to-end security protocol of WirelessHART and claimed to have found an attack. But no paper has managed to do a complete analysis of the entire WirelessHART protocol and all its details.

Previous security analyses of WirelessHART mainly used *semi-formal analysis*. The shortcoming of this approach is that it only covers some eventualities and possible executions. However, formal protocol verification techniques using *automated* proofs and the state-of-the-art prover Tamarin [Mei+13] can efficiently search for possible attacks over possible executions. Tamarin and other formal verification tools have demonstrated their utility by verifying known attacks on protocols and discovering new attacks. Analysis using Tamarin has been done on 5G-AKA [CD19], TLS1.3 [Cre+17], and recently the EMV standard [BST21], to only name a few.

The contributions of this paper are the following:

- We have cataloged and summarized all published security analyses of WirelessHART and attacks.
- We provide a detailed and comprehensive Tamarin model of the WirelessHART protocol. Our model covers the end-to-end security protocol, join sequence, network key change operation, and network advertisements. Our model allows us to find and verify all attacks already shown in previous works.
- During our analysis, we have found a novel, Malicious Re-keying attack. An insider attacker can send a malicious Network Key Change message impersonating the Security Manager. The Field Devices receiving these messages will change their keys, and the legitimate Security Manager and Gateway cannot contact the Field Devices.

The rest of the paper will start with a description of WirelessHART in section 2. Next, we will outline previous work on WirelessHART security in Section 3. In section 4, we state the system model, assumptions, and threat model used in our analysis. We will then describe the Tamarin model used in our analysis and the methodology behind it in Section 5. We present our findings in Sections 6. Finally, we present related work in Section 7 and conclude.

2 The WirelessHART protocol

In this section, we will present an overview of the WirelessHART specification. We present a shortened version focused on parts relevant to our security analysis. If the reader wishes to get all details of WirelessHART, we refer them to the WirelessHART specification[IEC16]. The WirelessHART specification details the actors and procedures needed to implement the protocol. We obtained the WirelessHART standard from the International Electrotechnical Commission (IEC)[1].

In the next section, we use the following notation: data items such as keys or identities have been written with typewriter-font, e.g., `Join_key`. We write actors in italics, e.g., *Network Manager*. We denote procedures and states defined in the WirelessHART specification with quotation marks, e.g. "Join Process".

[1] https://webstore.iec.ch/publication/24433

2.1 Actors and roles

WirelessHART is intended to facilitate communication with *Field Devices* in a factory environment. The *Plant Automation Host* will send *Commands* to the *Field Devices*. The commands can be requests to send a sensor value, move an actuator, or request information from the *Field Device*. The *Command* will be translated to WirelessHART in the *Gateway* and forwarded over WirelessHART.

An example of a small WirelessHART network can be seen in Figure 1. The figure shows all types of devices in a WirelessHART network. We will describe the types of devices below:

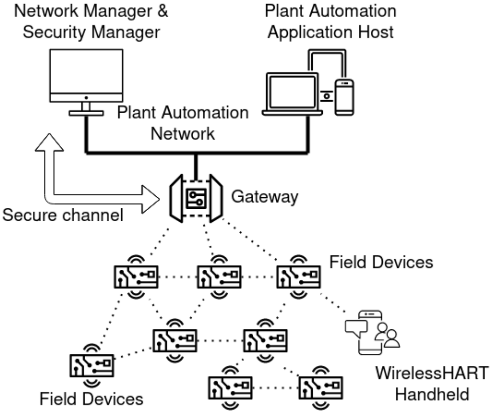


Figure 1: WirelessHART Automation Network.

Field Devices join the network existing of a *Gateway* and a *Network manager*. The *Field Devices* have a physical port, called the *Maintenance Port*, where a *Handheld Devices* can be connected for management and diagnostics. It is used to start the ”Join Process”, described later.

At least one *Gateway* connects the WirelessHART network to the rest of the factory network. WirelessHART terminates at the *Gateway*, and messages are translated and forwarded to the *Plant Automation Network*. We have limited the scope of our analysis to include the *Gateway* and left the *Plant Automation Network* out of this work since the WirelessHART standard does not cover it.

Every network must have one *Network Manager* that manages the *Field Devices* by assigning transmission network information, keys, and more required by the protocol.

Each WirelessHART network also needs a *Security Manager*. The *Security Manager* is responsible for the keys in the system. One *Security Manager* can manage more than one WirelessHART network, but each network has one, *Security Manager*. The *Security Manager* and the *Network Manager* can be co-hosted on the same machine. We illustrated such a scenario in Figure 1.

2.2 WirelessHART protocol architecture

WirelessHART defines both network architecture and the radio protocols used for communication. In addition, the specification includes several sub-protocols that form layers in a protocol stack. The WirelessHART protocol uses an altered ISO/OSI 7-layer model[Sta94] to separate the features and functions of the protocol into layers. This section will give a short overview of this.

The physical radio protocol used is IEEE 802.15.4-2006 [EE06]. Then the Physical Layer that has Medium Access Control (MAC)[2]. WirelessHART uses a mesh topology, with multiple routes for redundancy, for the *Field Devices* as indicated by the dotted lines in Figure 1.

The protocol abstraction layers are the physical Layer, the data-link Layer, the network layer, the transport layer, and the application layer. WirelessHART provides end-to-end encryption and single-hop integrity protection of messages. The security measures are handled at the network and data-link layers, and we will only focus on the relevant parts of the protocol stack. We will not go into further detail about the transport and application layers.

2.3 WirelessHART security

WirelessHART specifies several mechanisms to secure the network: single-hop message integrity protection, end-to-end encryption of data, and the management of keys.

The WirelessHART specification does not specify what adversary model the document's authors intended, and the security requirements of the protocol are not outright stated.

The specification mentions that the single-hop MIC layer is intended to protect against attackers that do not know any keys used in the network. The end-to-end transmission security protocol protects against attackers that are in the network.

To aid in reasoning about the different attacker models, we will introduce the notion of *Insider Attacker* and *Outsider Attacker* in Section 4. We will give a detailed description of what the different attackers know later, and until now, work with the assumption that *Insider Attacker* knows systems secrets and that *Outsider Attacker* does not know any secrets used in the network.

The WirelesHART specification states that *Field Devices* shall only be connected to the network after they receive the `Join_Key`. A MIC ensures hop-by-hop integrity. The key is the `Network_key` and protects against an *Outsider Attacker*. End-to-end confidentiality is achieved by encrypting messages. Here protection against *Insider Attackers* is managed. *Field Devices* use shared symmetric keys to secure communication with each other and provide authentication.

[2] Usually, MAC is the acronym for Message Authentication Code. In the WirelessHART specification, MAC stands for Medium Access Control. To avoid confusion, we use Message Integrity Code (MIC) in this work.

The WirelessHART standard also reserves itself against attacks on cryptographic primitives and the implementation of cryptographic functions.

2.4 Keys and Key Distribution

WirelessHART only uses symmetric key cryptography. The *Security Manager* is responsible for provisioning keys to all *Field Devices* in the network. When a new device joins the network, the `Join_key` is used. The Join_key *can* be different for each device according to the standard. The `Network_Key` is shared with all devices. It is used to compute and verify the DLPDU MIC. WirelessHART sends data through sessions between a *Field Device* and either the *Network Manager* or the *Gateway*. The session can be either unicast or broadcast. A `Unicast_Session_Key` is unique to a *Field Device*. `Broadcast_Session_Key`, are shared with all *Field Devices*. Each *Field Device* has a unicast session with the *Network Manager*. All *Field Devices* also share a broadcast channel with the *Network Manager*.

2.5 Error detection and the End-to-End security Protocol

Two parts of the protocol stack are used to ensure end-to-end message confidentiality and tampering resistance on messages.

The cryptographic primitives in WirelessHART is an Authenticated Encryption with Associated Data (AEAD), specifically AES-CCM*[EE06].

The Data-Link layer prevents the *Outsider Attacker* from tampering with messages as the Network_Key is used to compute a MIC over the contents of the DLPDU. The MIC is computed using AES-CCM* with the contents of the DLPDU as associated data and an empty plaintext field. The Network_Key is used to compute the DLPDU MIC. Each device that forwards the message can verify the MIC, preventing tampering with the message.

In Table 1, we show a DLPDU and the fields. The payload and all fields except the CRC are authenticated, and integrity protected using the `Network_key`. The nonce used for calculating the MIC included in the DLPDU is formed by concatenating the ASN, a network-wide counter value, with the source address.

Table 1: A WirelessHART Data Link Layer Protocol Data Unit (DLPDU)

Data Mode	Address Specifier	Sequence Number	Network_ID	Destination Address	Source Address	DLPDU Specifier	DLL Payload	MIC	CRC

Messages are encrypted to prevent information disclosure to either an *Outsider Attacker* or an *Insider Attacker*. Encryption is done at the Network Layer by encrypting the NPDU contents. The same AES-CCM* algorithm provides both encryption and a MIC. In Table 2, we show the fields in an NPDU. The NPDU Payload is encrypted and authenticated, and the rest of the NPDU is authenticated except the Hop-to-live, Counter, and MIC fields.

Table 2: A WirelessHART Network Layer Protocol Data Unit (NPDU)

NL Control	Hop-to-live	Sequence Number	Graph_ID	Destination Address	Source Address	(Proxy route)	0-N (source routes)	Security Control	Counter	MIC	Payload

A `Session_Key` or `Join_Key` is used to encrypt the NPDU. The nonce for the NPDU MIC is constructed by concatenating a counter with the source device's ID. Encrypted NPDU sent between two parties is a session and provides confidentiality, integrity protection, source authentication (provided it is a unicast session), and replay protection for the transmitted data. Replay protection is achieved by using incremental sequence numbers. A device will not accept messages with a lower sequence number than previously received and decrypted.

Two types of sessions exist, unicast and broadcast sessions. Sessions terminating in the *Network Manager* are used for managing the network and *Field Devices*. The sessions terminating in the *Gateway* are used for actual process data to and from the *Gateway*. During the "Join Sequence", a *Field Device* is provisioned with one unicast session and one broadcast session with both the *Gateway* and *Network Manager*, for a total of four sessions.

2.6 WirelessHART Join Sequence

The WirelessHART "Join Sequences" defines the messages transmitted and the steps taken when a new *Field Device* is connected to a WirelessHART network. The protocol can be seen in Figure 2. The process is described step by step below:

0. Before enrolling a device the *Security Manager* generate a `Join_Key` for the new device. The `Join_Key` is transferred to the hand-held Maintenance Tool.

1. The *Field Device* being connected is connected to a hand-held Maintenance Tool with a cable to the Maintenance Interface that is a physical port on the *Field Device*. The Maintenance Tool can then transfer a `Join_key` and a `Network_name` to the *Field Device*. The `Network_Name` allows the *Field Device* to start scanning for a network to join.

2. When a network advertisement with the correct name has been received, a Join Request is sent to the *Network Manager*. The Join Request is forwarded over the mesh network to the *Network Manager*. The Join Request comprises the joining *Field Device's* identity and information about the neighboring *Field Devices*. All this is encrypted with the joining *Field Device's* `Join_Key`. The `Well-Known-Key` is used to compute the DLPDU MIC. The *Network Manager* decrypts and verifies the message using the `Join_key`.

   The *Network Manager* responds by sending a *Field Device* Nickname, the `Network_Key` and one or more Session keys, generated by the *Network*

Manager, encrypted with the Join_Key. The Joining *Field Device* will acknowledge this message using the received *Network Manager* unicast session key and the Network_Key.

3. The *Network Manager* now sends network information to the joining *Field Device*, such as time sources and routes, and links.

4. The *Field Device* is left in a ”quarantined” state. It can communicate with the rest of the *Field Devices* and the *Network Manager* but does not have a session to communicate with the *Gateway*.

5. The *Network Manager* can leave the joining *Field Device* in ”quarantine” for some time or directly provision a *Gateway* session to the joining *Field Device*. Once the joining *Field Device* has a session with the *Gateway*, it can communicate with the process network. The joining *Field Device* is now a part of the network and considered operational.

2.7 Network Key Change Operation

WirelessHART has a special sub-protocol defined for changing keys for nodes in the network. All keys used by WirelessHART can be changed with this protocol, including Network_Key,
Broadcast_Session_Key, and Unicast_Session_Key. The protocol is executed between a *Field Device* and the *Network Manager*, with neighbors and mesh devices forwarding messages from the source to the destination. We show a diagram of the protocol in Figure 3.

The steps taken to change a key are:

1. The *Network Manager* transmits a Broadcast message that is forwarded to all *Field Devices*. The *Network Manager* Broadcast key protects the Command.

2. Each *Field Device* sends a response, and the response is encrypted with the *Network Manager* Unicast key preventing spoofing attacks.

3. The *Network Manager* can verify that all *Field Devices* have acknowledged the new key. All *Field Devices* will change to the new key at a later time slot.

The specification allows a key to be sent out before an anticipated change of keys, and sending the keys in advance allows retransmission if the message is lost. Since each *Field Device* acknowledges the received key, the *Network Manager* can verify that the key was received.

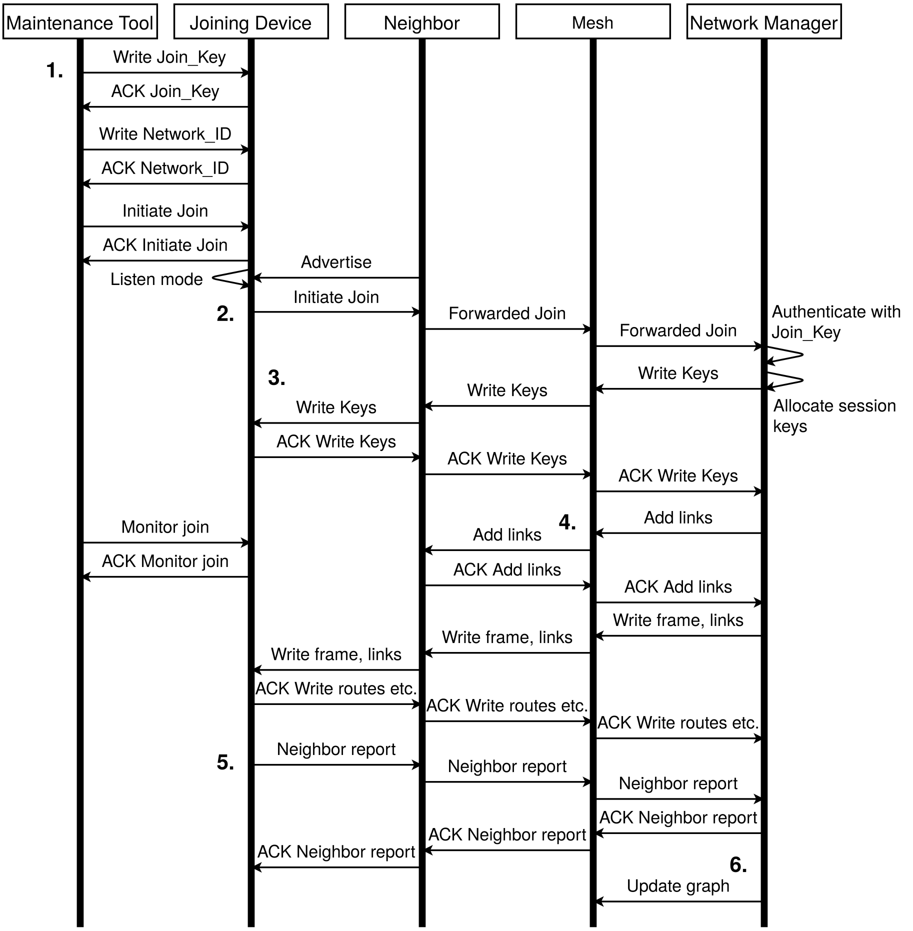


Figure 2: WirelessHART Join Sequence.

3 Previous security analysis of WirelessHART

In [Raz+09; Raz10], the authors note the lack of security requirements in the WirelessHART specification and claim that they have done the first security analysis of the protocol. Furthermore, the authors argue that the following attacks need more attention: message corruption attacks, jamming attacks, and traffic analysis. Denial of Service attacks, De-synchronization attacks, and Wormhole attacks unless source routing is used. The authors claim that tampering with messages can be done if an adversary knows the `Network_Key`. Therefore, the authors recommend regular changing of the `Network_Key`. Further, an adversary can spoof advertisement messages with the `Well-Known-Key`.

In [Bay+15a], Bayou et al. present a Disconnect Sybil attack against WirelessHART. In the Disconnect Sybil attack, the attacker A with knowledge of the

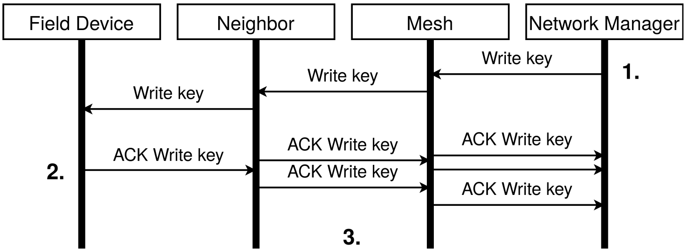


Figure 3: Network Key change operation.

`Network_Key` spoofs a DLPDU with the source address of the victim *V*. Knowing the Network_Key *A* can trick the neighbors of *V* into removing *V* from their tables. The neighbors of *V* will also forward the message to the *Network Manager* that *V* has left the network. *V* will be cleared from the *Network Managers* routing tables.

In [Bay+16], Bayou et al. present more attacks. This new attack allows an attacker to inject malicious Commands into a WirelessHART network. Additionally, the attack allows an *Insider Attacker* to leverage the knowledge of `Network_Key` to inject commands into the network.

The attacker will use broadcast session keys to enable this attack. Three attacks are presented: direct command injection bounced command injection and on-the-fly command injection. The three classes use the knowledge of a broadcast session key to spoof the sender of commands, tricking a *Field Device* into executing a malicious command.

In [Rap+18], the authors claim to have found a potential DOS attack similar to an attack briefly described in [Raz+09]. In this *exhaustion attack*, the attacker uses the `Well-Known-Key` to send Advertisements to a joining device. The joining device will respond to this (false) advertisement and be unable to join the legitimate network. An attacker can combine this attack with a de-authentication attack such as the one presented in [Bay+15a].

We have summarized published security analyses of WirelessHART. Most of these works present attacks against the protocol in some form or another. We will look closer at the published attacks and establish the root cause. Table 3 present the previously published attacks, the type of vulnerability, and their root causes. We have used the threat terminology from the STRIDE model [Her+06] to describe the root causes of the attacks. We have chosen the STRIDE model because it is complete and covers more threat classes than the Confidentiality Integrity Authentication (CIA) model.

The STRIDE model defines attacks of the following types: spoofing, tampering, replay attacks, information disclosure, denial of service, and elevation of privilege.

Table 3 shows that most of the published attacks against WirelessHART are

Table 3: A tabulation of published attacks against the WirelessHART protocol. (* Discussion, not a full attack)

Attack Name	Root Cause	Attacker	Published
Jamming Attacks	DoS	Outsider	[Raz+09]*
Wormhole Attacks	Spoofing	Insider	[Raz+09]*
Message Corruption	Tampering	Outsider	[Raz+09]*
Blackhole Attacks	Spoofing	Insider	[Raz+09]*
Disconnect Sybil Attack	Spoofing	Insider	[Bay+15a]
Pre-Join exhaustion	DoS	Outsider	[Raz+09]*[Rap+18]
Direct CI	Spoofing	Insider	[Bay+16]
Bounced CI	Spoofing	Insider	[Bay+16]
On-the-fly CI	Tampering	Insider	[Bay+16]

Spoofing attacks where the adversary knows the
`Network_Key`. One final remark we want to make here is that in several papers, the authors have claimed that a specific attack is not possible, only for the attack to be published later. This is a further argument for formal protocol verification. A rigorous analysis of a protocol has a greater chance of discovering all possible attacks and preventing erroneous claims about the security of a protocol.

4 Threat models and security assumptions

Since the WirelessHART specification does not properly specify the capabilities of a potential attacker, except for being unable to break cryptographic primitives or implementations, we have specified two potential attackers for our security analysis. We already introduced them briefly in Section 2.3, and here we provide the complete definitions.

The *Outside Attacker* is an active attacker, assumed to know no secrets from the system. The `Well-Known-Key` is explicitly known by all WirelessHART devices. The *Outside Attacker* can inject messages, replay messages, and discard messages. This is the same capabilities as a Dolev-Yao adversary [DY83].

The *Insider Attacker* is also an active attacker who knows one device's keys. We leave out of scope what specific device and how the attacker gained this knowledge. Apart from this, the *Insider Attacker* has the same capabilities as an *Outsider Attacker*, functioning as a Dolev-Yao adversary. The difference is that *Insider Attacker* can decrypt messages, spoof identities, and tamper with messages without being detected. This can be used for information disclosure and denial of service. In Table 4, we present all considered keys in the system and detail what keys the different Attacker knows.

The WirelessHART standard state that the `Join_key` *can* be unique. Using the same, `Join_key` for multiple *Field Devices* in the network opens possibilities

Table 4: Keys known by Inside Attacker and Outsider Attacker. (K - system-wide key known, U - Unknown, K* - One key is known)

	Outsider	Insider
Join_Key	U	K*
Network_Key	U	K
Network Manager Broadcast Session Key	U	K
Network Manager Unicast Session Key	U	K*
Gateway Unicast Session Key	U	K*
Well-Known-Key	K	K

of nonce-reuses and replay attacks on other ”Join Sequences”. A Join_Key known to the adversary provides no security for the ”Join Sequence”. An attacker could intercept all messages transmitted during the ”Join Sequence” and learn any new keys sent to a *Field Device*. The security of the ”Join Sequence” hinges on the secrecy and uniqueness of the Join_key.

5 Analyzing WirelessHART with Tamarin

The Tamarin Prover [Mei+13] is a tool developed to model and verify the properties of security protocols. Protocols are modeled in a domain-specific language based on multi-set rewriting rules. The domain-specific languages include constructions such as Diffie-Hellman, symmetric and asymmetric encryption, and hash functions. Tamarin users can specify custom constructions, like Hash-based Message Authentication Codes (HMAC).

The rules defined for both the protocol and adversary can generate facts. In this context, a *fact* can be understood as the knowledge of an entity at a specific time. The rule Fr(~n)]-->[Out(~n), A(~n)] generates a random number n. The number is then sent out on the communications network. The result is the *fact* A(~n) that can be used in further rules, and that ~n is sent out on the network, letting the adversary know n. A full description of Tamarin's modeling language can be found in the Tamarin manual[3].

In Tamarin, the adversary is also defined using rules and facts. These rules and facts represent the adversary's knowledge at different states of protocol execution. The adversary in Tamarin is modeled as a Dolev-Yao adversary [DY83]. A Dolev-Yao adversary can intercept all messages, forge messages, impersonate entities, and decrypt messages using information known to the adversary. Cryptographic primitives are assumed to be ideal and modeled algebraically [Mei13].

The execution of rules produces an ordered sequence of facts, called a *trace*. A trace can be seen as one possible execution of the rules in the protocol. *Lemmas*

[3] https://tamarin-prover.github.io/manual/index.html

are the properties of a trace that Tamarin tries to prove or find a counterexample. A lemma can specify that for all protocol executions, there must be no point where the adversary knows a secret message `m`. Tamarin will either prove that a property stated in a lemma holds or find a counterexample. Tamarin might not terminate since finding a proof or a counterexample is undecidable. The verification of properties always terminates in our analysis.

5.1 WirelessHART model

To model the WirelessHART protocol, we mainly used Tamarin's built-in formulas. Tamarin does not have built-in support for AEAD operations but allows the user to specify formulas. We have added `aead_encrypt/4`, `aead_decrypt/4`, `aead_verify/4`, `true/0`, `mic/2`, and `verify_mic/3`. In this notation `formula/n`, `n` is the number of arguments to the formula. In the following equations, `k` denotes the key, `n` the nonce, `aad` the additional authenticated data, and p the plaintext. The formulas are specified as follows:

```
aead_decrypt(k, n, aad, aead_enc(k, n, aad, p)) = p ]
aead\_verify(k, n, aad, aead\_enc(k, n, aad, p)) = true ]
verify_mic(mic(k,p), k, p) = true ]
```

We used the formulas for AEAD operations in the NPDU layer of the protocol, where both the encryption and authentication properties are used. We chose to use the MIC construction for the DLPDU, where an AEAD is used with the plaintext input set to NULL, providing integrity only.

We have also used the restriction `Eq_testing()` to force testing of equality in traces. We used this to force validation of `verify_mic` and `aead_verify`.

```
Eq_testing: "All x y #i.  Eq(x, y) @ i ==> x = y"
```

We focused on modeling the 'end-to-end security protocol', 'Join procedure', and 'Network key change operation'. These parts of WirelessHART are responsible for most security-critical operations. We chose to omit the replay protection scheme from our model since it uses sound constructions. For the 'Join procedure', we omitted messages after the step where the *Field Device* has accepted the new key since later steps of the procedure handles network management. When modeling the 'Network key change operation', we omitted the acknowledgment sent by the *Field Device*. The NLDPU and DLPDU headers were also simplified to reduce the number of terms.

The 'Pre-join exhaustion attack' and 'Disconnect Sybil attack' target the authentication of DLPDU messages. We, therefore, modeled the DLPDU message layer.

To keep our analysis manageable, we split our model into four parts: the End-to-End Security Protocol, the Join Sequence, the "Network Key Change Operation", and the DLPDU commands.

We will detail each part of the protocol in the following sections. We provide our model for further study[4].

5.2 Security Analysis of the WirelessHART End-to-End Security Protocol

The end-to-end security protocol transports data and transmits configurations to devices. We modeled the DLPDU and NPDU protocol layers and created rules for sending and receiving messages for both unicast and broadcast sessions.

Security Lemmas and results

We have tested the following security properties of the WirelessHART end-to-end security protocol: payload secrecy and impersonation of a gateway. In Section 4, we discussed the notion of *Insider Attacker* and *Outsider Attacker*. These are the notions we will use in this section.

We analyzed the end-to-end security protocol's security using unicast and broadcast keys. For brevity, we only provide the lemmas for the broadcast case here.

The first lemma we present verifies the security of the sent command. It states that the adversary shall not be able to learn any command sent by the *Gateway*. A command known by the adversary implies that the broadcast key has been compromised. In other words, an *Outsider Attacker* can use its broadcast session key and decrypt the transmitted message and learn the command.

A similar lemma holds for unicast sessions so long as the unicast session keys used to encrypt the command are not compromised.

```
Eq_testing: "All x y #i.  Eq(x, y) @ i ==> x = y"
lemma command_secrecy:
  "All GW D k #i #j.
    SendCommand_BC(GW, D, k) @i & K(k) @j
    ==>
    Ex #k. InsiderAttacker() @k"
```

The lemma below verifies the authentication and integrity of a received command encrypted with a broadcast session key. A command accepted by a *Field Device* means that it either was legitimately sent by the Gateway or that an *Outsider Attacker* has compromised the broadcast session key and impersonated the *Gateway*.

A command sent over a unicast session is authentic as long as the unicast session key is not compromised, requiring the compromise of either the *Gateway* or the *Field Device*.

[4] https://github.com/Gunzter/Formal-Verification-of-the-WirelessHART-Protocol

```
lemma command_authentication:
  "All GW D k #i. ReceivedCommand_BC(GW,D,k) @i
    ==> (Ex #j. SendCommand_BC(GW,D,k) @j & j<i)
    | (Ex #k. InsiderAttacker() @k)"
```

5.3 Security Analysis of the WirelessHART Join Sequence

We modeled the Join Sequence by writing rules for creating the *Security Manager*, Gateway, and *Field Devices*. To model the execution of the protocol, we created the following rules: a *Field Device* sending a Join request message, the *Security Manager* receiving the Join request message and responding with encrypted keys, and a *Field Device* receiving the encrypted keys. We have modeled the protocol so that the joining *Field Device* is provisioned with the Join Key out-of-band. This is done with a direct connection to a handheld Maintenance tool.

Security Lemmas and results

To verify the security of the WirelessHART Join Sequence, we will test the following security properties: key secrecy, key integrity, and resistance to impersonation attacks. These lemmas are tested with a unique and random Join_Key since the protocol is trivially broken when using a known Join_Key.

The first lemma verifies the key authentication and integrity from the joining *Field Device's* perspective. The lemma state that a received encrypted key means that either the encrypted keys were legitimately sent by the *Security Manager* to the *Field Device* or that the Join_Key has been compromised.

```
lemma  key_authentication:
  "All a  k #m.
    KeysReceived(a, k) @m
    ==> (Ex #l. KeysSent(a, k) @l) |
    (Ex #j. InsiderAttacker() @j)"
```

The second lemma verifies the secrecy of the received key. A received key known by the adversary implies that it was previously sent by the *Security Manager* and that the Join_Key was compromised.

```
lemma  key_secrecy:
  "All a k #m #n.
    KeysReceived(a, k) @m & K(k) @n
    ==> (Ex #l. KeysSent(a, k) @l) &
    (Ex #m. InsiderAttacker() @m)"
```

5.4 Security Analysis of the WirelessHART Network Key Change Operation

When modeling the ”Network Key Change Operation”, we have written rules corresponding to the *Security Manager* sending the change key message and the *Field Device* receiving the key change message. We wrote rules for the ”Network Key Change Operation” over unicast and broadcast sessions.

Security Lemmas

The following lemma was used to verify the security of the ”Network Key Change Operation”. The operation can be done over unicast and broadcast sessions. We have only stated the broadcast lemmas here for brevity. The first lemma verifies the secrecy of the new key. Provided no *Field Device* has been compromised and revealed the broadcast key, the new key is secret against an *Outsider Attacker*. An *Outsider Attacker* can decrypt the message containing the new key and use that to decrypt future traffic.

```
lemma key_secrecy:
  "All NM D k #i #j.
    SendKey_BC(NM, D, k) @i & K(k) @j
    ==>
    Ex #k. InsiderAttacker() @k"
```

The following lemma verifies the authentication and integrity properties of the ”Network Key Change Operation”. Like in the case of key secrecy, the protocol is secure against an *Outsider Attacker* but does not hold against an *Outsider Attacker*. An *Outsider Attacker* can use its broadcast session key and `Network_Key` and spoof a message containing a fake key to the *Device*.

```
lemma key_authentication:
  "All NM D k #i. NewKeyReceived_BC(NM,D,k) @i
    ==> (Ex #j. SendKey_BC(NM,D,k) @j & j<i)
    | (Ex #k. InsiderAttacker() @k)"
```

5.5 Verification of Previously Published Attacks

We had to model more parts of the WirelessHART protocol to verify the previously published attacks. The Command Injection attacks were confirmed with the lemmas used to verify the end-to-end security protocol. The Disconnect Sybil attack and the Pre-Join Exhaustion attack required us to model the DLPDU control messages. DLPDU messages do not use the NPDU layer and consist of a command sent between two neighboring devices. The message is only MIC:ed using the `Network_Key` or the `Well-Known-Key`.

Security Lemmas

To verify the Disconnect Sybil Attack, we constructed a lemma that tests the authenticity of received disconnect DLPDU messages. The lemma verifies the authenticity of received disconnect commands. A disconnect command will be verified as authentic and implies that it previously has been legitimately transmitted or that an adversary has compromised the Network_Key and sent a forged disconnect command.

```
lemma disconnect_authentication:
  "All A B #i. ReceivedDisconnectCommand(A,B) @i
    ==> (Ex #j. SendDisconnectCommand(A,B) @j & j<i)
    | (Ex #k. InsiderAttacker() @k)
```

To verify the Pre-Join Exhaustion attack, we created a lemma that verifies the authenticity of the received Advertisement message. This lemma was trivially falsified since the Advertisement message is MIC:ed using the Well-Known-Key. An adversary can use the Well-Known-Key to create malicious Advertisement messages that the joining *Field Device* will accept.

```
lemma advertisement_authentication:
  "All A B #l.
    ReceivedAdvertisement(A, B) @l
    ==> (Ex #m. SendAdvertisement(A, B) @m)"
```

6 Results and discussion

In this section, we will present the results of our formal protocol analysis. We will discuss our findings for WirelessHART as a protocol and formal protocol verification as a technique.

6.1 Previously Published Attacks

As described in the previous section, we have modeled and analyzed the WirelessHART protocol. The three main parts of the protocol, the Join Sequence, the end-to-end Security Protocol, and the ”Network Key Change Operation”, were then analyzed using the lemmas presented in Section 5. We show the previously published attacks in Table 5. Our Tamarin analysis found all previously published attacks or variants of them.

Table 5: An overview of the results of our analysis previously shown in Table 3. (✓- Attack could be verified)

Attack Name	Root Cause	Source	Verified?
Disconnect Sybil Attack	Impersonation	[Bay+15a]	✓
Pre-Join exhaustion	DoS	[Raz+09; Rap+18]	✓
Direct CI	Impersonation	[Bay+16]	✓
Bounced CI	Impersonation	[Bay+16]	✓
On-the-fly CI	Tampering	[Bay+16]	✓

6.2 Novel Malicious Re-keying Attack

Modeling the ”Network Key Change Operation”, we found a new, previously unpublished attack. An *Insider Attacker* can utilize the known Broadcast Keys and maliciously transmit re-key commands to *Devices*. These *Field Devices* will change their keys according to the received command, bringing them out of synchronization with the *Network Manager* and the *Gateway(s)*. This attack can change any key used in the system. Maliciously re-keying of *Devices* and preventing the *Security Manager* and *Gateway* from communicating with re-keyed *Devices* is, in effect, a DoS attack on the parts of the network. We show the attack in Figure 4. The malicious adversary has compromised one *Field Device* and is now an *Outsider Attacker*.

1. The *Outsider Attacker* sends two ”Command 963 - Write Session”, with Session Key 1* and Session Key 2* to the *Field Devices*. Next, a ”Command 961 - Write Network_Key”, containing `Network_Key*` is transmitted. These commands are encrypted with the current Session Key 1 and MIC:ed with the current `Network_Key`. These commands will be received by the targeted *Field Devices* and authenticated as coming from the *Security Manager*.

2. The *Field Devices* will change keys to the maliciously provided Session Key 1*, Session Key 2*, and `Network_Key*`. The *Security Manager* and Plant Automation Host do not know these keys, so the legitimate owners of the *Field Devices* cannot contact the *Field Devices*.

Recovering from this attack requires finding and recovering the compromised *Outsider Attacker Field Device* and a complete re-provisioning of all *Field Devices* in the network.

6.3 Security implications for WirelessHART

We can see that the root cause of many attacks is compromised broadcast keys. Symmetric key encryption needs key secrecy, and a compromised *Field Device* breaks this requirement, as in the case with *Insider Attacker*. As we show in Table 4, not only `Network_Key` is revealed to an *Insider Attacker*, but also the *Network Manager* unicast and broadcast keys and the session keys. The authors of

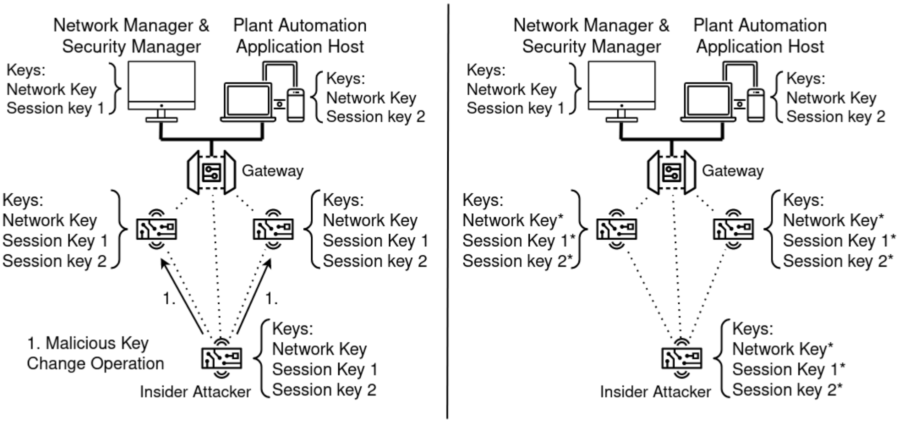


Figure 4: Malicous re-keying attack.

the WirelessHART specification have failed to account for this fact. Given the above insight, we conclude that the ”Network Key Change Operation” when Network Manager Broadcast keys are compromised is insecure. An *Outsider Attacker* can decrypt the messages in the Key Exchange messages with the Network Manager Broadcast key. The WirelessHART specification recommends periodic key changes [IEC16] (Appendix A.3) to ensure the system's secure operation. This would not help the network's security since new keys are transmitted encrypted using a key known to the attacker, and the new key would be revealed to the attacker if the old key was compromised.

A potential remedy would be to use Network Manager Unicast keys to update the keys and limit communication to unicast sessions only. Individual key changes with unicast keys will increase the network overhead. Still, the owner and operator of a network can avoid revealing a key to a compromised device, i.e., a *Field Device* controlled by *Insider Attacker*.

Furthermore, there is a severe issue of spoofing attacks. Broadcast Session Keys are used to encrypt and protect the integrity of messages using a symmetric algorithm, which is not secure. It is difficult to imagine how these attacks could be mitigated without considerable changes to the WirelessHART specification.

Lastly, the Pre-Join exhaustion attacks can be prevented by provisioning the current `Network_Key` to the joining *Field Device*. This will prevent any attacker in the proximity from abusing the well-known key to spoof messages.

The Disconnect Sybil attack is more difficult to remedy. A *Field Device* does not share a unicast key with its neighbors, which also may change over time, so only the `Network_Key` can be used to authenticate messages from a neighbor device.

7 Related work

We have divided the preliminary research for this paper into two categories. Prior work on WirelessHART security and Formal Protocol Verification.

7.1 WirelessHART Security

In Section 3, we have already covered published attacks and security analyses of WirelessHART [Raz+09; Raz10; Bay+15a; Bay+16; Rap+18]. In this section, we will expand the discussion to cover attacks that target the lower layer of the protocol stack, such as jamming attacks.

Several other papers have studied the security of WirelessHART, summarizing published attacks but not presenting any new attacks. In [AL10], the authors analyze several protocols for IoT and ICS, among them WirelessHART. NetSIM, a simulator for WirelessHART SCADA systems, is presented in [Bay+15b]. The simulator is used to demonstrate the Disconnect Sybil attack from [Bay+15a] Intrusion Detection Systems (IDS) and network monitoring systems for WirelssHART have been studied [Rap+18]. A jamming attack against WirelessHART was published in 2021 [Che+21].

7.2 Formal Protocol verification of ICS & IoT protocols

Formal Protocol verification has been used with great success to find attacks in other protocols for IoT and ICS.

In their paper[Kim+17], the authors write about their process of verifying IoT protocols. They use Tamarin to analyze CoAP over DTLS, SigFox, LoRa, and MQTT. LoRaWAN has been analyzed [BPG18] and [Wes+20], where more vulnerabilities were found. The proposed key establishment protocol EDHOC has been modeled and formally verified [Bru+18; NSB21]

The Virtual Private Network protocol Wireguard has also been formally verified [DM17]. Tamarin has been used during the development process of TLS1.3 [Cre+17]. The rationale was to verify the protocol before it was finalized and deployed. Then the protocol designers could address any issues before the protocol was deployed.

8 Conclusion

WirelessHART has been studied in the literature, and multiple attacks have been found in several published works. Formal verification of WirelessHART has been done, albeit in a limited scope. To our knowledge, our work is the first to model and verify all parts of WirelessHART.

We have modeled the WirelessHART end-to-end security protocol, the WirelessHART Joining sequence, and the WirelessHART Network Key Change Operation and tested them against both an *Insider Attacker* and *Outsider Attacker*.

We have shown that the previously published attacks can be found using formal protocol verification. The attacks were of varying types and against different parts of the WirelessHART protocol.

Further, we have extended the discussion about the threat model of WirelessHART. We have introduced a distinction in adversary models with the *Insider Attacker* and the *Outsider Attacker*. These attackers' knowledge necessitates considering two separate cases when analyzing the protocol.

When formally verifying the Network Key Change Operation, we found a new attack on the WirelessHART protocol. An Insider Attacker can maliciously change the keys of non-compromised devices in the network, making them uncontactable by the rest of the network, including the Security Manager. This attack is severe since only a single device needs to be compromised for an Insider Attacker to put the entire network offline.

The WirelessHART protocol would benefit from an update to the standard. New mechanisms with integrity protection of all messages are needed to prevent tampering and stronger message authentication. A topic for further research would be implementing the improvements and making a formal verification of the new version of the protocol.

Acknowledgements

We would like to thank Simon Bouget for his valuable help with Tamarin and the anonymous reviewers for their helpful comments. This work has been supported by framework grant RIT17-0032 from the Swedish Foundation for Strategic Research.

References

[AL10] C. Alcaraz and J. Lopez. "A security analysis for wireless sensor mesh networks in highly critical systems". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 40.4 (2010), pp. 419–428.

[Bay+15a] L. Bayou et al. "Security issue of wirelesshart based SCADA systems". In: *International Conference on Risks and Security of Internet and Systems*. Berlin, Germany: Springer, 2015, pp. 225–241.

[Bay+15b] L. Bayou et al. "WirelessHART NetSIM: a WirelessHART SCADA-based wireless sensor networks simulator". In: *Security of Industrial Control Systems and Cyber Physical Systems*. Berlin, Germany: Springer, 2015, pp. 63–78.

[Bay+16] L. Bayou et al. “Security analysis of WirelessHART communication scheme”. In: *International Symposium on Foundations and Practice of Security*. Berlin, Germany: Springer, 2016, pp. 223–238.

[BPG18] I. Butun, N. Pereira, and M. Gidlund. “Analysis of LoRaWAN v1. 1 security”. In: *Proceedings of the 4th ACM MobiHoc Workshop on Experiences with the Design and Implementation of Smart Objects*. New York, NY, USA: ACM, 2018, pp. 1–6.

[Bru+18] A. Bruni et al. “Formal verification of ephemeral Diffie-Hellman over COSE (EDHOC)”. In: *International Conference on Research in Security Standardisation*. Berlin, Germany: Springer, 2018, pp. 21–36.

[BST21] D. Basin, R. Sasse, and J. Toro-Pozo. “The EMV Standard: Break, Fix, Verify”. In: *2021 IEEE Symposium on Security and Privacy (SP)*. New York, NY, USA: IEEE, 2021, pp. 1766–1781.

[CD19] C. Cremers and M. Dehnel-Wild. “Component-based formal analysis of 5G-AKA: Channel assumptions and session confusion”. In: *Network and Distributed Systems Security (NDSS) Symposium 2019*. Internet Society. Reston, VA, USA: Internet Society, 2019, pp. 1–15.

[Che+14] D. Chen et al. “WirelessHART and IEEE 802.15. 4e”. In: *2014 IEEE International conference on industrial technology (ICIT)*. IEEE. New York, NY, USA: IEEE, 2014, pp. 760–765.

[Che+21] X. Cheng et al. “Launching smart selective jamming attacks in wirelesshart networks”. In: *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*. New York, NY, USA: IEEE, 2021, pp. 1–10.

[Cre+17] C. Cremers et al. “A comprehensive symbolic analysis of TLS 1.3”. In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. New York, NY, USA: ACM, 2017, pp. 1773–1788.

[DDC18] A. Di Pinto, Y. Dragoni, and A. Carcano. “TRITON: The first ICS cyber attack on safety instrument systems”. In: *Proc. Black Hat USA*. Vol. 2018. San Francisco, CA, USA: Black Hat, 2018, pp. 1–26.

[DM17] J. A. Donenfeld and K. Milner. *Formal Verification of the WireGuard Protocol*. Tech. rep. www.wireguard.com, 2017.

[DY83] D. Dolev and A. Yao. “On the Security of Public Key Protocols”. In: *IEEE Transactions on information theory* 29.2 (1983), pp. 198–208.

[EE06] I. of Electrical and E. Engineers. *IEEE 802.15.4-2006 - Local and metropolitan area networks - Specifications for Low Rate Wireless Personal Area Networks (WPANs)*. Specification. Institute of Electrical and Electronics Engineers, 2006.

[Gro20] F. Group. *HART Communication Protocol Specification*. Specification. Geneva, CH: FieldComm Group, May 2020.

[GS19] S. Ghosh and S. Sampalli. "A Survey of Security in SCADA Networks: Current Issues and Future Challenges". In: *IEEE Access* 7 (2019), pp. 135812–135831.

[Her+06] S. Hernan et al. "Threat modeling-uncover security design flaws using the stride approach". In: *MSDN Magazine-Louisville* 15.1 (2006), pp. 68–75.

[Hum+17b] A. Humayed et al. "Cyber-Physical Systems Security—A Survey". In: *IEEE Internet of Things Journal* 4.6 (2017), pp. 1802–1831.

[IEC16] I. E. C. (IEC). *Industrial networks – Wireless communication network and communication profiles – WirelessHART*. Standard. Geneva, CH: International Electrotechnical Commission (IEC), Mar. 2016.

[Kim+17] J. Y. Kim et al. "Automated analysis of secure internet of things protocols". In: *Proceedings of the 33rd Annual Computer Security Applications Conference*. New York, NY, USA: ACM, 2017, pp. 238–249.

[Lan11] R. Langner. "Stuxnet: Dissecting a cyberwarfare weapon". In: *IEEE Security & Privacy* 9.3 (2011), pp. 49–51.

[LFZ21] F. Luo, T. Feng, and L. Zheng. "Formal Security Evaluation and Improvement of Wireless HART Protocol in Industrial Wireless Network". In: *Security and Communication Networks* 2021 (2021), pp. 1–15.

[Lia+16] G. Liang et al. "The 2015 Ukraine blackout: Implications for false data injection attacks". In: *IEEE Transactions on Power Systems* 32.4 (2016), pp. 3317–3318.

[LSH08] T. Lennvall, S. Svensson, and F. Hekland. "A comparison of WirelessHART and ZigBee for industrial applications". In: *2008 IEEE international workshop on factory communication systems*. New York, NY, USA: IEEE, 2008, pp. 85–88.

[Mei+13] S. Meier et al. "The TAMARIN Prover for the Symbolic Analysis of Security Protocols". In: *International Conference on Computer Aided Verification*. Springer. 2013, pp. 696–701.

[Mei13] S. Meier. “Advancing automated security protocol verification”. PhD thesis. ETH Zurich, 2013.

[Neu+16] R. Neugebauer et al. *Industrie 4.0-From the perspective of applied research*. 2016.

[NSB21] K. Norrman., V. Sundararajan., and A. Bruni. “Formal Analysis of EDHOC Key Establishment for Constrained IoT Devices”. In: *Proceedings of the 18th International Conference on Security and Cryptography - SECRYPT,* INSTICC. Setúbal, Portugal: SciTePress, 2021, pp. 210–221.

[Ram+18] B. Ramos et al. “A Perfomance Comparison of WirelessHART and ZigBee in Oil Refinery”. In: *2018 IEEE-APS Topical Conference on Antennas and Propagation in Wireless Communications (APWC)*. New York, NY, USA: IEEE, 2018, pp. 846–849.

[Rap+18] D. Raposo et al. “Securing wirelessHART: Monitoring, exploring and detecting new vulnerabilities”. In: *2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*. New York, NY, USA: IEEE, 2018, pp. 1–9.

[Ray+16] A. Ray et al. “Future research challenges of secure heterogeneous industrial communication networks”. In: *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*. Vol. 1. New York, NY, USA: IEEE, 2016, pp. 1–6.

[Raz+09] S. Raza et al. “Security considerations for the WirelessHART protocol”. In: *2009 IEEE Conference on Emerging Technologies & Factory Automation*. New York, NY, USA: IEEE, 2009, pp. 1–8.

[Raz10] S. Raza. “Secure Communication in WirelessHART and its Integration with Legacy HART”. In: *Technical Report* 1.2010 (2010).

[Sen+13] U. Sendler et al. *Industrie 4.0*. Berlin, Germany: Springer, 2013.

[SGL20] G. Stergiopoulos, D. A. Gritzalis, and E. Limnaios. “Cyber-Attacks on the Oil & Gas Sector: A Survey on Incident Assessment and Attack Patterns”. In: *IEEE Access* 8 (2020), pp. 128440–128475.

[Slo19] J. Slowik. *Evolution of ICS attacks and the prospects for future disruptive events*. Tech. rep. Hanover, MD, USA: Threat Intelligence Centre Dragos Inc, 2019.

[Sta94] I. O. for Standardization. *ISO/IEC 7498-1:1994 - Information technology — Open Systems Interconnection — Basic Reference Model: The Basic Model*. Standard. International Organization for Standardization, 1994.

[The+22] T. Thepmanee et al. “Implementation of control and SCADA system: Case study of Allen Bradley PLC by using WirelessHART to temperature control and device diagnostic”. In: *Energy Reports* 8 (2022), pp. 934–941.

[Wes+20] S. Wesemeyer et al. “Extensive Security Verification of the LoRaWAN Key-Establishment: Insecurities & Patches”. In: *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*. New York, NY, USA: IEEE, 2020, pp. 425–444.

A Digital Twin Based Industrial Automation and Control System Security Architecture

Abstract

The digital twin is a rather new industrial control and automation systems concept. While the approach so far has gained interest mainly due to capabilities to make advanced simulations and optimizations, recently the possibilities for enhanced security have got attention within the research community. In this paper, we discuss how a digital twin replication model and corresponding security architecture can be used to allow data sharing and control of security-critical processes. We identify design-driving security requirements for digital twin based data sharing and control. We show that the proposed state synchronization design meets the expected digital twin synchronization requirements and give a high level design and evaluation of other security components of the architecture. We also make performance evaluations of a proof of concept for protected software upgrade using the proposed digital twin design. Our new security framework provides a foundation for future research work in this promising new area.

Christian Gehrmann, Martin Gunnarsson “A Digital Twin Based Industrial Automation and Control System Security Architecture”. In *IEEE Transactions on Industrial Informatics*, vol. 16, no. 1, pp. 669-680, Jan. 2020, IEEE.

1 Introduction

Industrial Automation and Control Systems (IACS) is a very broad term covering everything relating to control, monitoring and production in different industries and encompasses all parts of such systems.

While security for IACS in the past was neglected, in recent years security has obtained a lot of attention in the research community and indeed within the industry. Major security incidents such as the STUXNET worm in 2010 [FMC11], the Shamoon Saudi Aramao spear-phishing attack in 2012 [Ley12] and the German steel factory attack in 2014 [Rob14] have highlighted the risk of attacks on IACS. Even if the attacks have been of many different types and origins, they have highlighted the need for enhanced security mechanisms and countermeasures.

Clear evidence that the industry nowadays takes security issues seriously is the development of best practice security guidelines [PH+11] and the large number of security standards targeting the IACS domain, like ISO/IEC 27000 series[1], the ISA/IEC IEC 62443 series[2] and the NIST SP800 series. Among those, IEC 62443 is based on the very general ISO 27000 but specified for the IACS area and also the NIST SP 800-82 [15] in the SP800 series is an IACS standard. In addition, the industrial internet consortium has developed a new security framework [Sch+16a].

New technology trends affect IACS as well as the entire society. Security solutions, security recommendations as well as standards, need to adapt to the new technologies. One clear current trend is the move from legacy ISA-95 to highly distributed and cloud based architectures according to the Industry 4.0 and RAMI 4.0 models [LBK14]. This transition is demanding in many ways, one challenge is control and information sharing between production units and cloud based control functions. This constitutes a major security risk and requires careful system engineering not to jeopardize IACS reliability [Del17]. We tackle this general security issue in this paper by looking into the digital twin model as an *enabler* for enhanced security when opening up IACS low level control functions and data exchange according to the Industry 4.0 vision. Digital twins and state replication as security enablers were recently proposed by different researchers [Bit+18a],[EE18a], [EE18b]. Previous works have not taken an IACS holistic view and in this paper we look into the problem from a system security point of view. The work is focused on identifying main design driving requirements for a digital twin based IACS security architecture and with special attention to a state synchronization model fulfilling the requirements. Detailed design of the different components and protocols in the architecture as well as formal security analysis of these are left for future work. The main contributions of the paper are the following:

- We introduce a digital twin IACS adversary model and identify security requirements for this model.

[1] https://www.iso.org/isoiec-27001-information-security.html

[2] ISA, ISA99, Industrial Automation and Control Systems Security, https://www.isa.org/isa99/

- We suggest a novel digital twin based security architecture including a new state replication model.
- We evaluate the security of the proposed state replication model as well as present a proof of concept implementation for a PLC software upgrade case including performance figures.

We proceed as follows: we discuss the digital twin model and make basic definitions which we use throughout the paper (Section 2), we introduce our adversary model and derive security requirements (Section 3), we suggest a new digital twin security architecture and a novel digital twin design, including a state replication model (Section 4). We make a security analysis of the proposed model and architecture (Section 5) and present a proof of concept implementation, including performance figures (Section 6). Lastly we discuss related work (Section 2.2) and conclude (Section 7).

2 Digital twin concept, related work and definitions

2.1 Digital twin model and scenario

The digital twin was according to Grives [Gri14a], a terminology invented around 15 years ago by John Vickers of NASA and the term was introduced publicly by NASA in 2010 [Sha+10]. Originally, the concept was used to refer to the digital representation of a product used in simulations software but has been expanded to a concept where not only a physical product is represented in virtual form (software) but each product is directly connected with a virtual counterpart, the digital twin. The general model is depicted in Fig. 1.

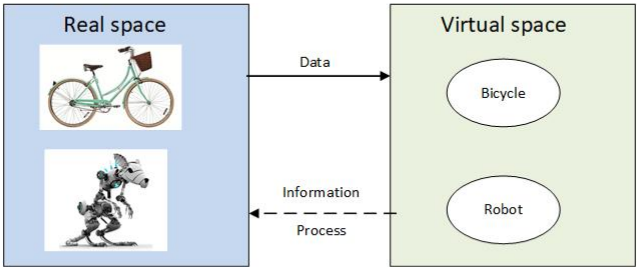


Figure 1: The original digital twin model.

The overall goal with this concept is to be able to closely follow products during production (the physical twin) and simulate the process to adjust the production with results of these simulations. This can be done in real-time or close to real-time to optimize production flows etc. [Uhl+17]. The concept has then been extended to include all units (robot loading stations, conveyor belts etc.) in a production system allowing advanced simulations of a complete manufacturing system and the units involved in an autonomous system [Ros+15]. Typically, then the digital twin part is represented and executed on cloud resources [Sha+18]. Fig. 2 illustrates the overall scenario and model.

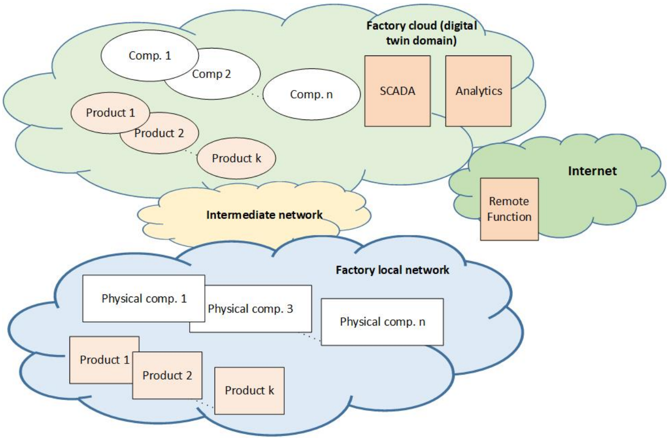


Figure 2: Digital twin cloud system scenario.

As can be seen in Fig. 2, according to this model not only are the products themselves reflected as digital twins in the virtual (cloud) domain but also the manufacturing units or what we here refer to as ”components”. Typical components here are PLCs, historians, sensors, actuators data acquisition units, HMI units etc. Several different models and principles for reflecting such units are possible [EE18b]. Here we focus on the network and logical state of a physical twin rather than the physical properties. The definitions and notations we use are introduced in Section 2.3 below.

2.2 Related work

Lots of work has been devoted to security in IACS. We will here briefly discuss literature surveys and how our architecture relates to the main security issues previously identified. Next, we will discuss some important previously introduced digital twin models and their relations to our approach. We mainly focus on prior work devoted to digital twin and state replication as enablers for enhanced security.

Security in IACS in general has been treated in several good surveys [KG13; Uch+17]. The work by Krotifil and Gollmann [KG13] focusing on different types of attacks on existing systems but also concluding that most efforts so far have been devoted to IDS. Many existing IDS are compatible with our suggested architecture but it has the benefit that such systems can be deployed in the virtual domain. A very broad systematic overview of security in cyber-physical systems in general (including IACS) is given in by Humayed *et al.* in [Hum+17a]. The authors identify that major security challenges in IACS are change management (including SW update) as well as the ability to handle legacy systems. Both these issues are tackles with the architecture we proposed in this paper. In addition, as we discussed in the introduction, several existing standards and new standard initiatives, are addressing IACS security in current and future systems. None of the main standardization bodies have so far been working with the digital twin concept as an enabler for enhanced security.

State machine replication has a very long history. Most of the work in this domain has been devoted to *fault tolerance* [Lam78; Sch90]. Achieving state replication under the assumption of fault is much more demanding than the security oriented state replication we consider in this paper. We use a different, simpler model, allowing to choose the correct level of state reflection on the digital twin side depending on the security needs (see our state replication model in Section 4.2). This is justified by the fact that the design goal of a digital twin security system is disparate from a fault tolerance system, as the digital twin cannot replace the physical twin if it fails, but is there to reflect the physical twin and protect it from direct, potential hostile, external interactions.

The digital twin model was first introduced in [Sha+10]. Lots of work has then been devoted to the topic in resent years and good overview is given in [NFM17]. The main focus has been on support of health analysis and improved maintenance as well as digitally mirroring the life of the physical entity. We are following the second approach but different from prior the majority of prior art, we are focusing on using the digital twin as an enabler for enhanced security.

The usage of digital twins for penetrations testing is discussed in a recent work by Bitton *et al.* [Bit+18a]. The author investigate the relation behind a penetration test specification and system realization with focus on system cost optimization. A non-linear programming solution to find an optimal digital twin implementation level needed to perform certain security analysis tasks is presented. This is an approach that also is applicable to the sub-problem of digital twin realization in a system realizing the security architecture we present in this paper.

In [GA16a] the idea of using state synchronization as an IoT security enabler was suggested. However, the model presented in [GA16a] does not cover state changes on the IoT device side and no complete digital twin state synchronization model is given. Most recently, a digital twin security framework was presented in [EE18b] and later extended in [EE18a]. In [EE18b], a digital twin specification principle using Automation ML (AML)[3] was described together with a proof of concept implementation detecting a man-in-the-middle PLC attack. In the follow up work, [EE18a], also the state replication problem is considered. In this work, a passive state replication model is presented where state updates are purely done based on inputs in the physical domain. The strength with such a model is that it avoids the negative performance impacts of active state monitoring. Inspired by

[3] Actually, automation ML for digital twin modelling was already suggested by Grecyce *et al.* in 2016 [Sch+16b] but not for any security applications.

the work in [EE18b] and [EE18a], we have also looked into the problem area of state modeling as security enabler. However, different from the work in [EE18a], we are looking into how digital twin can protect IACS from *external* attacks and not attacks on the factory domain. With this goal, we have proposed a different state propagation model and a security design allowing to identify attacks at the virtual domain and preventing them for even reaching the physical domain. Furthermore, we have analyzed a complete digital twin system scenario and proposed an overall security architecture for such scenario.

2.3 Digital twin definition and notations

For the purpose of the paper we denote by $u \in U$, a physical twin, where U denotes the set of physical twins in the system. Similarly, we denote by $u' \in U'$, a digital twin where U' is the set of digital twins in the system. Let then $S_u = \{s_{u0}, s_{u1}, ..., s_{um-1}\}$ and $S_{u'} = \{s_{u'0}, s_{u'1}, ..., s_{u'n-1}\}, m \geq n$, be the finite set of states of u and u', i.e., we assumes that the digital twin always only reflects a *subset* of the physical twin states and no states which are not represented in the physical twin. Furthermore, denote by $I_u = \{i_{u0}, i_{u1}, ..., i_{ur-1}\}$ the set of possible finite inputs to physical twin u and by $I_{u'} = \{i_{u'0}, i_{u'1}, ..., i_{u'd-1}\}$, the set of finite possible inputs to digital twin u'. We denote by $s_{u,t} \in S_u$, the state of physical twin u at clock cycle t and by $i_{u,t} \in I$ the input to u at clock cycle t. Similarly, denote by $s_{u',t} \in S_{u'}$, the state of digital twin u' at clock cycle t and the input to u' at clock cycle t by $i_{u',t} \in I'$. Hence, the initial state of the physical twin is $s_{u,0}$ and the initial state of the digital twin is $s_{u',0}$. Then we can define both the physical and digital twin as finite state machines. We then let $\delta_u : S_u \times I_u \rightarrow S_u$ and $\delta_{u'} : S_{u'} \times I_{u'} \rightarrow S_{u'}$ be the transition functions for the physical and digital twin respectively, i.e. $s_{u,t+1} = \delta_u(s_{u,t}, i_{u,t})$ and $s_{u',t+1} = \delta_{u'}(s_{u',t}, i_{u',t})$.

We assume a clock based digital twin state synchronization model where a each clock cycle, t, the state of the twins are synchronized with a message exchange starting with a first synchronization message from the u' to u and with a response synchronization message from u to u'. We denote these message as $m_{u' \rightarrow u}(t)$ and $m_{u \rightarrow u'}(t)$, respectively. These messages are typically not transferred in clear between the twin and intermediate nodes, but in protected/transformed form. We denote protected version of the synchronization messages by $e_{u' \rightarrow u}(t)$ and $e_{u \rightarrow u'}(t)$.

3 Adversary model and security requirements

Next, using the digital twin model and definition introduced in Section 2, we describe a digital twin threat model. Using this threat model we identify security requirements for a digital twin based IACS architecture.

3.1 Adversary model

Adversary models for digital twin systems have not been extensively treated in the literature as the concept mostly so far has been used for production optimization and not security. Certain security aspects regarding using digital twin as security enablers in IACS are considered in [EE18a] and [Bit+18a]. The authors in [EE18a] consider state replication for active monitoring and intrusion detection while [Bit+18a] consider the problem of penentration testing of IACS with focus on cost optimization for specific security penetration tests (performed on simulated or emulated digital twin or on an acutal physical component in the IACS). However, since these works have very specific security functions goals, they lack adversary model definitions for the digital twin scenario we are considering. Hence, we have developed a new adversary model below. This is *not* a generic digital twin adversary model but a model that makes sense in systems with cloud based data sharing and control in IACS. We also give the main motivations for using this rather restrictive adversary model.

Traditionally, IACS has been separated with firewalls from other networks such as corporate network and the internet. Several good architectures and recommendations are available [Sch+16a]. Here, we assume such principles are deployed and we have adopted an adversary model where we *do not* consider any attacks on the physical twin part or local factory network part of the system but assume these parts can be properly isolated from hostile external networks[4].

We assume that the digital twin can run in a separate process even on a third party cloud resource. Then the digital twin can be realized using virtualization techniques where the virtualization is offered on the most suitable level [SN05]. Providing strong isolation for virtualization and protection against hostile cloud providers is a very challenging topic which has been widely addressed with several different models and solutions the past ten years [Liu+15; Sch+15; PGM17]. Recent attacks Metldown [Lip+18] and Spectre [Koc+19] have shown that one cannot even trust the fundamental hardware functions needed for secure isolation currently in use. However, the security with respect to secure execution environment for virtualized systems is steadily improving and we will for simplicity in this paper disregard attacks on the isolation properties of the digital twin and assume that a secure execution environment and data storage is provided for the digital twin in the system.

We adopt the Dolev-Yao model [DY81] and assume that the attacker can influence the system in all other aspects including the following capabilities of the adversary:

- The attacker is able to intercept, modify and replay all communication from the physical domain to the digital domain and vice versa.

[4] Internal factory network attacks are of course also possible, but we do not consider those in our adversary model.

- The attacker is able to launch input attacks by sending arbitrary messages to a digital twin and input requests, i.e. he or she can choose to send arbitrary input from the set $I_{u'}$ to the digital twin u'.
- The attacker is able to launch intercept, modify and replay any information sent between digital twins or between digital twins and other units executing in the virtual domain.

3.2 Security definitions

Next, we give basic security definitions. The basis of the new security architecture is the introduction of state replication between the physical and digital twin. An expectation from such model from *robustness* perspective is that the synchronization is accurate over all system states and inputs. The synchronization consistent expectation is fundamental for deploying the architecture and very different from architectures introduced in the literature before. The main reason why consistency is important is that without it, one cannot rely on that all system changes in the digital twin part are correctly propagated to physical part of the system and vice versa, which will make it impossible to use the model in practise as the system behaviour would be unreliable. Hence, even if the synchronization consistency not is a pure security requirement, it is fundamental for the proposed architecture and we make a precise definition of synchronization consistent ency. It is also important to notice that one would expect from a specific design and implementation of our architecture to provide the synchronisation consistency property also under attack conditions. Hence, it is important to introduce a proper definition also in this regard.

Another fundamental, pure security expectation, with respect to the synchronization is the confidentiality and integrity of the synchronization process as such. Hence, we also provide precise definitions for these two aspects. Apart from these definitions, we adopt widely used computer and communication security definitions [SB14].

Definition 3.1. A digital twin system is *consistent* if there exist functions $\forall u \in U, f_u : S_u \to S_{u'}$such that the following is true:

$$\forall s \in S_u, f_u(\delta_u(s, \emptyset)) = \delta_{u'}(f_u(s), \emptyset), \quad (1)$$

$$s_{u',0} = f_u(s_{u,0}). \quad (2)$$

This definition reflects the requirement that when the digital twin starts in a state consistent with the staring state of the physical counterpart and whenever neither the physical twin nor the digital twin receive any input, they are both always transitioned to states that are consistent. i.e. the physical to digital twin state mapping agree with the state of the digital twin.

Definition 3.2. A digital twin system synchronization protocol provides *confidentiality protection* if an adversary, who observes information, $e_{u'\to u}(t)$ and $e_{u\to u'}(t))$, sent from the digital twin and from the physical twin respectively at time t, cannot execute any attack, A, that in polynomial time will allow the attacker to distinguish the state of the physical twin from any randomly selected state, i.e., after execution of A, the following is true:

$$\forall s \in S_u, Pr(s_u = s | e_{u'\to u}(t), e_{u\to u'}(t)) = Pr(s_u = s) \tag{3}$$

Definition 3.3. A digital twin system synchronization protocol provides *synchronization protection* if the adversary cannot execute any attack replacing message exchange $e_{u'\to u}(t)$ with $e'_{u'\to u'}(t)$ and/or replacing $e_{u\to u'}(t)$£ with $e'_{u\to u'}(t)$ which will be accepted by u and u' and making the twins out of synchronization, i.e. $f_u(s_{u,t}) = s_{u',t}$ is always true after successful synchronization independent of adversary substitution choices[5].

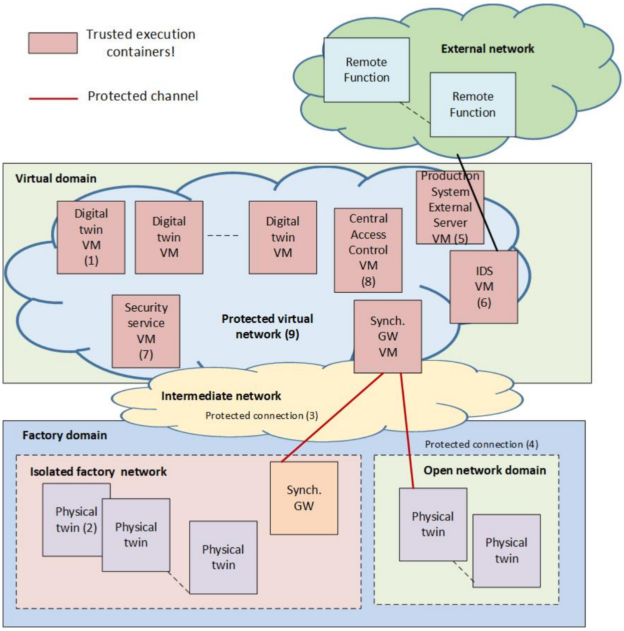


Figure 3: Security architecture overview.

3.3 Requirements

We have used the previously presented adversary model and security definitions to identify a set of system security, performance and accuracy requirements. This is

[5] This definition does *not* take a DoS attack into account and assumes that the synchronization messages arrives at each time slot.

not an exhaustive list but the major identified system architecture requirements.

R1. **Synchronization security:** We require the digital twin state replication model and protocol to be consistent (Definition 3.1), provide confidentiality protection (Definition 3.2) and synchronization protection (Definition 3.3).

R2. **Synchronization latency:** The synchronization message exchange must not cause any delays which prevent time critical control functions to be propagated to from the physical to the digital twin. The precis requirements are application dependent.

R3. **Digital twin external connections protection:** All connections between the digital twin and the external entities must be authenticated. According to the adopted adversary model, we assume each digital twin to run in a protected execution environment but all request external to this environment must be properly authenticated and all information sent from the digital twin to external trusted parties must be confidentiality and integrity protected.

R4. **Access control:** The digital twin itself or a secure entity in direct connection with the digital twin needs to make sure access control is applied on on all incoming requests. This includes request and information exchange with external parties as well as information exchange with other digital twins.

R5. **Software security:** The physical twin software must always be in a trustworthy state. This implies that the physical twin must be protected from installation of harmful software. Mechanisms shall be in place to recover the system in case of zero-days attacks on the physical twins.

R6. **Local factory network isolation:** The local factory network shall not accept any connection requests except for protected synchronization requests with the digital twin (see R1 above). Physical twins should be protected from DoS attacks through boarder unit such as a gateway or firewall making sure that only protected synchronization requests reach a physical twin and no other outside traffic.

R7. **Digital twin Denial-of-Service (DoS) resilience:** The digital twin must be protected from DoS attacks such as network flooding or distributed DoS directly targeting a digital twin. Proper DoS filters and router configurations must be deployed in the factory cloud domain to prevent or limit the DoS possibilities of the attacker. At the same time, filters and router policies must not prevent synchronization exchanges to reach the digital twins in the system.

4 A digital twin based security architecture and state replication design

4.1 Security architecture

We now have the definitions and requirements in place to define a generic digital twin security architecture. Fig. 3 gives a high-level picture of the proposed architecture. We have here focused on the main security properties and entities in the system. This is *not* a complete design in all details but a high level design including main components and their roles in the architecture. We verify the key digital twin design of it in our proof of concept evaluation but leave detailed design and evaluation of other components for future work.

A basic security assumption in this architecture is the possibility to launch digital twins as well as security services in *trusted execution containers* as Virtual Machines (VMs) on suitable cloud resources. The architecture is completely agnostic on the virutalization technique used for this or on which actual level the vitalization is applied [SN05] [GA16a]. However, the architecture requires the virtualization technology to provide trusted execution in the sense that different VMs are strongly isolated from each other and that they have access to protected volatile and non-volatile storage.

Using the numbering introduced in Fig. 3, we discuss the different properties of the components in the architecture below.

Digital twin component

The digital twin component is running as a VM in an isolated environment. An overview picture of the main logical functions of the twin is given in Fig. 4. The core functionality of the digital twin is the actual simulation of the physical counterpart. Only two direct external network interactions are allowed: the synchronization (which occurs over the synchronization GW) and the exchange with external requests and responses. This takes place either through the cloud server which takes *all* incoming requests and responses from external entities or directly to other digital twins or back-end components. The virtual domain external connections are protected through the cloud Virtual Private Network (VPN) (see Section 4.1). The state of the digital twin is *exported* directly to a *common* (for several digital twins in a system) security analysis component (see also Subsection 4.1). Also the intermediate state, $\hat{s}_{u'}$, is exposed to an analyzer in this way. This implies that an external analyzer can have access (if allowed by the access policy) to all digital twin states in the system. This in turn allows *abortion* of state propagation in case of detection of a fatal security issue by the external analyzer. The digital twin has access to a secure clock, t, for precise synchronization operations with the physical twin. The actual state propagation design we use is described in Section 4.2.

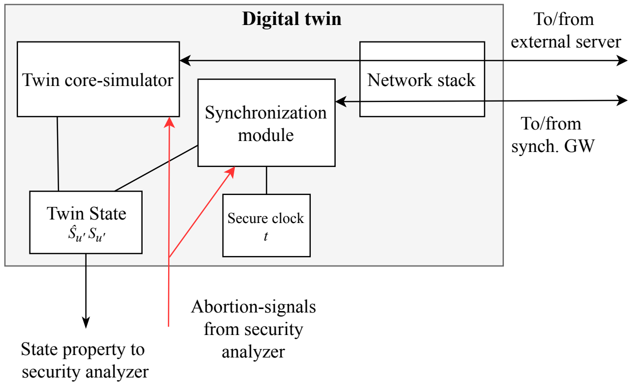


Figure 4: Digital twin main functions

Physical twin component

An overview picture of the main logical functions of the physical twin is given in Fig. 5. Similar to the digital counterpart, the physical twin executes the defined synchronization protocol. Depending on if the physical twin actually has network connectivity or not, it might run the synchronization itself or it is done through a ”measurement unit”[6]. A physical twin deployed in an isolated factory network will only exchange synchronization information with a dedicated synchronization GW on the same network. On the other hand, a single deployed physical twin outside such a network will need to directly exchange synchronization information with the synchronization GW in the virtual domain and needs access to the key material needed for such secure interactions. The physical twin will apart from this, not need any specific security adaptations at all. The state propagation design applicable to the physical twin is described in Section 4.2.

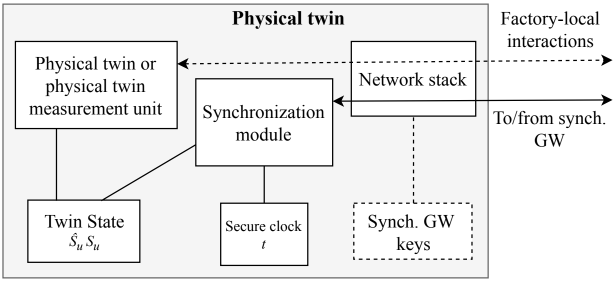


Figure 5: Physical twin main functions.

[6] For a physical twin that is in production, it could be that it has no program execution capabilities, but its state is only measured through external sensors for instance.

Protected connection between synchronization gateways

The connection between the synchronization GW on the local factory and the virtual domain is protected through a secure channel. We have chosen this principle instead of end-to end synchronization protection as we assume it will be possible to deploy synchronization GWs in trusted containers in the virtual domain. Standard IPsec [KS05] VPN or a TLS/DTL channels [DR08a] [RM12] are assumed. A major advantage with such solution from security management point of view is that this allows a *single* security relation between the physical and digital domain. Such single relation is very easy to maintain from security perspective. For instance, can a pres-shared key TLS or DTLS relation for instance be used. This can be compared to a situation where external entities are allowed to directly connect to the physical domain. In such situation, each external connection would need a separate security relation with the physical domain. Now, such relations are instead moved to the digital domain, where the security risk is much lower and where it is much less complex to handle such relations from a security configuration management point of view.

Protected connection from isolated physical twin to synchronization gateway

A physical twin not deployed in a protected local factory network, needs to directly connect to the synchronization GW in the virtual domain. This connection then obviously needs to be confidentiality and integrity protected using a suitable secure channel (see Section 4.1).

Production system external server

The architecture assumes all external requests arrives in the virtual domain, i.e. external input to digital twin u' from the set $I_{u'}$ arrives to the production system external server prior to (potential) being forwarded to the digital twin u'. Similar responses from a digital twin are routed through this sever as well. This allows advanced network filtering at a single point and avoids having such functionality duplicated at each digital twin virtual instance[7].

Intrusion Detection System (IDS)

State-of-the art IDS are best deployed at the boarder to the internet [Kru+02]. We adopt this principle and assume the actual intrusion analysis to be done by a VM with direct access to the external network interface traffic.

[7] Recall that in our adversary model we assume all inputs to a *physical* twin to be trustworthy and not subject to direct security analysis

Security analysis service

The core benefit from a security perspective with a digital twin model like the one we have defined, is the possibility to do security analysis directly on the digital twin state and even on the states of a whole family of digital twins. By letting the analyzing engine having access not only to the final states, but also intermediate states, i.e. the $\hat{s}_{u'}$ states in the system, it is possible for a security analysis function to detect harmful state transitions (prior to the state propagating to the physical twin) and take direct action in the digital domain (see also Fig. 4).

Central access control

By letting all external digital twin access be subject to a single point access control, system wide policies can easily be deployed in the system. Advanced security policies can be defined through standard access control frameworks such as Extensible Access Control Markup Language (XACML) [Ris13]. In order to allow direct interaction between digital twins, this is preferably combined with component local policy enforcement through tokens issued at the central access control entity using standard tokens such as SAML [CMJ15] or OAuth [Har12].

Protected virtual network

Most cloud providers offer network isolation between VMs launched on cloud resources[8][9]. Even if we have not assumed all trusted execution services to be deployed as complete, ”traditional” VMs in the virtual domain, higher layer VMs can be launched on such VMs allowing re-use of standard principles for network isolation. There are also other, non-provider dependent solutions to achieve this [LW10].

4.2 State replication model and design

Several different state replication principles for digital twins are possible. Recently, a specification-based state replication model for digital twins was proposed [EE18a]. We have adopted a similar physical and digital twin state transition model. However, the state replication design in [EE18a] is built upon measurement of input values and that the physical and digital twin runs *functional identical programs* or what the authors refers to as ”passive state replication”. This is an approach that is efficient if the main purpose of the design is to evaluate security breaches stemming from the physical domain. Instead, we in our security architecture use the digital twin as a ”guard” against all, potential hostile, *external* stimuli on the physical domain. Hence, even if demanding from real-time perspective, we instead have adopted a direct state replication or what the authors

[8] https://docs.aws.amazon.com/vpc/index.html#lang/en_us

[9] https://docs.microsoft.com/en-us/azure/virtual-network/virtual-networks-overview

in [EE18a] refers to as "active monitoring". This different security goal and approach also allow us to abandon the functional identical program requirements. We assume a model, where the physical and digital twin are synchronized on regular basis. Without loss of generality, we assume that a synchronization is done at each clock cycle. Let $z_u : S_u \times S_{u'} \to S_u$ be a synchronization function and $h_u : S_u \to S_{u'}$ a physical to digital state mapping function for twin u. The complete synchronization (including the twins state updates) then consists of the following operations:

$$\begin{aligned} \hat{s}_{u,t+1} &= \delta_u(s_{u,t}, i_{u,t}), && (4) \\ \hat{s}_{u',t+1} &= \delta_{u'}(s_{u',t}, i_{u',t}), && (5) \\ s_{u,t+1} &= z_u(\hat{s}_{u,t+1}, \hat{s}_{u',t+1}), && (6) \\ s_{u',t+1} &= h_u(s_{u,t+1}). && (7) \end{aligned}$$

This synchronization model works such that the physical and digital twin treat their respective inputs independently. We assume that the input will change the state of the (respective) twins independently, and then at the next time slot, they will synchronize their states to make them consistent considering the inputs received before last synchronization.

The choice of the functions z_u and h_u will depend on the digital twin model and the exact relation between the physical and digital twin. Many different models are possible. For the purpose of this paper, we choose a simple twin model but still a model allow to cover several important security cases as we show in Section 6. Denote by $S_u = S1_u \bigcup S2_u \bigcup S3_u$, we then make the following assumption: $S1_u \bigcap S2_u = S1_u \bigcap S3_u = S2_u \bigcap S3_u = \emptyset$. Let $S_{u'} = S1_{u'} \bigcup S2_{u'}$ and we assume that $S1_{u'} \bigcap S2_{u'} = \emptyset$. Then we can write the state of the physical twin as $s_u = (s1_u, s2_u, s3_u)$ and the state of the digital twin as $s_{u'} = (s1_{u'}, s2_{u'})$. We then apply the following restrictions:

$$\begin{aligned} S2_u &= S1_{u'}, && (8) \\ S3_u &= S2_{u'}, && (9) \end{aligned}$$

$$\begin{aligned} &\forall s_u \in S_u, \forall i_u \in I_u, \delta_u(s_u, i_u) = \\ &= (\delta 1_u(s_u, i_u), \delta 2_u((s2_u, s3_u), i_u), s3_u), && (10) \\ &\forall s_{u'} \in S_{u'}, \forall i_{u'} \in I_{u'}, \delta_{u'}(s_{u'}, i_{u'}) = \\ &= (s1_{u'}, \delta 2_{u'}(s_{u'}, i_{u'})) && (11) \end{aligned}$$

In addition, we let

$$s_{u',0} = (s1_{u',0}, s2_{u',0}) = (s2_{u,0}, s2_{u',0}), \qquad (12)$$

$$s_{u,0} = (s1_{u,0}, s2_{u,0}, s3_{u,0}) = (s1_{u,0}, s2_{u,0}, s2_{u',0}) \tag{13}$$

With these restrictions, we then let $z_u(\hat{s}_{u,t}, \hat{s}_{u',t}) = (\hat{s}1_{u,t}, \hat{s}2_{u,t}, \hat{s}2_{u',t})$ and

$$h_u(s_{u,t+1}) = \begin{cases} (s2_{u,0}, s3_{u,0}) & \text{if } t < 0 \\ (\hat{s}2_{u,t+1}, \hat{s}2_{u',t+1}) & \text{otherwise} \end{cases} \tag{14}$$

To send the *complete* state at each synchronization occasion is very inefficient. Instead, the state changes (deltas) are calculated:

$$\begin{aligned} m_{u' \to u}(t) &= \Delta_{\hat{s}_{u'}} = \text{Diff}(\hat{s}_{u',t+1}, s_{u',t}), & (15) \\ m_{u \to u'}(t) &= \Delta_{s_{u'}} = \text{Diff}(\hat{s}2_{u,t+1}, s2_{u,t}), & (16) \end{aligned}$$

This implies that the digital twin calculates a first delta, $\Delta_{\hat{s}_{u'}}$, and sends it to the physical twin. This delta is then used by the physical twin to reconstruct $\hat{s}_{u',t+1}$, which is the input to the z function, i.e. equation (6). Next, the physical twin calculates the "return delta", $\Delta_{s_{u'}}$, that is sent back to the digital twin. The principle is illustrated in Fig. 6 below. Observe, that we here only illustrate the synchronization information exchange and not the protection of the synchronization messages as such. The protection principles we apply was described in Section 4.1.

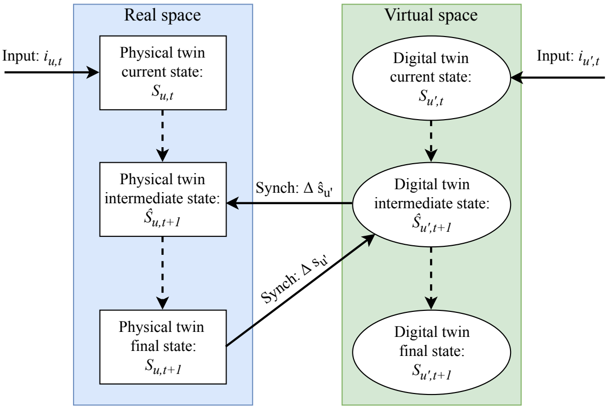

Real space
Virtual space
Input: $i_{u,t}$
Physical twin current state: $S_{u,t}$
Digital twin current state: $S_{u',t}$
Input: $i_{u',t}$
Physical twin intermediate state: $\hat{S}_{u,t+1}$
Synch: Δ $\hat{s}_{u'}$
Digital twin intermediate state: $\hat{S}_{u',t+1}$
Synch: Δ $s_{u'}$
Physical twin final state: $S_{u,t+1}$
Digital twin final state: $S_{u',t+1}$

Figure 6: Synchronization principle.

It is important to notice from real-time and communication overhead perspectives that when no input is received neither on the physical or digital side, there is no need for the twins to exchange any deltas. This is true given a consistent digital twin system synchronized with accurate clocks.

5 Security analysis

Next, we analyze the proposed framework from security and performance perspectives. We here mainly focus on the synchronization security characteristics. We also give arguments regarding how the proposed architecture meets the other security requirements listed in Section 3.3. As the architecture in many aspects only include a high level design, we here postpone detailed security evaluation of these aspects to future work and for specific implementation designs.

Synchronization security

Proposition 1. The digital twin synchronization model and protocol is consistent.

Proof. Let:

$$f_u(s) = f_u((s1, s2, s3)) = (s2, s3). \tag{17}$$

From (13) we have that $s_{u,0} = (s1_{u,0}, s2_{u,0}, s2_{u',0})$ and from (12) and (14), it then follows that $f_u(s_{u,0}) = h_u(s_{u,0}) = (s2_{u,0}, s3_{u,0}) = (s1_{u',0}, s2_{u',0}) = s_{u',0}$, which fulfils condition (2).

Now, using the assumptions (8), (9), (10) and (11), let:

$$\hat{\delta}_u(s_u, i_u) = (\delta 1_u(s_u, i_u), \delta 2_u((s2_u, s3_u), i_u), \delta 2_{u'}((s2_u, s3_u), \emptyset).$$

Then, it follows from (17),

$$f_u(\hat{\delta}_u(s_u, \emptyset)) = (\delta 2_u((s2_u, s3_u), \emptyset), \delta 2_{u'}((s2_u, s3_u, \emptyset)).$$

Similar, let: $\hat{\delta}_{u'}(s_{u'}, i_{u'}) = (\delta 2_u((s1_{u'}, s2_{u'}), \emptyset), \delta 2_{u'}(s_{u'}, i_{u'}))$.

Then by direct calculation:

$$\hat{\delta}_{u'}(f_u(s_u), \emptyset) = \hat{\delta}_{u'}((s2_u, s3_u), \emptyset) =$$
$$(\delta 2_u((s2_u, s3_u), \emptyset), \delta 2_{u'}((s2_u, s3_u), \emptyset)) = f_u(\hat{\delta}(s_u, \emptyset)).$$

By then letting the state s_u taking any value in S_u, it follows that also condition (1) is fulfilled. □

Proposition 2. If the secure channel used for communication towards and between synchronization GW in the architecture provides confidentiality, the digital twin synchronization design also provides confidentiality.

Proof. According to our attacker model, an adversary can intercept any message sent from the digital twin to the synchronization GW in the virtual domain or any messages sent between synchronization GWs. He or she might also intercept message sent from physical twins towards the GW deployed in the virtual domain. The attacker has no other option to intercept any synchronization information. According to (15) and (16), at each clock cycle, one delta message is sent from the digital twin towards the physical twin and a replay delta message is sent in return. An adversary has two options to intercept the first message, $e_{u' \to u}(t)$; Either he or she intercept it when it is sent from the digital twin the synchronization GW in the virtual domain *or* when it is forwarded from the synchronization to the GW in the factory domain (or physical twin in the second option). As long as both these channels provide confidentiality the attacker will not get any information on s_u. As the return message follows the very same path, the also the return message, $e_{u \to u'}(t)$, will have the very same protection and equation (3) is fulfilled. □

Proposition 3. If the secure channel used for communication towards and between synchronization GW in the architecture provides integrity and replay protection, the digital twin synchronization design also provides synchronization protection.

Proof. According to Proposition 1 the proposed synchronization model is consistent and consequently if no input is received on neither the digital nor physical twin, $h_u(s_{u,t+1}) = s_{u',t+1}$. Furthermore, if the synchronization messages also arrives unmodified equation (7) guarantees that $h_u(s_{u,t+1}) = s_{u',t+1}$ holds also in this case. Hence, the only option for an attacker would be to modify any messages $e_{u' \to u}(t)$ or $e_{u \to u'}(t))$. In analogue with the proof of Proposition 2, if the used secure channels provides integrity and replay protection, such modification will be detected and a modified or replayed message will be rejected. □

Latency

The architecture as such does not make any direct assumption regarding the synchronization real-time behaviour. Depending on the specific IACS application, the networks must be chosen and configured accordingly. Similarly, the synchronization GW must be implemented on platforms powerful enough to fulfill real-time requirements. For some applications, deploying the virtual domain on an edge cloud [Del17] can be used to meet R2.

External connections

The architecture assumes all external connection to be intermediates by the external server entity at the boarder of the external network. The external server will only accept authenticated requests. Furthermore, the final hop for the external server to the digital twin runs through the virtual domain VPN. This, if properly implemented, implies that the system fulfills the requirement R3.

Access control

According to the proposed security architecture, the centralized access control VM deployed in the virtual domain makes sure all access requests towards the digital twin are properly authorized. Access control enforcement then takes place at the digital twin VM. This means that the main building blocks are included to fulfil R4. However, the actually authorization and access control mechanisms which are supported are subject to detailed design, which have been left for future work.

Software security

The software state of the physical twin can be replicated to the digital counterpart. A security service with direct access to the twin state can be launched. This service then controls the physical twin software state and upgrade. This is a very efficient way to both monitor the SW status and control upgrades as we show with the experimental evaluation in Section Section 6. Even if this is an important step to meet R5, further SW monitoring tools needs to be deployed in the system to give the wanted software security level.

Network isolation and DoS resilience

The architecture adopts best practise for factory network isolation [15] to meet R6. In addition, external interaction with the factory domain is only possible indirectly through the protected synchronization. All direct requests towards digital twin are subject to IDS and filtering and additional security protection mechanism can be launched as security service VMs in the virtual domain. With proper design and implementation, such measures will provide network isolation and DoS resilience as required by R7.

6 Proof of concept and performance evaluation

In order to test the feasibility of the proposed architecture and approach, we have implemented a low complexity system with digital twins using our proposed state synchronization protocol. Our main goal here is to get an impression of how the proposed synchronization framework, which is the fundamental basis of the proposed architecture, affects the production units in the system as well as the bandwidth consumption[10]. It was argued in [EE18a] that direct state synchronization or what the authors refer to as ‘‘active monitoring" is not feasible in real-time critical systems due to large bandwidth overhead. While we argue that this is not the case for low complexity digital twin state models and for moderate synchronization frequencies, we are interested to measure the production unit actual computation and bandwidth overhead in a real system. To make the evaluation feasible, we here focus on the *first three* components in the architecture in Fig. 3. We have implemented a simple manufacturing scenario, as seen in Fig. 7 consisting of a PLC unit, u_1, controlling an industrial process. In addition, we have a software upgrade server, u_2, holding software upgrade information, that is deployed in the factory local network. The PLC and the upgrade server are reflected as digital twins: u_1', u_2'. The goal with introducing the virtual domain is to allow secure software control and upgrade of the production system units. To facilitate

[10] We recall that the synchronization including the protection of the synchronization is the only parts of the architecture that directly affects the production domain.

this, the software state and software control state are replicated to the digital twin domain.

It should be noted, that additional components and more complex production scenarios, will give a more detailed picture of how the proposed synchronization model effects the system performance. However, as the proposed synchronization protocol scales linear with the number of units with respect to bandwidth consumption, we argue that measurements in a small systems will give a good view of the overall system impact. Furthermore, the actual effect in terms of computational overhead on a particular production unit, will obvious depend on the computational power of the unit. Here, we use a fairly constrained platform, a RaspberryPI, for the evaluation. Other platforms and systems will be affected in similar ways but obviously platforms with less resources will be affected more. How, different platforms with different resources are affected, is left for future work as our main goal here is to verify the general feasibility of the approach.

Our proof-of concept implementation shows that as long as we have moderate state changes and the synchronization happens less than 100 times a second, clock synchronization is not an issue. The platform we have worked with can timely process a request and send a response without major delays. Hence there is no need to have a more precise clock synchronization. Here we let the digital twin act as a "master" and the physical twin as a "slave" unit at each synchronization occasion.

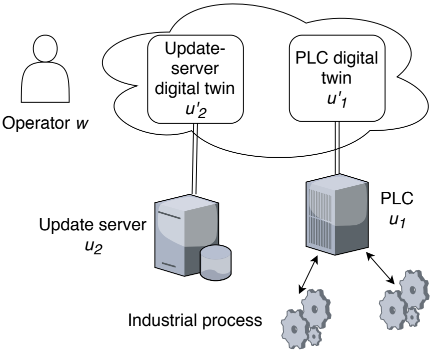


Figure 7: Setup of out digital twin and software update scenario.

The state information for the supported twins are selected to be: $s_{u'_1} =$ [ctrl_flag, ctrl_url, sw_state] and $s_{u'_2} =$ [ctrl_url, sw_package][11]. ctrl_flag is a value holding software upgrade request control and error information and the ctrl_url is a URL of a new software package to be installed. sw_state is a list of all current software packages

[11] Here is actually no state information with origin from the physical twin, u_2, but just digital twin state information which is propagated to the physical twin.

and versions installed on a unit and sw_package is a new software package. We also assumes a remote operator, w, to be present in the system controlling software upgrades through a remote user device over standard internet.

6.1 PLC software update process

w identifies a new software package, q, and connects to the external server u'_2. w then downloads q to u'_2 and w receives a ctrl_url value for the package in return. u'_2 then updates the state $\hat{s}_{u'_2,0}$ to reflect the storage of the new software package. Then a synchronization takes place between u'_2 and u_2. The synchronization is done by sending $\Delta_{\hat{s}_{u'_2}} =$ ctrl_url$+q$ from u'_2 to u_2. This in turn, triggers u_2 through the functions h_{u_2} and z_{u_2}, to update its internal state, resulting in the storage of q which can be downloaded from ctrl_url to other units within the local factory network.

w makes a second request using the newly received ctrl_url and with information regarding the new software packages towards u'_1. The request trigger u'_1 to update states $s_{u'_1,1}$: ctrl_flag, ctrl_url, sw_state, where ctrl_flag contains ”available software update indicator”, ctrl_url contains the URL to the new software package on u_2 and sw_state contains version information for the pending new software. In the clock cycle 2, this information is propagated to u_1 through $\Delta_{\hat{s}_{u'_1}}$. This values in combination with the functions h_{u_1} and z_{u_1} give an updated state $s_{u_1,2}$. The SW update flag in state $s_{u_1,2}$ triggers u_1 to set the state to update pending allowing to u_2 using ctrl_url to download and install the new SW package, q. Once, the update is finalized, the update status information as well as the new SW state information is propagated back to u'_1 through updates of the ctrl_flag and sw_state.

6.2 Performance evaluation

We have implemented the scenario, described above, with a SW update process using digital twins. As the PLC u_1 we have used OpenPLC[Alv+14], a free, open source PLC implementation, running on a RaspberryPI[12]. The Raspberry Pi we have used is a model 2 v1.1 with an ARM Cortex-A7 quad-core processor, clocked at 900MHz.

The digital twins u'_1, u'_2 are running as separate processes in a Ubuntu 18.04 desktop host. The same host also functions as the update server u_2. Since the physical entities synchronize with digital-twins outside the protected factory network the synchronization protocol is secured by DLTS.

[12] https://www.raspberrypi.org

Update time depending on synchronization frequency

In order to evaluate the state synchronization protocol we have looked at the SW update scenario. We want to examine how the state synchronization process affects other processes running on the system.

First we ran tests without state synchronization to establish a base line for how long time the update process takes. Then we ran the SW update process with state synchronization at different frequencies. We evaluated performance at 1, 10 and 100 state synchronizations per second. The result can be seen in Fig. 8[13].

As can be seen from the figure the performance impact of the state synchronization is very small. Only at a large number of synchronizations per second is the performance noticeable.

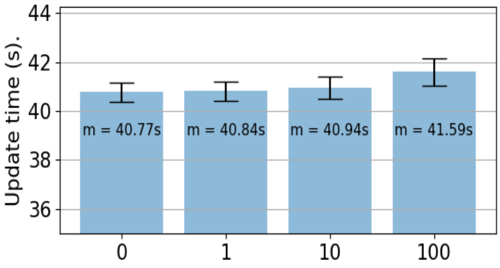


Figure 8: Update times when using state synchronization at different frequencies.

Compassion of DTLS Cipher Suites

We have compared different DLTS cipher suites to evaluate if this impacts performance. The default strong suite AES-256-GCM with SHA384 was compared to the weaker AES-128-GCM with SHA256. The results can be seen in Figure 9. It can be noted that the choice of ciphers has only a very small impact on the performance of the update process.

Computation cost

A PLC is not a constrained device in a traditional sense, however, since it controls a time-critical process CPU-time is limited. Any added features must consider this so time-critical deadlines are kept.

We have measured the CPU-time needed by the PLC to implement our state synchronization protocol. By running the protocol over an extended time we have come to the following numbers as seen in Table 1.

[13] In the simple system we are using, actually, the state exchange can be omitted in most cases as we very seldom have state changes, but in our evaluation, we anyway forced a state exchange to take place in order to test the synchronization frequency performance impact.

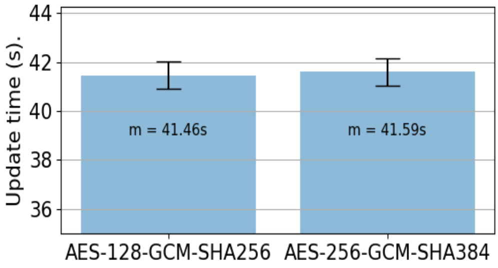


Figure 9: Comparison of update times with different DTLS cipher-suites.

As shown in the table the CPU-time needed by the PLC to implement the state synchronization protocol is very small. An even slower CPU will still be able to run the state synchronization without overloading the processor.

Table 1: Measurements of CPU-time per state synchronization message and CPU-load.

CPU-time (ms) per synchronization	CPU-load 10 synchronizations/s	CPU-load 100 synchronizations/s
0.3772 ($s = 0.0602$)	0.0038%	0.0377%

Network performance

Evaluating network performance for the state synchronization process is difficult to do without real ICS network traffic to base an evaluation scenario on. Hence, instead we evaluated the performance in an isolated system. We measured the bandwidth consumption for the PLC during the update process. We then measured the bandwidth for the update process while synchronizing with the PLC's digital-twin. The synchronization messages were of size 22 bytes in each direction. The bandwidth consumption can be seen in Table 2. As can be seen from the Table the bandwidth consumption is reasonable for small synchronization frequencies.

Table 2: Bandwidth to and from the PLC when updating.

	Bandwidth to PLC	Bandwidth from PLC
No synch	0.97 KB/s	2.06 KB/s
1 synch/s	1.20 KB/s	2.38 KB/s
10 synch/s	2.16 KB/s	3.35 KB/s
100 synch/s	10.88 KB/s	12.06 KB/s

7 Conclusion and future work

Motivated by the need for new security models and principles in IACS to open up the systems for cloud based processing and data sharing, we investigated how digital twins can work as a security enablers in IACS. We introduced a new adversary model, made basic security definitions, identified security requirements, made a novel security architecture and in particular state replication design for a digital twin based IACS. The new state replication design as well as the architecture were then security evaluated against the identified requirements. We showed that the proposed synchronization design meets the introduced digital twin synchronization requirements. Furthermore, we made a high-level design of the other security components in the architecture and argue about how the suggested functions will help in meeting the identified security requirements. Through our proof of concept implementation and performance evaluation, we also showed that the new digital twin synchronization model works well in practice for a small but real production case with reasonable performance impact. Especially, we show that as long as we have not too high update frequency, the performance impact on a platform like RaspberryPI is negligible. As expected, the bandwidth increases linear with the synchronization frequency. In our evaluation, we only reflected a few PLC states, and obviously, the more fine grain states that are reflected, the more impact it will have on the system performance and bandwidth consumption.

The results shows that a digital twin based security architecture can be a promising way to protect IACS while open them up for external data sharing and access. We have here worked with defining a suitable overall architecture and synchronization model. In order to develop a fully working system based on our architecture and approach, more work is needed. Below, we discuss the most important future work:

- **Performance:** We have here made first proof of concept of the architecture. In order to see the effect of the architecture on different platform and production scenarios, more performance evaluations on different platform, with more complex digital twin state models and with larger amount of production nodes are needed.

- **Intrusion detection:** In our security architecture, we have only show how on principle level how to integrate intrusion detection at the boarder to the virtual domain. It is left for future research to design and integrate intrusion detection in a fully working system.

- **Access control:** The architecture allows for advanced access control in the virtual domain. The main advantage with this approach is that this can be supported without affecting the production domain at all. It remains to design and evaluate this approach in a full system implementation of the architecture.

- **Formal security analysis:** We have proven the consistency of the proposed synchronization protocol and showed that the security of the protocol depends on the security of the underlying used secure channel. Formal analysis of the security of the complete system design and all protocols are left for future work.

- **Security analysis services:** Apart from IDS and access control enforcement in the virtual domain, additional security analysis services may be supported as virtual components as we showed in our architecture design. This include services such as virus scan, DoS prevention etc. The design and evaluation of such services is left to future research as well.

Acknowledgements

We would like to thank the SSF SEC4FACTORY project team for valuable discussions regarding the research direction and results presented in the paper. In particular we would like to thank the TetraPak project members for their valuable feedback and suggestions.

References

[15] *Guide to Industrial Control Systems (ICS) Security*. NIST Special Publication 800-82, 2, Version 2. 2015.

[Alv+14] T. R. Alves et al. "OpenPLC: An open source alternative to automation". In: *IEEE Global Humanitarian Technology Conference (GHTC 2014)*. Oct. 2014, pp. 585–589.

[Bit+18a] R. Bitton et al. "Deriving a Cost-Effective Digital Twin of an ICS to Facilitate Security Evaluation". In: *Computer Security*. Ed. by J. Lopez, J. Zhou, and M. Soriano. Cham: Springer International Publishing, 2018, pp. 533–554.

[CMJ15] B. Campbell, C. Mortimore, and M. Jones. *Security Assertion Markup Language (SAML) 2.0 Profile for OAuth 2.0 Client Authentication and Authorization Grants*. RFC 7522. May 2015.

[Del17] J. Delsing. "Local Cloud Internet of Things Automation: Technology and Business Model Features of Distributed Internet of Things Automation Solutions". In: *IEEE Industrial Electronics Magazine* 11.4 (Dec. 2017), pp. 8–21.

[DR08a] T. Dierks and E. Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.2*. RFC 5246 (Proposed Standard). Internet Engineering Task Force, Aug. 2008.

[DY81] D. Dolev and A. C. Yao. "On the Security of Public Key Protocols". In: *Proceedings of the 22nd Annual Symposium on Foundations of Computer Science*. SFCS '81. Washington, DC, USA: IEEE Computer Society, 1981, pp. 350–357.

[EE18a] M. Eckhart and A. Ekelhart. "A Specification-based State Replication Approach for Digital Twins". In: *Proceedings of the 2018 Workshop on Cyber-Physical Systems Security and PrivaCy*. CPS-SPC '18. Toronto, Canada: ACM, 2018, pp. 36–47.

[EE18b] M. Eckhart and A. Ekelhart. "Towards Security-Aware Virtual Environments for Digital Twins". In: *Proceedings of the 4th ACM Workshop on Cyber-Physical System Security*. CPSS '18. Incheon, Republic of Korea: ACM, 2018, pp. 61–72.

[FMC11] N. Falliere, Murchu, and E. Chien. *W32.Stuxnet Dossier*. Symantec Security Response online report. Feb. 2011.

[GA16a] C. Gehrmann and M. A. Abdelraheem. "IoT Protection through Device to Cloud Synchronization". In: *2016 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*. Dec. 2016, pp. 527–532.

[Gri14a] M. Grieves. *Digital Twin Manufacturing Excellence Through Virtual Factory Replication*. Dassault Syst'emes. Paris, France, 2014.

[Har12] D. Hardt. *The OAuth 2.0 Authorization Framework*. RFC 6749. Oct. 2012.

[Hum+17a] A. Humayed et al. "Cyber-Physical Systems Security–A Survey". In: *IEEE Internet of Things Journal* 4.6 (Dec. 2017), pp. 1802–1831.

[KG13] M. Krotofil and D. Gollmann. "Industrial control systems security: What is happening?" In: *2013 11th IEEE International Conference on Industrial Informatics (INDIN)*. July 2013, pp. 670–675.

[Koc+19] P. Kocher et al. "Spectre Attacks: Exploiting Speculative Execution". In: *2019 2019 IEEE Symposium on Security and Privacy (SP)*. Vol. 00. 2019, pp. 19–37.

[Kru+02] C. Kruegel et al. "Stateful intrusion detection for high-speed network's". In: *Proceedings 2002 IEEE Symposium on Security and Privacy*. May 2002, pp. 285–293.

[KS05] S. Kent and K. Seo. *Security Architecture for the Internet Protocol*. RFC 4301 (Proposed Standard). Internet Engineering Task Force, Dec. 2005.

[Lam78] L. Lamport. "Time, Clocks, and the Ordering of Events in a Distributed System". In: *Commun. ACM* 21.7 (July 1978), pp. 558–565.

[LBK14] J. Lee, B. Bagheri, and H.-A. Kao. “A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems”. In: *SME Manufacturing Letters* 3 (Dec. 2014).

[Ley12] A. Leyden. *Hack on Saudi Aramco hit 30,000 workstations, oil firm admits*. 2012.

[Lip+18] M. Lipp et al. “Meltdown: Reading Kernel Memory from User Space”. In: *27th USENIX Security Symposium (USENIX Security 18)*. Baltimore, MD: USENIX Association, 2018, pp. 973–990.

[Liu+15] C. Liu et al. “ObliVM: A Programming Framework for Secure Computation”. In: *2015 IEEE Symposium on Security and Privacy*. May 2015, pp. 359–376.

[LW10] L. E. Li and T. Woo. “VSITE: A scalable and secure architecture for seamless L2 enterprise extension in the cloud”. In: *2010 6th IEEE Workshop on Secure Network Protocols*. Oct. 2010, pp. 31–36.

[NFM17] E. Negri, L. Fumagalli, and M. Macchi. “A Review of the Roles of Digital Twin in CPS-based Production Systems”. In: *Procedia Manufacturing* 11 (2017). 27th International Conference on Flexible Automation and Intelligent Manufacturing, FAIM2017, 27-30 June 2017, Modena, Italy, pp. 939–948.

[PGM17] N. Paladi, C. Gehrmann, and A. Michalas. “Providing User Security Guarantees in Public Infrastructure Clouds”. In: *IEEE Transactions on Cloud Computing* 5.3 (July 2017), pp. 405–419.

[PH+11] F. M. P. Didier P, J. Harstad, et al. *Converged Plantwide Ethernet solution - Converged Plantwide Ethernet (CPwE) design implementation guide*. Cisco Systems and Rockwell Automation. 2011.

[Ris13] E. Rissanen, ed. *eXtensible Access Control Markup Language (XACML) Version 3.0*. OASIS Standard. 2013.

[RM12] E. Rescorla and N. Modadugu. *Datagram Transport Layer Security Version 1.2*. RFC 6347. Jan. 2012.

[Rob14] P. F. Roberts. *Cyberattack inflicts massive damage on German steel factory*. The security ledger. 2014.

[Ros+15] R. Rosen et al. “About The Importance of Autonomy and Digital Twins for the Future of Manufacturing”. In: *IFAC-PapersOnLine* 48.3 (2015). 15th IFAC Symposium onInformation Control Problems inManufacturing, pp. 567–572.

[SB14] W. Stallings and L. Brown. *Computer Security: Principles and Practice*. 3rd. Upper Saddle River, NJ, USA: Prentice Hall Press, 2014.

[Sch+15] F. Schuster et al. “VC3: Trustworthy Data Analytics in the Cloud Using SGX”. In: *2015 IEEE Symposium on Security and Privacy*. May 2015, pp. 38–54.

[Sch+16a] S. Schrecker et al. *The industrial Internet of Things - Volume G4: Security Framework*. Industrial Internet Consortium. 2016.

[Sch+16b] G. N. Schroeder et al. “Digital Twin Data Modeling with AutomationML and a Communication Methodology for Data Exchange”. In: *IFAC-PapersOnLine* 49.30 (2016). 4th IFAC Symposium on Telematics Applications TA 2016, pp. 12–17.

[Sch90] F. B. Schneider. “Implementing Fault-tolerant Services Using the State Machine Approach: A Tutorial”. In: *ACM Comput. Surv.* 22.4 (Dec. 1990), pp. 299–319.

[Sha+10] M. Shafto et al. *Modeling, simulation, information technology & processing roadmap*. National Aeronautics and Space Administration (NASA). 2010.

[Sha+18] M. R. Shahriar et al. “MTComm Based Virtualization and Integration of Physical Machine Operations with Digital-Twins in Cyber-Physical Manufacturing Cloud”. In: *2018 5th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2018 4th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom)*. June 2018, pp. 46–51.

[SN05] J. Smith and R. Nair. *Virtual Machines: Versatile Platforms for Systems and Processes (The Morgan Kaufmann Series in Computer Architecture and Design)*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2005.

[Uch+17] P. Uchenna et al. “Review of cybersecurity issues in industrial critical infrastructure: manufacturing in perspective”. In: *Journal of Cyber Security Technology* 1.1 (2017), pp. 32–74.

[Uhl+17] T. H.-J. Uhlemann et al. “The Digital Twin: Demonstrating the Potential of Real Time Data Acquisition in Production Systems”. In: *Procedia Manufacturing* 9 (2017). 7th Conference on Learning Factories, CLF 2017, pp. 113–120.

Secure Ownership Transfer for Resource Constrained IoT Infrastructures

Abstract

Internet of Things or IoT deployments are becoming more and more common. The list of use-cases for IoT is getting longer and longer, but some examples are smart home appliances and wireless sensor networks. When IoT devices are deployed and used over an extended time, it is not guaranteed that one owner will control the IoT devices over their entire lifetime. If the ownership of an IoT system shall be transferred between two entities, secure ownership transfer arises.

In this paper we propose a protocol that enables secure ownership transfer of constrained IoT devices. The protocol is resource-efficient and only rely on symmetric cryptography for the IoT devices. The protocol has been rigorously analyzed to prove the state security requirements. The security analysis has been done partially using formal protocol verification tools, particularly Tamarin Prover. To show our proposed protocol's resource efficiency, we have done a proof of concept implementation. This implementation, for constrained IoT devices, has been used to verify the efficiency of the protocol. The results presented in this paper, an an extend version of previously published work on secure ownership transfer protocols for constrained IoT devices by the same authors.

Martin Gunnarsson, Christian Gehrmann "Secure Ownership Transfer for Resource Constrained IoT Infrastructures". In *Information Systems Security and Privacy. ICISSP 2020. Communications in Computer and Information Science*, vol. 1545, pp. 22-47, Jan. 2022, Springer.

1 Introduction

Internet of Things or IoT is a relatively well-established term in computer science research and the IT industry. IoT is a concept or vision where *things* are connected to some network, usually the internet. Network connectivity enables connected devices to send and receive data and interact with other computing resources connected to the same network. The types of devices that have gained networking capability are, to name a few: industrial sensors and actuators, connected medical devices, and smart consumer devices. One application of IoT is vast deployments with many devices [Vög+16]; these deployments can be used to monitor large areas such as cities for thins such as pollution and noise levels. Large deployments of connected devices can be a challenge to manage for the owner of the devices and research on how to manage large IoT deployments [Lan+16] [DMR16].

Like any computing device network, a large IoT deployment must be managed and supervised to be secure. An IoT deployment is expected to facilitate secure communication between devices, confidentiality and integrity protection, etc. for the transmitted data. To enable this, each IoT device in the deployment needs keys. These keys need to be issued and updated [RNL11] to each IoT device. The problem of key management has been explored and investigated before, and protocols and standards exist that describe how key management can be done securely. Examples of such key management standards are: IKE [Ero+10] and HIP [SO12]. These standards are tried and proven for a scenario when one organization is managing devices over their entire life cycle.

However, if the IoT deployment's owner wishes to sell or otherwise transfer the entire deployment to a new owner, existing key management solutions are no longer sufficient. For IoT systems deployed into an environment such as a factory or a city, it might not be feasible to physically access each IoT device to reprogram the devices to change ownership. The process of transferring the ownership must be done remotely, without physical access to the individual IoT devices. It is also essential that the old owner have all access revoked after the new owners assume ownership of the devices. It should not be possible for the new owner to access any data, or decrypt any messages sent by the old owner before the ownership transfer. When deploying IoT devices, especially in large numbers, it is important that the cost of devices must be kept low. Cost constraints typically result in IoT devices with limited performance, or *constrained* IoT devices. The limited capability of these types of IoT devices make certain types of cryptography to resource intensive.

Secure ownership transfer has been studied in the research for IoT devices and also for RFID tags. Deployments of RFID tags can be used for inventory tracking and supply chain management. For these use-cases, with many RFID tags that switch owners, the problem of secure ownership transfer has been studied [Taq+18]. For IoT, the question of secure ownership transfer has been studied for several applications, such as medical IoT and smart home appliances. The proposed solutions for these applications do not work for our intended use case with a system of constrained IoT devices.

In this paper we provide our extended work on the problem of secure ownership transfer for constrained IoT deployments. This work is the extended version of our previous work [GG20]. In this paper we have extended the security evaluation, provide more results from an experimental evaluation and have included a more comprehensive overview of related work.

We started by analyzing the intended deployment scenario for our protocol, including trust assumptions for the different entities in the system. We have stated formal security requirements for a secure ownership transfer protocol in the domain of constrained IoT devices from these preliminaries and prior research. Next, we present a protocol, with a trusted third party, referred to as a ”Reset Server” (RS) in this work. We have performed a rigorous security analysis of the protocol we proposed. We have used formal protocol verification with Tamarin Prover to aid in proving the previously stated security requirements. In addition to proving that the protocol fulfills the security requirements, we have done a proof of concept implementation of our protocol to experimentally verify that the protocol performs as desired when deployed in a constrained IoT environment.

The contribution of this paper is as follows:

- We analyze the IoT infrastructure ownership transfer problem and conclude that it has similar but not equal security requirements than those identified in previous analyses of group ownership transfer for tags.
- We suggest a novel IoT infrastructure ownership transfer model and protocol. The protocol uses a Trusted third party (TTP) and only symmetric cryptography to facilitate secure ownership transfer of constrained IoT devices.
- We present a proof of concept implementation and performance evaluation of the proposed ownership transfer scheme.
- We make a security analysis of the proposed ownership transfer protocol using both Tamarin Prover and logical reasoning.

We proceed as follows: first, we introduce our system model in Section 2, identify security requirements, and give a problem definition in Section 3. In Section 4, we present our ownership transfer model and protocol design; we perform a security analysis of the proposed transfer protocol in Section 5. We then describe our proof-of-concept implementation, including performance benchmarks in Section 6. Finally, we present and discuss related work in Section 7 and conclude with a discussion of our contribution and future work in Section 8.

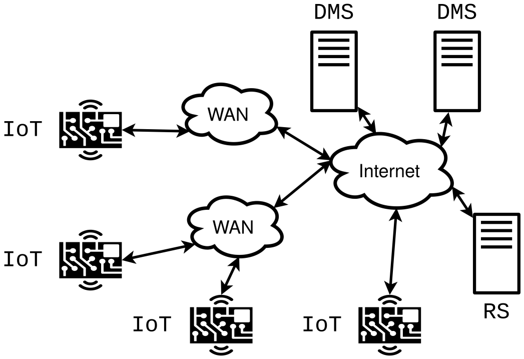


Figure 1: An overview of the considered system

2 System model and assumptions

In this paper, we consider IoT deployments, as seen in Figure 1. The IoT deployment consists of many IoT devices, deployed and managed by a Device Management Server (DMS) that is owned and operated by some entity. The entity will typically be a company but can also be an individual or an institution. The IoT deployment can serve a variety of purposes; it can, for example, be a part of an industrial control system, a building automation system, or a smart sensor network deployed to monitor the environment for pollution. The DMS is typically not located physically adjacent to the IoT devices, and communicate with the IoT devices through intermediary networks, typically the Internet. The last hop in communication

The IoT devices considered in this paper are typically constrained devices described in this document [BEK14a], which means that their computational capabilities, such as processing power and memory, are limited. In this paper, we assume that the IoT devices are capable of symmetric cryptography. Asymmetric cryptography is not feasible for these devices due to the complex computation needed. These computations consume energy and memory that is scarce on these types of constrained devices. The wireless communication technology available to constrained devices is usually Low-Power and Lossy Networks (LLNs) [Vas14]. As the name implies, LLNs have limited bandwidth, range and suffer from packet loss. These constraints restrict the amount of data that can be transmitted to and from the IoT devices. The DMS is assumed to be a computationally powerful server, either located on-premise in the entity owning and operating the IoT deployment or running in a cloud environment.

We show a schematic overview of a considered IoT deployment in Figure 1; the IoT units use LLNs or WAN to communicate locally, connectivity to the DMS is provided over the Internet. Since the IoT devices are connected to the Internet, they are vulnerable to remote attacks. Because of this threat of a remote attack, the

IoT units must be properly secured against these threats. The IoT devices must authenticate all incoming messages, and the DMS must authenticate all messages believed to originate from an IoT device. Since messages traverse networks that cannot be trusted not to eavesdrop on the communication, messages between the DMS and IoT devices must be encrypted. Thus, independent of the particular communications technology used, secure communication requires keys in place on the IoT units to mutually authenticate with the back-end DMS.

The protocol we propose uses several keys to perform secure ownership transfer. These symmetric keys are shared between individual IoT devices and the owner's DMS. In this paper we refer to these key(s) collectively as *credentials*.

Ownership transfer implies that the ownership is transferred from one organization to another. In this paper, we have assumed that another entity, be it another company, individual, or other organization that wishes to acquire ownership of the IoT deployment. We assume that the parties, in this paper called *old owner* and *new owner*, can agree on eventual payment and other compensation outside the scope of this protocol. Of more relevance for our protocol, we assume that the new owner has its own DMS and that the old owner and new owner can establish mutual authentication, possibly with a Public Key Infrastructure (PKI).

To facilitate secure ownership transfer using symmetric keys, we assume a trusted third party called the RS. The RS will aid in the deployment and ownership transfer of the IoT deployment. Since RS is a trusted third party, it is naturally trusted to a high degree by both new owner and old owner. In this paper, RS refers only to the server directly participating in the protocol. The organization operating the RS is left out of scope.

We suppose that the RS and DMS are servers of standard computational capabilities. There are no practical limitations in what types of cryptographic operations they can perform, specifically asymmetric cryptography. But, even if resources are abundant on the RS we want to keep operating costs and the computation and storage needed for the RS as low as possible. We also assume that the DMS servers and RS can exchange keys and authenticate each other, possibly with a PKI. Throughout this paper, the cryptographic functions used are assumed to be secure.

3 Adversary model and problem description

3.1 Adversary model

To enable structured and sound reasoning about our proposed protocol's security properties, we have chosen to use a model for the adversary and its capabilities. In this paper, we assume an adversary according to the Dolev-Yao model [DY81]. The attacker can intercept, delete, re-order, or modify all messages sent over any entity's communication channel. The adversary can also destroy messages but can not break cryptographic functions that are assumed secure.

We also assume that the IoT devices are placed in an environment where physical attacks from an insider are possible. One such attacker could be the current owner. The DMS and the RS are assumed to be in a secure location or in protected, isolated environments protected from external and insider software attacks.

Concerning direct physical attacks on the IoT units, we assume that an adversary and the old and new DMS can compromise, with a given effort, some or a limited number of IoT units through direct physical attacks on the devices. Here a compromised device refers to a device where the attacker has full control of the execution environment and volatile and persistent storage units of the device. Such a model is motivated by the fact that the needed effort for direct physical attacks is proportional to the number of compromised devices. Attacks from the current or new owner on a large scale can be difficult to perform in practice due to the location of the IoT devices or hardware protection mechanisms on the IoT units.

3.2 Trust model

The RS is assumed to be ”honest but curious” [Ode09]: The RS will be a legitimate and honest participant in the protocol execution, it will not deviate from the defined protocol, but will attempt to learn all possible information from legitimately received messages.

Realistically, there needs to be a certain level of trust between New owner and Old owner to transfer the IoT deployment ownership. It is, for example, difficult to imagine two companies with mutual distrust to do business. However, a company might have malicious insiders or might change its operating values and actions after a leadership change. Therefore, it is essential to design a protocol that assures both the New owner and the Old owner’s security and privacy, even if the other party is malicious. For the security analysis of our proposed protocol, the Old owner and the New owner are assumed not to trust each other; the Old owner is interested in learning the New owner’s secrets. Similarly, the New owner would like to learn the secrets held by the Old owner.

3.3 Requirements

We have started from the previously introduced adversary model for security protocols and have added general ownership transfer security requirements identified in previous work on RFID tags [Taq+18]. These security requirements for ownership transfer schemes for RFID tags have been adapted them to our system of IoT devices and our considered adversary model:

R1. **IoT unit impersonation security:** The protocol shall not allow an adversary to impersonate legitimate IoT units during or after the ownership transfer process.

R2. **Old DMS impersonation security:** The protocol shall not allow an adversary or the new DMS to impersonate the old DMS.

R3. **New DMS impersonation security:** The protocol shall not allow an adversary or the old DMS to impersonate the new DMS.

R4. **RS impersonation security:** The protocol shall not allow an adversary, any IoT unit or any DMS in the system to impersonate the RS.

R5. **Reply attack resistance:** The protocol shall be resistant against attacks where an adversary tries to complete sessions with any entities in the system by replaying old, observed messages.

R6. **Resistance to Man-in-the-Middle attacks** (**MitM**)**:** The protocol shall not allow insertion or modification of any messages sent between trusted entities in the system.

R7. **Resistance to de-synchronization attack:** The protocol should not allow the IoT units and the new or old DMS to enter a state where necessary secure communications is prevented by a credential mismatch.

R8. **Backward security:** During and after an IoT ownership transfer, the new owner shall not be given access to any secrets allowing the new owner to get access to any identities or confidential information used in past sessions between the old DMS and the IoT units.

R9. **Forward security:** During and after an IoT ownership transfer, the old owner shall not be given access to any secrets allowing the old owner to get access to any identities or confidential information used in sessions between the new DMS and the IoT units.

R10. **No double ownership:** There shall not be any time period during the ownership transfer process when both the old and the new owner has control over an IoT unit in the system.

In addition to these requirements, our adversary model does not imply full trust in the RS, and we also take into account the risk that IoT units might be compromised through attacks with direct physical access. These two assumptions result in the following additional two security requirements:

R11. **Protection of new credentials :** After completing the ownership transfer, the RS shall not know the new IoT credentials and shall not be able to set impersonate the new DMS or have access to secure sessions between the new DMS and the IoT units in the system.

R12. **IoT compromise resilience:** A successful compromise of an IoT unit by an external or internal adversary shall only give the adversary the power to impersonate this single IoT unit in the system and not impersonate or break any secure sessions between other, non-compromised IoT units in the system and the new DMS.

To make our proposed protocol usable for different types of IoT infrastructures, we must add more requirements. In some IoT deployments, the IoT units are connected to local networks, not publicly accessible, and only accessible by the owner system. For our purposes, this means that the current only DMS can access the IoT devices but not the RS. If a protocol was designed in such a way that RS needed to connect directly to the IoT devices, each IoT device would require a public IP-address. It would limit the suitability of our protocol for certain IoT deployments. Instead, by imposing this requirement on the protocol, our proposed protocol can be more general and fit for more IoT deployments. We add the following additional requirement to the system solution:

R13. **IoT unit isolation:** An ownership transfer shall not require any direct interactions between the IoT unit and the RS but only between the IoT unit and the DMS (old or new) in the system.

3.4 Problem statement

We define ownership as holding the credentials needed to authenticate to and securely communicate with the individual IoT devices. Each individual IoT device has credentials that it shares with a remote entity, i.e., its owner. The purpose of the protocol we propose is to transfer the ownership of a set of deployed IoT devices from the current owner to a new owner. The problem of ownership transfer then thus the process of updating credentials shared with the old owner to credentials shared with the new owner. We want to find an ownership transfer protocol that is secure under the specified adversary model and prove that the protocol fulfills the security properties stated in Section 3.3.

4 IoT infrastructure ownership transfer model and protocol design

In our solution, we divide the process of ownership transfer into three phases:

- Deployment
- Ownership transfer preparation
- Ownership transfer

In the deployment phase, the RS and the first owner of the IoT units provisions keys to the individual devices. The devices are then deployed and placed into the environment where they will be active.

In the ownership transfer preparation phase, the owner, from now on called *old owner*, and the entity that will assume ownership, from now on *new owner*, signs a list of all devices that shall be transferred and forwards this list to the RS. The RS verifies both signatures on the list of IoT devices. The RS distributes the keys needed for the ownership transfer and generates an ownership transfer token and the individual intermittent keys to the new owner.

In the final ownership transfer stage, the old owner receives the ownership transfer token from RS and forwards it to the IoT units. After receiving the token, each IoT devices verify the authenticity of the token. If the token is authentic, the IoT device decrypts it. In the token is information, such as IP-addresses and URL:s specifying how the IoT devices shall contact the new owner. The new owner and the IoT units can then mutually authenticate, and new credentials can be provisioned by the new owner to the IoT devices.

We describe the protocol in detail below, using terminology defined in Table 1. A schematic overview of messages transmitted between the entities in the protocol, the contents of the messages, and the computations done by each entity is illustrated in Figure 2. The steps from Figure 2 are references by bold numbers e.g. **(1.1)** in the subsections below.

4.1 Deployment

RS generates the keys KR_E, KR_M and K_{RS}. RS provides each IoT device with a unique identifier ID_i. K_{RS} is then used to generate a device unique key, $K_i = PRF(K_{RS}||ID_i)$, for each IoT_i. Each device IoT_i is provided with the corresponding KR_E, KR_M, ID_i and K_i. After transferring the keys RS can discard all keys K_i. RS sets its counter $Ctr_{RS} = 0$ and all IoT devices counters Ctr_i are also set to zero. These counters are used to verify the freshness of the ownership tokens later on. The first owner, DMS_{old}, takes control of the system and provides the owner-key $KO_i = \{KO_{i1}, KO_{i2}\}$ to each device IoT_i. The system is then ready for deployment and regular use, with KO_i used for securing the communication with DMS_{old}.

4.2 Ownership transfer preparation

The ownership transfer process starts with a preparation phase with interactions between the RS, DMS_{old} and DMS_{new}. DMS_{old} creates a list of all IoT device identities ID_i called **ID** and a list of identities and partial keys $\{ID_i, KO_{i2}\}$ called **ID-K** that shall switch owner **(1.1)**. The list of identities is signed $Sign(DMS_{old}, \mathbf{ID})$. Both lists are sent to DMS_{new} **(1.2)**, DMS_{new} first verifies the signature of the list, the list of identifiers are then signed by DMS_{new}. The result is $Sign(DMS_{new}, Sign(DMS_{old}, \mathbf{ID}))$, the list **ID-K** is kept by

Table 1: Notations used in protocol description. Originally published in [GG20].

DMS_{old}	Old Device Management Server
DMS_{new}	New Device Management Server
RS	Reset server
$Sign(P, d)$	Digital signature of data d by party P.
$E(k, m)$	Symmetric encryption of message m with key k.
$D(k, c)$	Symmetric decryption of ciphertext c with key k.
$MAC(k, m)$	Message Authentication Code of message m with key k.
$PRF(s)$	Pseudo-random function with seed s, generating a pseudo random key.
IoT_i	IoT device number i.
ID_i	Identifier of IoT device i.
ID_{new}	Identifier of DMS_{new}.
URL_{new}	Uniform resource locator to DMS_{new}.
K_i	Key for IoT device number i, shared with RS
KR_E	Reset-key used for encryption.
KR_M	Reset-key used for message authentication.
KO_i	Owner-key for IoT device number i, divided into two parts $KO_i = \{KO_{i1}, KO_{i2}\}$
K_{RS}	Master-key for RS used for deriving K_i.
N	Ownership-transfer nonce.
Ctr_i	Counter for device i is used for verifying freshness of nonces.
Ctr_{RS}	Counter for RS, incremented at every ownership transfer. Used for verifying freshness of nonces.
KS_i	Ownership transfer key for device i composed by: $KS_i = PRF(K_i \vert\vert N \vert\vert Ctr_{RS})$
T	Ownership-transfer token, calculated by: $T = E(KR_E, ID_{new} \vert\vert URL_{new} \vert\vert N \vert\vert Ctr_{RS} \vert\vert MAC(KR_M, ID_{new} \vert\vert URL_{new} \vert\vert N \vert\vert Ctr_{RS}))$
PSK_i	DLTS-PSK for IoT device i, generated by $PSK_i = PRF(KS_i \vert\vert KO_{i2})$
ID	List of IoT device identities **ID** $= \{ID_1, ID_2, ..., ID_i\}$
ID-K	List of pairs of IoT device identities and KO_{i2}: **ID-K** $= \{(ID_1, KO_{12}), ..., (ID_i, KO_{i2})\}$
K	List of keys K_i **K** $= \{K_1, K_2, ..., K_i\}$
KO	List of owner-keys KO_i, **KO** $= \{KO_1, KO_2, ..., KO_i\}$
KS	List of keys KS_i, **KS** $= \{KS_1, KS_2, ..., KS_i\}$
ID-KS	List of IoT device identities and keys: **ID-KS** $= \{(ID_1, KS_1), ..., (ID_i, KS_i)\}$

DMS_{new} (**1.3**). The list **ID** is sent to RS, to prove that ownership transfer shall take place and that both DMS_{old} and DMS_{new} are agreeing to the transfer (**1.4**). DMS_{new} also sends its identifier and URL to RS. After verifying that the list **ID** is correctly signed by both DMS_{old} and DMS_{new} (**1.5**), RS can start the ownership transfer protocol.

4.3 Ownership transfer

RS start the ownership transfer process by re-generating the keys K_i. A nonce N is generated, that together with Ctr_{RS} is used to generate the individual ownership transfer keys $KS_i = PRF(K_i||N||Ctr_{RS})$ (**2.1**). The list of ownership transfer keys **ID-KS** is sent to DMS_{new} (**2.2**). The RS creates the ownership transfer token T, with information needed by the IoT devices, authorizing an ownership transfer and information for how to do it. $T = E(KR_E, ID_{new} ||URL_{new}||N||Ctr_{RS}||MAC(KR_M, ID_{new}||URL_{new}||N||Ctr_{RS}))$
RS sends the token T to DMS_{old} (**2.3**). DMS_{old} forwards the Ownership Transfer Token T to all IoT devices (**2.4**). The devices decrypts T with KR_E and verifies the MAC with KR_M. If the MAC verification succeed, the freshness of the nonce is checked by verifying $Ctr_{RS} > Ctr_i$ (**2.5**). After these checks each IoT device IoT_i can compute the ownership transfer key $KS_i = PRF(K_i||N||Ctr_{RS})$ (**2.6**). With KS_i and KO_{i2} the IoT devices can connect to DMS_{New} using DTLS-PSK[TF16]. The parameters used are PSK-ID = ID_i and PSK = $PRF(KS_i||KO_{i2})$ (**2.7**). After a successful contact has been made with DMS_{new} IoT_i destroys KO_{i1} (**2.8**). DMS_{new} then generates a new key KO'_i (**2.9**). The new key KO'_i is sent to IoT_i, that also sets Ctr_i to the received value Ctr_{RS}(**2.10**). After DMS_{new} has provisioned new keys to all IoT devices the ownership transfer process is concluded. DMS_{new} can securely communicate with all IoT devices using the new keys KO'_i.

4.4 Handling of ownership transfer failures

In the previous sections we have described the ownership transfer process in detail. However, there is a risk that the ownership transfer succeeds for one set of IoT units but not for another set due to communication errors or similar. Such situation will be detected by the DMS_{New} as it will notice that it has not been able get in contact and authenticate some units part of the IoT transfer list given in step 1.2. DMS_{New} can first retransmit the ownership transfer token T to the devices that has not changed ownership. Some protocols provide a mechanism of notifying a sender that a message has been received. Such a mechanism can be used to verify the proper delivery of T. If T has been delivered but an IoT device still does not connect to DMS_{New} the issue lies with the device IoT_i, that situation will have to be resolved by DMS_{Old} before a new attempt can be made. In such situation, it is possible is for DMS_{New} to issue a ”recovery” procedure by sending a signed list of missing units back to DMS_{Old}, which then will be requested to contact

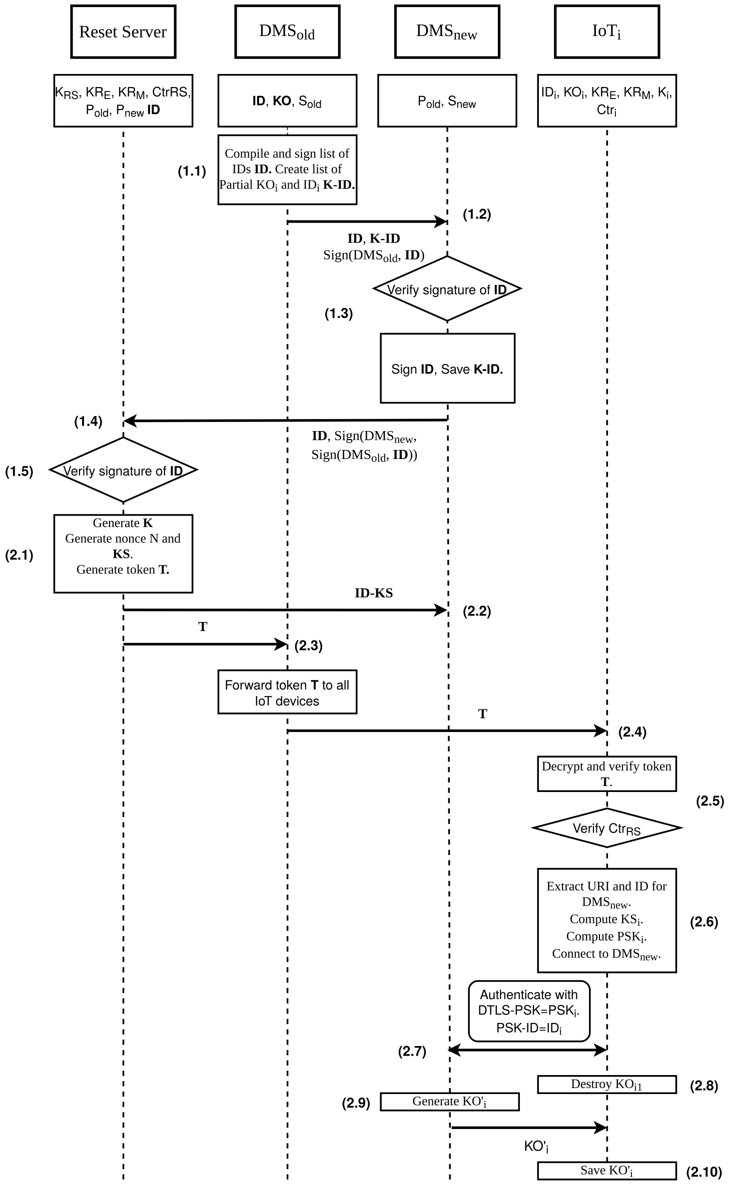

Reset Server
DMS$_{old}$
DMS$_{new}$
IoT$_{i}$
K$_{RS}$, KR$_{E}$, KR$_{M}$, CtrRS, P$_{old}$, P$_{new}$ **ID**
ID, **KO**, S$_{old}$
P$_{old}$, S$_{new}$
ID$_{i}$, KO$_{i}$, KR$_{E}$, KR$_{M}$, K$_{i}$, Ctr$_{i}$
(1.1)
Compile and sign list of IDs **ID.** Create list of Partial KO$_{i}$ and ID$_{i}$ **K-ID.**
(1.2)
ID, **K-ID**
Sign(DMS$_{old}$, **ID**)
Verify signature of **ID**
(1.3)
Sign **ID**, Save **K-ID.**
(1.4)
ID, Sign(DMS$_{new}$, Sign(DMS$_{old}$, **ID**))
(1.5)
Verify signature of **ID**
(2.1)
Generate **K**
Generate nonce N and **KS**.
Generate token **T.**
ID-KS
(2.2)
T
(2.3)
Forward token **T** to all IoT devices
T
(2.4)
Decrypt and verify token **T**.
(2.5)
Verify Ctr$_{RS}$
Extract URI and ID for DMS$_{new}$.
Compute KS$_{i}$.
Compute PSK$_{i}$.
Connect to DMS$_{new}$.
(2.6)
Authenticate with DTLS-PSK=PSK$_{i}$. PSK-ID=ID$_{i}$
(2.7)
Destroy KO$_{i1}$
(2.8)
(2.9)
Generate KO'$_{i}$
KO'$_{i}$
Save KO'$_{i}$
(2.10)

Figure 2: Messages and computations done during the ownership transfer. The figure is an updated version of a figure originally from [GG20]

.

each of the missing IoT units (still under ownership of DMS_{Old}) over a mutual authenticated DTLS channel re-sending the transfer token, T. Such procedure can be repeated, until the whole set of IoT units are successfully transferred to DMS_{New}.

5 Security analysis

We will now analyze our proposed ownership transfer protocol in the scope of the system model presented in Section 2 and the threat model from Section 3. We will address each requirement from 3.3 except R13 that is a functional requirement. We give special attention to the requirements R8, R9 and the requirement for PSK_i to be secure. We formally prove these requirements with Tamarin Prover[Bas+17]. The requirement to protect PSK_i from an outside adversary is important for requirements R1, R3 and R6 while backward (R8) and forward (R9) secrecy are a core features of the suggested protocol.

R1. **IoT unit impersonation security:** Each IoT unit i holds a unique key K_i. The nonce and counter in the token together with this key are used to calculate KS_i. In turn, KS_i and the second part of KO_i are used to calculate the PSK, used to authenticate the connection between the IoT unit and DMS_{New}. Both key parts needed to calculate the PSK are only known to DMS_{New} apart from the IoT unit as long as the RS and old owner do not collude, which contradicts the trust assumption regarding the reset server. Hence, given that the IoT unit itself can securely store and keep K_i, IoT impersonation is not possible for an external attacker or DMS_{Old}.

R2. **Old DMS impersonation security:** The ownership transfer is triggered by letting DMS_{Old} send a signed list of IoT identities (step 1.2). This signature is verified by the RS at step 1.5. As long as the signature scheme is secure and the private key of the DMS_{Old} not is compromised, an attacker cannot impersonate the DMS_{Old} at the ownership transfer ”triggering moment”. As we do not require protected transfer of the token (step 2.4), DMS_{Old} impersonation at this step is possible. However, it is not crucial for the protocol that it is indeed DMS_{Old} that sends the token but it can be transferred in arbitrary way, as the IoT unit does not finally accept the token unless the authentication in step 2.7 is performed successfully. The latter requires the genuine key KO_{i2} from old owner, and this key is sent protected to the DMS_{New} at step 1.2.

R3. **New DMS impersonation security:** Similar to the DMS_{Old}, DMS_{New} signs the list of IoT IDs subject to ownership transfer (step 1.3). This signature is verified by the RS at step 1.5. As long as the signature scheme is secure and the private key of the DMS_{New} not is compromised, an attacker cannot impersonate the DMS_{New} at the ownership transfer ”trigering moment”. Mutual authentication applies at step 2.7 when the IoT unit connects to the DMS_{New}. Impersonation at this step requires knowledge of the PSK, which (similar to the reasoning regarding R1 above), requires knowledge of both KS_i and KO_i, and if not the RS and old owner collude, these two values are only known to DMS_{New} and the IoT unit itself. Hence, DMS_{New}

impersonation is not possible unless DMS_{New} is compromised such that the secure keys leaks or the secure key transfers at step 1.2 or step 2.2 are broken. The latter is only possible if the mutually authenticated secure channel is weak.

R4. **RS impersonation security:** Only DMS_{New} and DMS_{Old} communicate directly with RS. They do so over a secure channel that protects against impersonation of RS.

R5. **Reply attack resistance:** All messages between RS, DMS_{Old} and DMS_{New} are sent over secure channels that provides protection against replay attacks (steps 1.2, 1.4, 2.2 and 2.3). The Token T transferred from DMS_{Old} to IoT_i (step 2.4) contains Ctr_{RS} that is verified against Ctr_i by IoT_i. This provides replay attack resistance since a replayed T will be rejected due to the counter check. When IoT_i connects to DMS_{New} (step 2.7) it is done with DTLS protected by PSK_i, which is only known to DMS_{New} and IoT_i. This DTLS channel is also used to protect the transfer of the new credentials KO'_i (step 2.10).

R6. **Resistance to Man-in-the-Middle attacks (MitM):** All messages between RS, DMS_{Old} and DMS_{New} are sent over secure channels that provides mutual authentication (steps 1.2, 1.4, 2.2 and 2.3) and thus prevents against MitM attacks. Communication with the IoT devices and DMS_{New} (steps 2.7 and 2.10) is done over DLTS-PSK that provides mutual authentication and with MitM protection. An attacker with knowledge of the keys KR_E and KR_M[1], can perform a successful man-in-the-middle substitution attack at step 2.4. Potential values to substitute are ID_{new}, URL_{new}, N or Ctr_{RS}. The IoT unit will not accept a wrong Ctr_{RS} as it is checked against the internal counter. Furthermore, substituted ID_{new} or N will not match the PSK values used in the mutual authentication in step 2.7 and the MitM substitution attack will fail. A substitution of URL_{new} will have no affect as long as the IoT unit still reach the legitimate DM_{new} with the given URL. If this not is the case,the ownership transfer for the affected unit will simple be aborted (see also the recovery discussion in Section 4.4).

R7. **Resistance to de-synchronization attack:** If DMS_{Old} should send a modified token, T' (through access to the keys KR_E and KR_M), with modified nonce N', in step 2.4, the key KS'_i will not match the key KS_i held by DMS_{New}. Hence, in this case, the IoT device will not remove the KO_i key, and will remain in the ownership of DMS_{Old}.

R8. **Backward security:** All traffic sent between the DMS_{Old} and the IoT devices is sent over a channel protected by the key KO_i, the IoT devices destroy

[1] These keys are included not to give protection against IoT compromise but to make denial-of-service type of attacks less likely.

KS_i when contact is made with DMS_{New}. DMS_{New} can not recover KO_i and is unable to learn any previous secrets (see also the Tamarin proof of Section 5.1).

R9. **Forward security:** After DMS_{New} has made contact with the IoT devices and the old key KO_i has been destroyed, DMS_{New} provisions a new key KO'_i and sends it to the IoT devices over a secure channel protected by the key KS_i that DMS_{Old} does not hold. DMS_{Old} is thus unable to decrypt any future message sent to the IoT devices (see also the Tamarin proof of Section 5.1).

R10. **No double ownership:** The ownership hand-over is made when the IoT device connects to DMS_{New} with PSK_i and removes ownership from DMS_{Old} by removing KO_i. DMS_{New} takes ownership when it provisions KS'_i to IoT_i. Failure in any protocol step might results in that some IoT units are still owned by the DM_{old}. However, as we discuss in Section 4.4 below, such situation can be detected by DMS_{New} and a recovery process can be initiated.

R11. **Protection of new credentials:** After the ownership transfer process IoT_i is provided with new credentials KO'_i. The only way RS can gain access to the system is by launching a MitM attack on the DLTS connection between IoT_i and DMS_{New}. Thus this property hinges on PSK_i, RS does not know KO_{i2} needed to derive PSK_i. As long as RS does not gain access to KO_{i2} by collusion with DMS_{Old}, the new credentials are protected (see also the Tamarin proof of Section 5.1).

R12. **IoT compromise resilience:** If an adversary compromises an IoT device IoT_i it will gain the following keys: KO_i, KR_E, KR_M and K_i. KO_i is only shared with the current owner and used for securing communication between the owner and IoT device, the adversary can not impersonate or compromise any other IoT device. KR_E and KR_M are shared with all IoT devices, an adversary could try to spoof an ownership transfer token T. Since the adversary only have KO_i it is impossible for the adversary to complete a malicious ownership transfer with an other IoT device IoT_j since the adversary does not know KO_j, thus providing resilience against compromises.

5.1 Tamarin Prover

Tamarin Prover[Mei+13] is a tool for formal analysis of security protocols. By creating a symbolic model of a protocol, stating security lemmas and then using the automatic reasoning to analyse the model the prover can show that the security lemmas hold or show a counter-example of when they do not hold. Tamarin represents protocols as a multi-set rewrite rules using first order logic. The automatic prover represent the state of the execution as a bag of multi-set of Facts. The adversary model used in Tamarin is the Dolev-Yao model. [DY83]. In the Dolev-Yao model the adversary is able to read, modify, replay and send any message to any participant in the system. One way of phrasing this, is to say that the adversary *is* the network itself.

5.2 Modeling the Ownership Transfer Protocol

We have modeled a simplified version of our proposed Ownership Transfer Protocol in Tamarin. We have excluded the steps 1.1 - 1.5 and 2.7 - 2.10 to prove the correctness of the core ownership transfer steps. During our process to verify the security of our proposed protocol we have introduced five lemmas. We have created one lemma, Protocol Correctness, to verify that our protocol can execute with a successful conclusion of the ownership transfer process. We have created another lemma, Outsider secrecy, to prove that PSK_i is secret from an outside adversary. The next two lemmas Old Owner Secrecy and New Owner Secrecy are lemmas about attacks done by a party in the protocol that misbehaves. These types of attacks are not included in a standard Dolev-Yao model. To solve this problem, we have chosen to give the Dolev-Yao adversary all keys and secrets from the malicious party. The Dolev-Yao adversary then has all the capabilities to intercept, replay and send any message together with the capability to decrypt, encrypt and sign messages with the keys from the malicious party. We argue that this is a stronger attacker than a real-world malicious Old owner or New owner would be. We have assumed that to provide New Owner secrecy PSK_i has to be kept secret from DMS_{Old}. To Provide Old Owner Secrecy the two keys KO_{i1} and KO_{i2} have to remain secret from DMS_{New}. For the Outsider Secrecy Property we state that no outside party can learn PSK_i. The last lemma is used to prove that the RS indeed will not learn the long term secret of the IoTs after the ownership transfer is completed. Our Tamarin model of our proposed protocol can be found here [2].

Below we list the five lemmas:

L1 **Protocol Correctness.** The modeled protocol shall execute as specified.

```
lemma protocol_correctness:
        exists_trace
        "Ex PSK1 PSK2 #i #j.
        (( New_owner_PSK(PSK1) @ #i) &
        ( IoT_PSK(PSK2) @ #j)) &
        (PSK1 = PSK2)"
```

L2 **Outsider secrecy.** The Ownership Transfer protocol shall be secure against outside attackers. No outside party shall be able to learn PSK_i.

[2] https://github.com/Gunzter/iot-ownership-transfer-protocol-tamarin-model

```
lemma outsider_secrecy:
        "All PSK #i #j.
        ( IoT_PSK(PSK) @ #i &
        New_owner_ownership_transfer_key(PSK) @ #j &
        not( Ex Old_owner #k. Reveal(Old_owner) @ #k ) &
        not( Ex New_owner #l. Reveal(New_owner) @ #l ) )
        ==>
        not(Ex #k. K(PSK) @ #k )"
```

L3 **Old Owner secrecy.** The New Owner shall not be able to learn anything that has been sent before the ownership transfer, thus KO_{i1} and KO_{i2} has to be secure against an adversary that knows everything DMS_{New} knows.

```
lemma backwards_secrecy:
        "All New_owner PSK #i #j #k.
        (IoT_PSK(PSK) @ #i &
        New_owner_ownership_transfer_key(PSK) @ #j &
        Reveal(New_owner) @ #k &
        not(Ex Old_owner #l.  Reveal(Old_owner)  @ #l ))
        ==>
        not( Ex OwnerKey1 OwnerKey2 #m #n. K(OwnerKey1)
          @ #m & K(OwnerKey2) @ #n)"
```

L4 **New Owner secrecy.** The Old owner shall not be able to learn anything that occurs after the ownership transfer is complete. No adversary that knows everything DMS_{Old} knows shall be able to learn PSK_i.

```
lemma forward_secrecy:
        "All Old_owner PSK #i #j #k.
        (IoT_PSK(PSK) @ #i &
        New_owner_ownership_transfer_key(PSK) @ #j &
        Reveal(Old_owner) @ #k &
        not( Ex New_owner #l. Reveal(New_owner) @ #l ))
        ==>
        not (Ex #m.  K(PSK) @ #m) "
```

L5 **RS secrecy from.** The RS shall not be able to learn anything that occurs after the ownership transfer is complete. No adversary that knows everything RS knows shall be able to learn PSK_i.

```
lemma secrecy_from_rs:
        "All RS PSK #i #j #k.
```

```
(IoT_PSK(PSK) @ #i &
New_owner_ownership_transfer_key(PSK) @ #j &
Reveal(RS) @ #k &
not(Ex Old_owner #l.  Reveal(Old_owner)  @ #l ) &
not(Ex New_owner #l.  Reveal(New_owner)  @ #l ))
==>
not( Ex OwnerKey1 OwnerKey2 #m #n. K(OwnerKey1)
  @ #m & K(OwnerKey2) @ #n)"
```

Using our modeled protocol we let Tamarin prove the five stated lemmas. All of them were found to hold. We conclude that our protocol fulfills the previously stated security properties.

6 Implementation and experimental evaluations

We have implemented our proposed protocol for an IoT environment running Contiki-NG[3]. Contiki-NG is a light-weight operating system designed for constrained devices. We have used some other protocols to structure our data. Most significantly we use COSE [Sch17] to encode and encrypt the ownership transfer tokens. We assume secure communication between the RS, DMS_{old} and DMS_{new}. The connections to the IoT devices are secured with DTLS[RM12]. Since SHA256 is assumed to be included on the IoT device from DTLS, we have selected HKDF-SHA256 as our key derivation function.

We have designed the system to use the REST-model[Fie00b]. Sending the ownership transfer token to the IoT device is done with a POST operation to /transfer-ownership. The IoT device then sends a GET message to /key to receive the new keys K_i' and KO_i'.

6.1 Test Setup

The evaluated scenario is executed on the following setup. One Desktop PC running the RS, DMS_{Old} and DMS_{New}. The PC is connected to a Border-Router that acts as an IEEE 802.15.4 network interface. We have used four Zolertia Firefly-A development boards[4] that are going to transfer from owner Old to New. The experimental setup is illustrated below in Figure 3. The IoT devices are based on the cc2538 system on chip made by Texas Instruments[Tex15]. They have an ARM Cortex-M3 CPU clocked at 32MHz together with 32KB of RAM and 512KB of flash. Connectivity is provided by an IEEE 802.15.4 radio providing about 100Kb/s of bandwidth.

[3] https://github.com/contiki-ng/contiki-ng

[4] https://github.com/Zolertia/Resources/wiki/Firefly

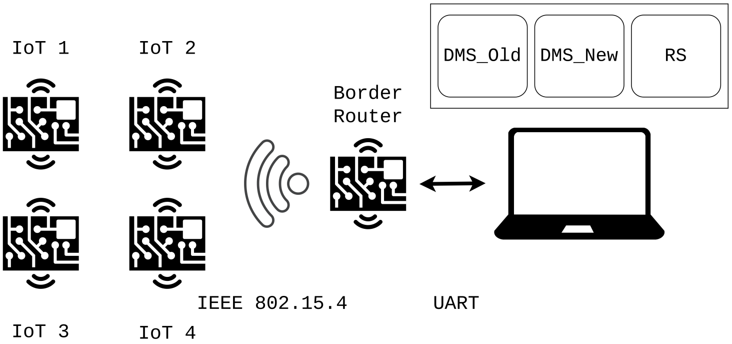


Figure 3: Experimental setup used in the evaluation.

6.2 Test Scenario

The test scenario consists of an initial setup phase where keys are distributed to the individual IoT devices and an ownership transfer phase. The initial setup phase is not in scope for the performance evaluation, only the ownership transfer process is included. We ran the ownership transfer scenario, of the four IoT devices, ten times.

6.3 Ownership transfer time

In order to evaluate the efficiency of our proposed scheme from a system perspective we timed the entire ownership transfer process. We measured the time elapsed from that the RS sends out the token T to when all IoT devices has been provisioned with new owner keys KO_i'. The time taken for the ownership transfer process is measured to a mean of 4.7s with a 95% confidence interval between 4.4s and 5s.

It should be noted that these times are for a single link-layer hop. Doing the ownership transfer process over another network, with higher latency, such as the internet would take longer time.

6.4 Energy consumption

Since the devices considered for this protocol usually are powered by a battery it is important that the energy consumed by the IoT device when executing the ownership transfer protocol is reasonable.

We have measured the energy usage on the constrained devices for both the radio modem and the CPU. The total energy consumption was measured to a mean of 0.18mJ. With a 95% confidence interval of the mean between 0.14mJ and 0.22mJ. For comparisons sake, the mean energy consumption of 0.18mJ is equal to the energy consumed by the CPU executing at full power for four seconds.

6.5 Memory Overhead

Constrained devices usually have a limited amount of memory available to store both code in ROM and variables and data in RAM. It is important for all protocols aimed at these types of devices are efficient in terms of memory usage. This is especially true for an ownership-transfer protocol, that is not used often.

To evaluate the memory utilization of our proposed protocol we have used the GNU-tools **size**[5] and **nm**[6] to evaluate and break down the the memory utilization of our implementation. The detailed breakdown of memory utilization can be seen in Table 2 below.

Table 2: Memory utilization.

Functions	Storage location	Utilization
HKDF-SHA256	ROM	256 Bytes
CBOR	ROM	165 Bytes
COSE	ROM	292 Bytes
Ownership Transfer	ROM	340 Bytes
Keys	RAM/ROM	100 Bytes

In summary; 100 Bytes of keys needs to be stored, together with around $\sim$ 500 Bytes of ROM for extra functions. Another $\sim$ 500 bytes is needed for the COSE functionality. Since DTLS is assumed to be existing on the device, AES-128 and SHA256, or equivalent are assumed to exist on the device. Either implemented in software or accelerated in hardware.

7 Related work

Protocols for ownership transfer have been studied in several fields. Both recently for IoT devices and earlier for RFID-tags. IoT infrastructures and RFID systems are not equal but have some common characteristics. RFID-tags and IoT systems are deployed in large numbers and efficient management of a large number of devices is necessary. IoT devices might have constrained resources and RFID-tags typically even less resources for computation and storage. IoT units are connected, usually wireless, and the ability to initiate communication with external entities. RFID-tags however are only capable of responding to requests. RFID-tags can only be read and written to locally, a reader must be in physical proximity to the RFID-tag to be able to communicate with the device. An IoT device can however receive communication originating practically anywhere, this creates a

[5] https://ftp.gnu.org/old-gnu/Manuals/binutils-2.12/html_node/binutils_8.html

[6] https://sourceware.org/binutils/docs/binutils/nm.html

bigger attack surface on IoT devices since an attack on the system can, in theory, originate from anywhere on the planet.

7.1 IoT Ownership Transfer

Internet of Things (IoT) are a very wide category of devices for a wide variety of purposes, with the common property that they are connected to a network in some way. When ownership transfer is studied in the realm of IoT devices authors often have different views of what types of devices constitute an IoT device. Devices considered can be connected medical equipment, wearables, smart consumer electronics such as fridges and CCTV-cameras. Other devices that are often grouped into IoT are sensor networks, building automation and connected sensors and actuators for industrial applications.

Tam and Newmarch state the problem of transferring ownership in [TN04] for Ubiquitous Computing Networks, a term that predates IoT. They define the term ownership and provide requirements for an ownership system. They also provide an example of an ownership transfer protocol. The protocol is based on public-key cryptography and defines how two parties transfer the ownership of a device.

Khan et. al. discuss ownership transfer for connected consumer products [Kha+19]. The focus of the ownership transfer process is less about re-keying the device and more about preserving privacy for information stored on the device. They also propose a novel idea of how to automatically start the ownership transfer process by detecting changes in the environment to determine if the device has been sold or given away.

Pradeep and Singh propose a protocol in [PS13] utilizing a trusted third party that they call a Central Key Server. The protocol requires physical proximity when the ownership transfer process is about to take place. The protocol does not specify exactly what type of IoT device that is considered, but only one device is transferred during each execution of the protocol.

In [LML14] the authors, Leng et al., propose an ownership transfer protocol for IoT devices with a TTP in the system. The authors intended use-case is traceability and monitoring of supply chains rather then re-keying an IoT deployment. The TTP needs, like in our system to establish secret keys, from the beginning, with IoT devices that will be transferred. Since the use-case of the protocol is traceability of IoT devices and the security analysis provided in the paper is brief, it is uncertain if this protocol can fulfill the use-case we present in this work. No implementation and experimental results are provided, but owing to the large amount of direct communication between the IoT device and TTP the network overhead of an ownership transfer will be big. In this work we have done a rigorous security analysis using formal verification methods and implemented our protocol on a physical constrained IoT device to verify its efficiency.

In [Mül+19] Müller et al. propose HomeCA. A life-cycle management system for consumer IoT. HomeCA uses certificates and by extension asymmetric cryptography, making it unsuitable for our use-case of constrained IoT devices. The authors present a comprehensive work, based on open standards, that is intended for more powerful devices compared to the very constrained devices that we consider in this work.

In [Agh+19] medical IoT unit authentication as well as ownership transfer is considered. The authors propose a new scheme called LACO, which is an improvement to an earlier medical three factor authentication protocol proposed by Zhang et al. in [Zha+17]. The authors have done a formal verification of their proposed protocol. Different from our work, the authors behind the LACO, do not considered the ownership transfer with respect to the IoT units themselves but only ownership transfer between users connected to a medical server, which in turn controls the medical IoT units.

7.2 IoT ownership transfer with Blockchain Technology and Smart Contracts

Many proposed protocols for Ownership Transfer utilize a TTP. In later years work has been done investigating if the TTP can be replaced with either entries on a Blockchain or Smart contracts. Some works aim to just keep track of the changing owners of an IoT device, this can be used to keep track of the current ownership status. For such an application a distributed ledger, on a Blockchain is a suitable solution instead of a TTP. For protocols where the TTP is used to facilitate the transfer of ownership, more functionality is needed, compared to a traditional Blockchain. Smart contracts is one such solution. The most common platform for smart contracts is Ethereum [Woo+14].

In [Bor+20] Borah et al. present a Blockchain based, used for Supply chain management. In this work the considered IoT devices are Mobile phones. In this work the ownership transfer is logged to a Blockhain as to later being auditable without needing a TTP in the system. The work of Borath et al. and other simmilar works such as [Sed+19] is one example of how Blockchains and Smart contracts can be used in the field of ownership management. The difference between these works and our, is that they present how to monitor who owns an asset after the ownership has changed whereas we are interested in *how* the ownership transfer is facilitated with regards to IoT security. However the present work can show the utility of Blockchains and Smart contracts as TTPs.

In [ATY19] Altun et al. present an IoT ownership and management scheme for home appliances with Digital Twins in a fog-computing environment. Digital Twins is a concept where a physical device has a digital replica, the Digital Twin. This Digital Twin can be used to synchronize data

The authors Islam et al. propose a smart contract-based ownership transfer scheme in [IK20]. It is intended for use in the sharing economy or rentals, such as

AirBnB etc. The specified use use case is an IP camera in a rented property. An IP camera is a more powerful devices compared to the one we consider. The system use a PUF to authenticate the IoT device to outside parties. The authors use a smart contract on a Blockchain to eliminate TTP. The re-keying of the IoT device is done with a TPM in the IP-camera. The scheme uses public-key cryptography on the IoT device to securely transport keys to the IoT device and is thus to resource intensive compared for our intended use-case.

In a paper by Alblooshi et al. [ASA18], the authors propose an ownership transfer protocol for Medical IoT devices, the scheme uses Smart contracts (Ethereum) to eliminate a TTP in the system. The intended use-case of this protocol is to track ownership with the purpose of establishing authenticity of medical IoT devices.

As can be seen from the previous work with replacing the TTP with a Blockchain together with smart contracts there seem to be promise in the field. However, some drawbacks with smart contracts and Blockchains exist. Blockchains uses asymmetric cryptography, that might be to resource intensive for very constrained devices. The idea of replacing the TTP with a smart contract looks attractive, but there are still issues. A smart contract is difficult or impossible to update or patch, either with bug-fixes or additional functionality. In addition, an incorrect smart contract can be a major security vulnerability, for example as in the case with the DAO vulnerability in 2016, where an implementation error caused large monetary losses.

7.3 Ownership Transfer Protocols for RFID-tags

The subject of secure ownership transfer has been studied in the field of RFID technology since 2005 [SIS05]. In the paper ”Tag Ownership in RFID systems: Survey of Existing Protocols and Open Challenges”[Taq+18] the authors list the research done in the field from 2005 to 2018. The authors also group protocols by features; Group transfer protocols and individual tag transfer protocols, trusted Third Party (TTP) protocols, and protocols where only the new and current owner take part. Lastly EPC-C1G2 [EPC08] compliant protocols and protocols that require more resources from the tags. The first papers for RFID-tag ownership transfer generally suffered from not satisfying some important security requirements. The early Satio paper [SIS05], does for instance not provide forward and backwards secrecy for the owners.

We are considering a model with IoT ownership transfer with the assistance of a trusted third party, the so-called ”Reset Server” (RS) (see Section 2 and Section 4). This entity has a very similar role as a TTP in RFID ownership transfer solutions. However, *different* from prior art work, we think that for IoT infrastructures, one would like to avoid the TTP to actual *choose* the credentials for the devices in the system but merely ”supervise” the transferring process. This has the main advantage that the RS, unlike the TTP in prior-art solutions, will not have complete knowledge of the final device credential after completing the ownership transfer process. TTP based protocols in prior-art are the ones that most closely resemble the model we consider and we will in the related work summary below, focus on TTP based protocols."

7.4 RFID Single ownership transfer

Much work has been done for owner transfer of single RFID-tags. Since we consider group transfer of IoT devices these protocols are mainly mentioned for completeness sake. Protocols that are intended for EPC-compliance are often forced to use non-standard solutions due to the extremely constrained nature of EPC-compliant RFID-tags. One such scheme can be found in [Cao+16]. The protocols that are not restricted by EPC-compliance often make use of standard cryptological functions such as symmetric ciphers and hash functions. One example of an ownership transfer protocol using a TTP can be found in [ZYP12].

7.5 RFID Group ownership transfer

Several group transfer protocols with a TTP have been proposed in the literature [KZP11] [Zuo10] [Sun+15] [HGY14] [BAS18]. The design goals of the different protocols are not uniform. They do not work with the very same security requirements. They also differ with respect to that one solution wants to achieve EPC-C1G2 compliance [Sun+15] and another want to have a group of RFID-tags to switch ownership simultaneously for instance [Zuo10].

A core characteristic we expect from an ownership transfer protocol, is backward and forward secrecy. This is not offered by the protocol suggested by Sundaresan et al. [Sun+15]. The group transfer protocol by Kapoor [KZP11] is an extension of an earlier variant for singe tag transfer [KP08]. Even if this is a simple and rather straightforward protocol, these protocols were later shown by Bagheri et al [BAS18] to be vulnerable to de-synchronization attacks (due to the simple fact that the message exchange between the TTP and the tag was not authenticated). The authors in [BAS18] also showed how to fix these shortcomings, but unlike our suggested protocol, their solution is dependent on a direct session between the tag (the IoT unit in our case) and the TTP. They also give the full power to the TTP that must have access to all key information (both the old and the new).

Inspired by an earlier work on grouping proofs for RFID tags [BMM08], Zuo proposed a new TTP based protocol for RFID ownership transfer [Zuo10]. Similar to the earlier grouping proof protocols, the design goal is to provide a proof of the ownership transfer of all tags in a group *simultaneously*, i.e., without the need of having connection to the back-end system representing the tag owner during the ownership switch. This means that the ownership transfer interactions only take place locally between the tag reader and the tags in the group connected to this reader. Later, the back-end system just can verify that the transfer has occurred. In and RFID system scenario this has some communication overhead reduction advantages but not in a system scenario with distributed IoT units. Hence, the off line requirement makes the ownership transfer unnecessarily complex for the IoT scenario we are considering. Furthermore, similar to other ownership protocols, the TTP is given full power by selecting all the new credentials using the solution in [Zuo10].

In [HGY14] another group ownership transfer protocol was proposed. This protocol shares our design goals with respect to forward and backward secrecy. Furthermore, it allows arbitrary location and grouping of tags based on group keys. This is a property most suitable also for IoT infrastructures. However, similar to other prior art, the solution in [HGY14] gives the TTP full knowledge of the key information. It also must has active sessions with all tags taking part in the ownership transfer process. Our protocol does not have these two limitations.

8 Conclusion

In this paper, we have presented the extended version of our previous work, where we presented an ownership transfer protocol for constrained IoT devices [GG20]. In our previous work, we identified the need for an ownership transfer protocol for constrained IoT devices. The constrained nature of the considered IoT devices necessitates that the protocol is resource-efficient, both in computational overhead and communication overhead. These requirements require a solution based on symmetric cryptography. We have investigated previous models and protocols for ownership transfer in the fields of IoT and RFID tags. Since IoT is such a different field, there are many different IoT devices with different capabilities. We have found that protocols for RFID-tags most closely related to the requirements we have identified.

We stated formal requirements from the related work, having investigated the state-of-the-art protocols for ownership transfer for both IoT devices and RFID-tags. We have formulated security requirements and functional requirements for a protocol for ownership transfer of constrained IoT devices. After stating the requirements, we have proposed and presented our protocol. After describing our proposed protocol, we have performed a rigorous security analysis of our proposed protocol. We have used two security analysis methods: formal protocol verification with Tamarin Prover and traditional reasoning, based on the security requirements. In the security analysis, we show that the previously stated security requirements hold. Next, the protocol was implemented as a proof-of-concept to be evaluated in terms of performance. The protocol proved to be as resource-efficient as hoped. The time required to transfer ownership is small for resource-constrained IoT units, and ownership transfer will not be frequent in the use-case we envision. We have also investigated the energy consumption of the protocol and found it to be reasonable. The memory footprint needed for the implemented protocol is small and is not prohibitively large for code that will be executed relatively infrequent.

The research field of ownership transfer protocols for IoT deployments has previously been explored for more powerful IoT devices, especially medical and consumer IoT devices. The area of constrained IoT devices is still relatively unexplored. However, there are many open possibilities for further work. For example, evaluating the performance of protocols in large infrastructures, i.e., hundreds to thousands of IoT units, is an interesting question to investigate. Implementing and deploying our protocol for real systems, such as industrial control systems or building automation, is another interesting question. Furthermore, investigating other trust models where no trusted third party is required is also an exciting research question. Last but not least is the question about the ownership models of the future. Data and computational devices are shared in a larger and larger extent, and it will be necessary to investigate how IoT devices will be handled in the future. Will devices always belong to one owner? Will they be transferred between owners, or will they be rented out to clients from one owner? These questions need to be considered when designing protocols that can accommodate more complex ownership structures for future infrastructure.

References

[Agh+19] S. F. Aghili et al. "LACO: Lightweight Three-Factor Authentication, Access Control and Ownership Transfer Scheme for E-Health Systems in IoT". In: *Future Generation Computer Systems* 96 (2019), pp. 410–424.

[ASA18] M. Alblooshi, K. Salah, and Y. Alhammadi. "Blockchain-based ownership management for medical IoT (MIoT) devices". In: *2018 International Conference on Innovations in Information Technology (IIT)*. IEEE. 2018, pp. 151–156.

[ATY19] C. Altun, B. Tavli, and H. Yanikomeroglu. "Liberalization of Digital Twins of IoT-Enabled Home Appliances via Blockchains and Absolute Ownership Rights". In: *IEEE Communications Magazine* 57.12 (2019), pp. 65–71.

[Bas+17] D. Basin et al. "Symbolically Analyzing Security Protocols using Tamarin". In: *ACM SIGLOG News* (Oct. 2017).

[BAS18] N. Bagheri, S. F. Aghili, and M. Safkhani. "On the security of two ownership transfer protocols and their improvements". In: *Int. Arab J. Inf. Technol.* 15.1 (2018), pp. 87–93.

[BEK14a] C. Bormann, M. Ersue, and A. Keranen. "Terminology for Constrained-Node Networks". In: *Internet Engineering Task Force (IETF): Fremont, CA, USA* (2014), pp. 2070–1721.

[BMM08] M. Burmester, B. de Medeiros, and R. Motta. "Provably Secure Grouping-Proofs for RFID Tags". In: *Smart Card Research and Advanced Applications*. Ed. by G. Grimaud and F.-X. Standaert. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 176–190.

[Bor+20] M. D. Borah et al. "Supply Chain Management in Agriculture Using Blockchain and IoT". In: *Advanced Applications of Blockchain Technology*. Ed. by S. Kim and G. C. Deka. Singapore: Springer Singapore, 2020, pp. 227–242.

[Cao+16] T. Cao et al. "RFID ownership transfer protocol based on cloud". In: *Computer Networks* 105 (2016), pp. 47–59.

[DMR16] M. Díaz, C. Martín, and B. Rubio. "State-of-the-art, challenges, and open issues in the integration of Internet of things and cloud computing". In: *Journal of Network and Computer Applications* 67 (2016), pp. 99–117.

[DY81] D. Dolev and A. C. Yao. "On the Security of Public Key Protocols". In: *Proceedings of the 22nd Annual Symposium on Foundations of Computer Science*. SFCS '81. Washington, DC, USA: IEEE Computer Society, 1981, pp. 350–357.

[DY83] D. Dolev and A. Yao. "On the Security of Public Key Protocols". In: *IEEE Transactions on information theory* 29.2 (1983), pp. 198–208.

[EPC08] EPCglobal Inc. *EPC Radio-Frequency Identity Protocols Generation-2 UHF RFID*. Report 1.2.0. EPCglobal Inc., 2008.

[Ero+10] P. Eronen et al. *Internet Key Exchange Protocol Version 2 (IKEv2)*. RFC 5996. 2010.

[Fie00b] R. Fielding. "Representational state transfer". In: *Architectural Styles and the Design of Netowork-based Software Architecture* (2000), pp. 76–85.

[GG20] M. Gunnarsson and C. Gehrmann. "Secure Ownership Transfer for the Internet of Things". In: *6th International Conference on Information Systems Security and Privacy, ICISSP 2020, 25 February 2020 through 27 February 2020*. SciTePress. 2020, pp. 33–44.

[HGY14] L. He, Y. Gan, and Y. Yin. "Secure group ownership transfer protocol with independence of old owner for RFID tags". In: *Computer modelling and new technologies* 18.12B (2014), pp. 209–214.

[IK20] M. N. Islam and S. Kundu. "IoT Security, Privacy and Trust in Home-Sharing Economy via Blockchain". In: *Blockchain Cybersecurity, Trust and Privacy*. Springer, 2020, pp. 33–50.

[Kha+19] M. S. N. Khan et al. "chownIoT: Enhancing IoT Privacy by Automated Handling of Ownership Change". In: *Privacy and Identity Management. Fairness, Accountability, and Transparency in the Age of Big Data: 13th IFIP WG 9.2, 9.6/11.7, 11.6/SIG 9.2.2 International Summer School, Vienna, Austria, August 20-24, 2018, Revised Selected Papers*. Cham: Springer International Publishing, 2019, pp. 205–221.

[KP08] G. Kapoor and S. Piramuthu. "Protocols for Objects with Multiple RFID Tags". In: *2008 16th International Conference on Advanced Computing and Communications*. Dec. 2008, pp. 208–213.

[KZP11] G. Kapoor, W. Zhou, and S. Piramuthu. "Multi-Tag and Multi-Owner RFID Ownership Transfer in Supply Chains". In: *Decision Support Systems* 52.1 (2011), pp. 258–270.

[Lan+16] J. Lanza et al. "Managing Large Amounts of Data Generated by a Smart City Internet of Things Deployment". In: *Int. J. Semant. Web Inf. Syst.* 12.4 (Oct. 2016), pp. 22–42.

[LML14] X. Leng, K. Mayes, and Y. Lien. "Ownership Management in the Context of the Internet of Things". In: *2014 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*. IEEE. 2014, pp. 150–153.

[Mei+13] S. Meier et al. "The TAMARIN Prover for the Symbolic Analysis of Security Protocols". In: *International Conference on Computer Aided Verification*. Springer. 2013, pp. 696–701.

[Mül+19] R. Müller et al. *HomeCA: Scalable Secure IoT Network Integration*. Gesellschaft für Informatik eV, 2019.

[Ode09] G. Oded. *Foundations of Cryptography: Volume 2, Basic Applications*. 1st. New York, NY, USA: Cambridge University Press, 2009.

[PS13] B. H. Pradeep and S. Singh. "Ownership Authentication Transfer Protocol for Ubiquitous Computing Devices". In: *2013 International Conference on Computer Communication and Informatics*. Jan. 2013, pp. 1–6.

[RM12] E. Rescorla and N. Modadugu. *Datagram Transport Layer Security Version 1.2*. RFC 6347. Jan. 2012.

[RNL11] R. Roman, P. Najera, and J. Lopez. "Securing the Internet of Things". In: *Computer* 44.9 (Sept. 2011), pp. 51–58.

[Sch17] J. Schaad. *CBOR Object Signing and Encryption (COSE)*. RFC 8152. RFC Editor, July 2017.

[Sed+19] K. Sedlak et al. *0xcert protocol*. 2019.

[SIS05] J. Saito, K. Imamoto, and K. Sakurai. “Reassignment Scheme of an RFID Tag’s Key for Owner Transfer”. In: *International Conference on Embedded and Ubiquitous Computing*. Springer. 2005, pp. 1303–1312.

[SO12] Y. B. Saied and A. Olivereau. “D-HIP: A distributed key exchange scheme for HIP-based Internet of Things”. In: *2012 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. June 2012, pp. 1–7.

[Sun+15] S. Sundaresan et al. “Secure Ownership Transfer for Multi-Tag Multi-Owner Passive RFID Environment with Individual-Owner-Privacy”. In: *Computer Communications* 55 (2015), pp. 112–124.

[Taq+18] E. Taqieddin et al. “Tag Ownership Transfer in Radio Frequency Identification Systems: A Survey of Existing Protocols and Open Challenges”. In: *IEEE Access* (2018).

[Tex15] I. Texas Instruments. “CC2538 powerful wireless microcontroller system-on-chip for 2.4-GHz IEEE 802.15. 4, 6LoWPAN, and Zigbee applications”. In: *CC2538 datasheet (April 2015)* (2015).

[TF16] H. Tschofenig and T. Fossati. “Transport Layer Security (TLS)/Datagram Transport Layer Security (DTLS) Profiles for the Internet of Things”. In: *RFC 7925*. Internet Engineering Task Force, 2016.

[TN04] P. Tam and J. Newmarch. “Protocol for Ownership of Physical Objects in Ubiquitous Computing Environments”. In: *IADIS international conference E-Society*. Vol. 2004. 2004, pp. 614–621.

[Vas14] J. Vasseur. *Terms used in routing for low-power and lossy networks*. Tech. rep. rfc 7102, January, 2014.

[Vög+16] M. Vögler et al. “A Scalable Framework for Provisioning Large-Scale IoT Deployments”. In: *ACM Trans. Internet Technol.* 16.2 (Mar. 2016), 11:1–11:20.

[Woo+14] G. Wood et al. “Ethereum: A secure decentralised generalised transaction ledger”. In: *Ethereum project yellow paper* 151.2014 (2014), pp. 1–32.

[Zha+17] L. Zhang et al. “Privacy Protection For E-Health Systems by Means of Dynamic Authentication and Three-Factor Key Agreement”. In: *IEEE Transactions on Industrial Electronics* 65.3 (2017), pp. 2795–2805.

[Zuo10] Y. Zuo. “Changing hands together: a secure group ownership transfer protocol for RFID tags”. In: *System Sciences (HICSS), 2010 43rd Hawaii International Conference on*. IEEE. 2010, pp. 1–10.

[ZYP12] W. Zhou, E. J. Yoon, and S. Piramuthu. “Simultaneous Multi-Level RFID Tag Ownership & Transfer in Health Care Environments”. In: *Decision Support Systems* 54.1 (2012), pp. 98–108.

DIPSAUCE: Efficient Private Stream Aggregation Without Trusted Parties

Abstract

Private Stream Aggregation (PSA) schemes are efficient protocols for distributed data analytics. In a PSA scheme, a set of data producers can encrypt data for a central party so that it learns the sum of all (encrypted) values, but nothing about each individual value. Due to this ability to efficiently enable central data analytics without leaking individual user data, PSA schemes are often used for IoT data analytics scenarios where privacy is important, such as smart metering. However, all known PSA schemes require a trusted party for key generation, which is undesirable from a privacy standpoint. Further, even though the main benefit of PSA schemes over alternative technologies such as Functional Encryption is that they are efficient enough to run on IoT devices, there exists no evaluation of the efficiency of existing PSA schemes on realistic IoT devices.

In this paper, we address both these issues. We first evaluate the efficiency of the state of the art PSA schemes on realistic IoT devices. We then propose, implement and evaluate a DIstributed setup PSA scheme for Use in Constrained Environments (DIPSAUCE). DIPSAUCE is the first PSA scheme that does not rely on a trusted party. Our security analysis and efficiency evaluation show that it is indeed possible to construct an efficient PSA scheme without a trusted central party. Suprisingly, our results also show that our method for distributing the setup procedure as a side effect makes the encryption procedure *more efficient* than the state of the art PSA schemes which rely on trusted parties.

Joakim Brorsson, Martin Gunnarsson “DIPSAUCE: Efficient Private Stream Aggregation Without Trusted Parties”. Submitted to *Privacy Enhancing Technologies Symposium (PETS) 2023*.

1 Introduction

Internet of Things (IoT) data analytics enable central parties to learn statistics derived from device data. In many such scenarios, this data is privacy sensitive. Thus systems must be designed with privacy in mind. Further, IoT devices are often *constrained*, *i.e.* they have one or more of the following characteristics: low computational power and memory, operate over low throughput lossy networks or are battery powered [BEK14b]. Thus, the computational and network cost of any scheme for constrained devices is of high importance.

Consider for example the concept of smart metering [Kab16] where a central party can calculate the sum of readings of household electricity meters in real-time. Disclosing individual readings in real-time reveals a surprisingly high amount of privacy sensitive data [Mol+10] about a household. Thus the central party is often considered untrusted and cannot be given individual data readings. There exist works studying how to centrally derive statistics without revealing individual data points for specific usecases (*e.g.* [KDK11; Lyu+18; GS18] in the case of smart meters). We are however interested in developing general techniques for IoT data analytics.

Functional Encryption A first technique that comes to mind is that of Functional Encryption (FE) [BSW11], which allows for evaluating a function on encrypted data if the evaluating party knows a *functional decryption key* for that function. For IoT data analytics on privacy sensitive data, the FE subclass of (Decentralized) Multi Client Functional Encryption ((D)MCFE) is particularly interesting, since it defines FE for multiple parties contributing encrypted data, and allows a central party to evaluate a function on the encrypted data. However, even the most efficient DMCFE schemes [Cho+18; Abd+19; Cho+20], which evaluate inner products of encrypted data, are too costly for constrained environments since they rely on bilinear parings or have ciphertext sizes proportional to the number of data producers.

Private Steam Aggregation In use cases which only require evaluating the sum of encrypted inputs (*e.g.* smart metering) rather than a general function or the inner product, we can look to PSA for more efficient constructions. The notion of PSA, was introduced in [Shi+11]. PSA is a similar concept to (D)MCFE, however it is restricted to computing sums rather than inner products (or general functions). This restriction allows more efficient constructions.

Both in the original PSA scheme [Shi+11] and in follow up works [CSS12; JL13; LEM14; BJL16; Emu17; BGZ18; EK21; Wal+21; Tak+23], the setup (which includes key generation) relies upon a *trusted party*. Such a design choice erodes trust in a privacy enhancing technology and is particularly engraving in the case of PSA schemes since the purpose of PSA is to avoid a central party with access to individual data.

We argue that since the purpose of a PSA scheme is to allow an untrusted party to derive statistics without learning anything about individual data points, relying on a trusted party is not in line with the goals of PSA. To the best of the authors knowledge, none of the known PSA schemes avoids a trusted party. Notably, in current state of the art PSA schemes [EK21; Wal+21], there are brief discussions on how the schemes could be modified to not rely upon trusted party by employing a distributed setup inspired by the DMCFE scheme in [Cho+20]. However in neither work is there any formal protocol description, security evaluation or efficiency evaluation of the proposed modification. In Section 3 we show that the proposed modifications are too inefficient for constrained environments. There is thus a need to develop a distributed setup PSA scheme which is *efficient* and *proven secure*.

1.1 Contributions

In this paper we (1) introduce a definition and a security model for PSA scheme without trusted setup, (2) present DIPSAUCE, the first PSA scheme which does not rely on a trusted party, (3) prove this scheme secure under static corruptions (and sketch modifications for security under mobile corruptions), (4) show its practical feasibility by implementing it on realistic, off-the-shelf devices. Since no other PSA scheme is evaluated on realistic devices, we also (5) implement two state-of-the-art PSA schemes [EK21; Wal+21] on realistic devices and compare the performance to our scheme. The devices we have selected are advertised as being suitable for smart-metering. DIPSAUCE is defined and proved in the standard model. Our implementation, however, uses a more practical building block with a hash function assumed to be a random oracle. Note that this is implementation specific rather than a limitation of the protocol.

Looking ahead, comparing the setup and keygen procedures in DIPSAUCE with the suggested distributed setups of KH-PRF-PSA and LaSS-PSA with 10000 parties, our results show a speedup of 78x and 49x respectively. For the encryption procedure our protocol shows a speedup of 22x compared to KH-PRF-PSA and 50x to LaSS-PSA.

1.2 Our Techniques

Our PSA scheme DIPSAUCE is a variant of the scheme proposed in [Wal+21], and takes inspiration from the sketch of a distributed setup proposed in [EK21; Wal+21]. The security of the sketched setup procedures in these schemes crucially relies on each party deriving a shared key for *each other party*. This approach is secure against an adaptive adversary allowed to corrupt up to $n-2$ parties. However, as we show later, the $\mathcal{O}(n)$ complexity of this technique makes such a protocol infeasible for the number of parties (1000-10000) suggested in [EK21; Wal+21], on constrained devices.

To avoid this situation, we instead consider a static adversary which can corrupt up to t out of the n parties (where *e.g.* $t = \frac{n}{2}$). This allows us to design a more efficient protocol. Specifically, we leverage a *randomness beacon* to determine a smaller and *random committee* [CM19], which is unpredictable for the adversary, of $k, k \ll n$ other parties to derive a shared key with. When selecting parties at random from the set of all parties, the ratio of corrupted parties is (probabilistically) preserved [Dav+21]. Thus the complexity can be reduced from $\mathcal{O}(n)$ to $\mathcal{O}(k)$, while maintaining security. When $k \ll t < n$ the setup stage is efficient enough to be feasible.

Finally, building upon such an efficient distributed setup, it is then possible to modify the protocol to be secure against a mobile adversary, by periodically rerunning the setup stage.

2 Preliminaries

In this section we introduce notation and recall known constructions relevant to our scheme.

Notation Throughout the paper $\lambda \in \mathbb{N}$ denotes the computational security parameter. A specific party in a scheme is denoted as $\mathcal{P}_i$. We will use the notation $\vec{a}[i]$ to denote the i'th element of the vector $\vec{a}$. We use $[n]$ as a short hand notation for $\{1, \ldots, n\}$. We denote the set of permutations of $[n]$ by $\mathsf{Perm}(n)$ and the k:th permutation of this set as $\mathsf{Perm}_k(n)$. For a permutation of $[n]$, $\rho_k = \mathsf{Perm}_k(n)$, we denote the mapping of the i:th element in $[n]$ as $\rho_k(i)$. We denote a graph as $G = (V, E)$, where V is the set of vertices in the graph and E the set of edges. The set of neighbouring vertices of $v_i \in V$ is denoted $N(v_i)$. We also let $\vec{J_i}$ denote the set of all indices of vertices in $N(v_i)$. We denote the floor function of x, *i.e.* the greatest integer less than or equal to x, as $\lfloor x \rfloor$. As a shorthand we sometimes write $(-1)^{(i<j)}$. In this notation $(i < j)$ is the boolean function so that $(-1)^{(i<j)} = (-1)$ when $i < j$ and $(-1)^{(i<j)} = 1$ when $i > j$. The function is undefined for $i = j$.

k-Regular Graphs A k-regular graph is a graph in which each vertex has exactly k neighbours. Efficient algorithms for generating regular graphs are well known [Mer99].

Private Stream Aggregation A Private Stream Aggregation (PSA) scheme is a scheme which for each time-step l allows an evaluator to learn the sum of inputs $\{m_1, \ldots, m_n\}$, from a set of parties $\{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$ without learning the individual inputs.

In existing PSA schemes, a trusted third party executes the setup procedure and distributes the secret keys to the aggregator and clients. We here recall the definition of such centralized setup PSA schemes and their corresponding security notion of *aggregator obliviousness*, as defined in [EK21].

Definition 2.1 (Private Stream Aggregation)**.** A PSA scheme is defined by the procedures:

- $\mathsf{Setup}(\lambda, n)$: On input the security parameter λ and the number of parties n, output public parameters pp and $n+1$ secret symmetric encryption keys $\{\mathsf{ek}_i\}_{i\in[n+1]}$ (where ek_{n+1} is the aggregator key, sometimes alternatively denoted as ek_a).

- $\mathsf{Enc}(\mathsf{pp}, \mathsf{ek}_i, m_i, l)$: On input the public parameters pp, an encryption key $\mathsf{ek}_i, i \leq n$, a message $m_i \in \mathbb{Z}_R, R \in \mathbb{N}$ and a label $l \in \mathcal{L}$, output the ciphertext c_i.

- $\mathsf{Aggr}(\mathsf{pp}, \mathsf{ek}_a, \{c_i\}_{c\in[n]}, l)$: Given the public parameters pp, the aggregator key ek_a, a set of n ciphertexts $\{c_i\}_{c\in[n]}$, and a label l, it outputs the sum of all plaintexts, $M \pmod R$.

A PSA scheme $\mathsf{PSA} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{AggrDec})$ must satisfy *correctness*. For any $n, \lambda \in \mathbb{N}, m_1, \ldots, m_n \in \mathbb{Z}_R$ and any label $l \in \mathcal{L}$, so that
$(\mathsf{pp}, \{\mathsf{ek}_i\}_{i\in[n+1]}) \leftarrow \mathsf{Setup}(\lambda, n)$, and $\forall\{c_i\}_{i\in[n]} : c_i = \mathsf{Enc}(\mathsf{pp}, \mathsf{ek}_i, l, m_i)$, correctness is satisfied if:

$$\mathsf{AggrDec}(\mathsf{pp}, \mathsf{ek}_a, l, \{c_i\}_{i\in[n]}) = \sum_{i\in[n]} m_i \pmod R$$

Further, a *secure* PSA scheme must satisfy Aggregator Obliviousness (AO). The below definition of AO regards *encrypt-once* security, where a client only encrypt one value per label.

Definition 2.2 (Aggregator Obliviousness)**.** Let PSA be a *PSA* scheme. Let the experiment AO_b in Figure 1 be defined with the following oracles:

- $\mathsf{QCorrupt}(i)$: The oracle outputs the encryption key ek_i of user i. For $i = n+1$, it outputs the aggregator key ek_a.

- $\mathsf{QEnc}(i, m_i, l^*)$: The oracle outputs $ct_i = \mathsf{Enc}(\mathsf{ek}_i, m_i, l^*)$ on a query.

- $\mathsf{QChallenge}(\mathcal{U}, \{m_i^0\}_{i\in\mathcal{U}}, \{m_i^1\}_{i\in\mathcal{U}}, l^*)$: The adversary specifies a set of user indices $\mathcal{U} \subseteq [n]$, a label l^* and two challenge messages for each user from $\mathcal{U}$. The oracle answers with encryptions of m_i^b, that is
$\{c_i \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{ek}_i, m_i^b, l^*)\}_{i\in\mathcal{U}}$. This oracle can only be queried once during the game. If the adversary does not query this oracle, $\mathcal{U} = \emptyset$.

At the end of the game, $\mathcal{A}$ outputs a guess α, whether $b = 0$ or $b = 1$. A PSA scheme is *Aggregator Oblivious*, if for every PPT adversary $\mathcal{A}$ there exists a negligible function negl such that for all sufficiently large λ, $\mathsf{Adv}^{AO}_{\mathcal{A},\mathsf{PSA}}(\lambda, n) = \mathsf{negl}(\lambda)$.

```
$\mathsf{AO}_b(\lambda, n, \mathcal{A})$

$(\mathsf{pp}, \{\mathsf{ek}_i\}_{i \in [n+1]}) \leftarrow \mathsf{Setup}(\lambda, n)$
$\alpha \leftarrow A^{\mathsf{QCorrupt}(\cdot),\mathsf{QEnc}(\cdot),\mathsf{QChallenge}(\cdot,\cdot,\cdot,\cdot)}(\mathsf{pp})$
if condition $(*)$ is satisfied then
    Output $\alpha \overset{?}{=} b$
else
    Output 0
end if
```

Figure 1: The aggregator obliviousness game defining security for a PSA scheme.

To formally define the condition $(*)$, the following sets are introduced:

- Let $\mathcal{E}_l^* \subseteq [n]$ be the set of all users for which $\mathcal{A}$ has asked an encryption query on label l.
- Let $\mathcal{CS} \subseteq [n]$ be the set of *users* for which $\mathcal{A}$ has asked a corruption query. Even if the aggregator is corrupted, this set only contains the corrupted *users* and not the aggregator.
- Let $\mathcal{Q}_{l^*} := \mathcal{U} \cup \mathcal{E}_{l^*}$ be the set of users for which $\mathcal{A}$ asked a challenge or encryption query on label l^*.

Condition $(*)$ is *satisfied* (as used in Figure 1), if all of the following conditions are satisfied:

- $\mathcal{U} \cap \mathcal{CS} = \emptyset$. This means that all users for which $\mathcal{A}$ received a challenge ciphertext must stay uncorrupted during the entire game.
- $\mathcal{A}$ has not queried $\mathsf{QEnc}(i, m_i, l^*)$ twice for the same (i, l^*). Doing so would violate the encrypt-once restriction.
- $\mathcal{U} \cap \mathcal{E}_{l^*} = \emptyset$. This means that $\mathcal{A}$ is allowed to get a challenge ciphertext only from users for which they ask an encryption query on the challenge label l^*. Doing so would violate the encrypt-once restriction.
- If $\mathcal{A}$ has corrupted the aggregator *and* $\mathcal{Q}_{l^*} \cup \mathcal{CS} = [n]$ the following equality must hold in order to prevent trivial wins by using the knowledge of the aggregators knowledge of the sum of all honest parties plaintexts.

$$\sum_{i \in U} x_i^0 = \sum_{i \in U} x_i^1$$

This condition is called the *balance-condition*.

In [Wal+21] and [EK21] the authors consider adaptive corruptions. $\mathcal{A}$'s advantage is defined as

$$\mathsf{Adv}^{AO}_{\mathcal{A},\mathsf{PSA}}(\lambda, n) =| Pr[\mathsf{AO}_O(\lambda, n, \mathcal{A}) = 1] - Pr[\mathsf{AO}_1(\lambda, n, \mathcal{A}) = 1] |$$

Randomness Beacons A beacon [BCG15] is a function $r = \mathsf{Beacon}(t)$ which returns an m-bit near-uniformly random value r at each time interval t. Informally, a secure beacon should be *unpredictable*, *i.e.* the advantage for an adversary predicting r before time t should be negligible, *unbiased*, *i.e.* r is statistically close to an m-bit uniformly random string, *universally samplable*, *i.e.* any party should be able to obtain r after time t, and *universally verifiable i.e.* any party can verify that no party had access to the random sample used to construct r before time t.

Non-interactive Key Exchange We here recall Non-Interactive Key Exchange (NIKE) as defined in [Cho+20].

Definition 2.3 (Non-Interactive Key Exchange). A Non-Interactive Key Exchange scheme consists of the following algorithms:

$\mathsf{Setup}(\lambda)$: On input a security parameter λ, output public parameters pp which are implicit input to other NIKE procedures.

$\mathsf{KeyGen}()$: Outputs a party's public key pk_i and corresponding secret key sk_i.

$\mathsf{SharedKey}(\mathsf{pk}_i, \mathsf{sk}_j)$: On input a public key pk_i and secret key sk_j, deterministically output a shared key K.

We say that a NIKE scheme is correct if $\Pr[\mathsf{SharedKey}(\mathsf{pk}_i, \mathsf{sk}_j) = \mathsf{SharedKey}(\mathsf{pk}_j, \mathsf{sk}_i)] = 1$. We say that a NIKE scheme is secure against a computationally bounded adversary given $(\mathsf{pk}_i, \mathsf{pk}_j)$ if the adversary cannot distinguish the output of $\mathsf{SharedKey}(\mathsf{pk}_i, \mathsf{sk}_j)$ from from a random string of the same length, when both $\mathcal{P}_i$ and $\mathcal{P}_j$ are honest. We refer to [Cho+20, Def. 15] for a full definition of the security game.

Pseudo Random Functions Let $\mathcal{F}$ denote a family of efficiently-computable functions $F_k : K \times X \to Y$ indexed by $k \in K$. The family $\mathcal{F}$ is said to be a (t, ϵ) strong PRF if for every $k \in K$, and all adversaries $\mathcal{A}$ running in time t can not distinguish $F(k, .)$ from a random function $f : X \to Y$. To be formal, we write:

$$|Pr[\mathcal{A}^{F_k(.)} = 1] - Pr[\mathcal{A}^{f(.)} = 1]| < \epsilon$$

To aid the reader, we will denote such a function F_k PRF_k from here on.

$(\mathcal{F})_{\mathsf{KH\text{-}PRF}} \subset \mathcal{F}$ is said to be *additively* key-homomorphic if for $F_{k_i}, F_{k_j} \in (\mathcal{F})_{\mathsf{KH\text{-}PRF}}$ the following condition holds

$$F_{k_i}(x) + F_{k_j}(x) = F_{k_i+k_j}(x)$$

We will denote such a function $\mathsf{KH\text{-}PRF}_k$.

3 Evaluating the performance of state of the art PSA schemes

In this section, we evaluate the performance of the client side operations (*i.e.* the encryption procedure, since the schemes are defined with a centralized setup) of the two PSA schemes [EK21] and [Wal+21], which can be considered the current state of the art due to them being the most efficient schemes. The security model in [EK21] and [Wal+21] is stronger, than the one in [Tak+23] since [Tak+23] requires one trusted, powerful, device for each constrained device for their claim of better performance. We therefore consider [EK21] and [Wal+21] to be the state of the art.

The first scheme [EK21], which we will call KH-PRF-PSA is based on a Key Homomorphic PRF. The second scheme [Wal+21], which we refer to as LaSS-PSA is based upon a primitive called Labeled Secret sharing (LaSS), which is also introduced in [Wal+21].

Both these works evaluate the performance of their schemes on Intel i5 CPUs. Therefore, these performance evaluations gives little insight into how the schemes perform on realistic hardware. We therefore investigate whether these schemes can be run on constrained devices.

3.1 Scenario and Experiment Setup

Scenario In our IoT data analytics scenario, a set of n *Clients*, each measures some statistic, *e.g.* power in a smart metering scenario, and wishes to communicates the sum of the measurements to a *Server* (without revealing individual measurements).

Setup Since IoT data analytics schemes must be able to operate on constrained hardware, we evaluate the client side operations of PSA schemes on devices from the CC1352 series of devices with ARM Cortex M4 processors. These devices can be considered “mid-range” constrained devices. They are classified as C3 devices in the IETF draft for terminology on constrained devices [Bor+22]. The CC1352 platform has hardware acceleration for Elliptic Curve Diffie-Hellman, AES and SHA256, as well as a True Random Number Generator and a capabilities to run the wireless protocol IEEE802.15.4 Low Power Personal Area Network (LoWPAN). The computer running the *Server* has an *Border-Router* that connects to the *Client*. We provide additional details on the experiment setup and the CC1352 platform in Appendix 7.

Experiments We evaluate the client side efficiency of LaSS-PSA and KH-PRF-PSA by measuring the execution time of the respective Enc procedures. Time is measured from the start of the process until the ciphertext is ready to be transmitted. No network operations are included in this test. In the experiments in [Wal+21; EK21], the number of parties tested are from 1000 to 10000 in even increments of 1000. Our tests are done for client group sizes (*i.e.* n) of 1024, 2025, 3025, 4096, 5041, 6084, 7056, 8100, 9025 and 10000. These sizes are selected to be comparable with the experiments in previous works, while remaining compatible with requirements in our specific implementation of the DIPSAUCE protocol, which has additional requirements on the group sizes as explained in Section 6.1. The encrypt procedure was repeated 10 times for group size.

3.2 Protocol Definitions and Implementations

KH-PRF-PSA

We recall the definition of the KH-PRF-PSA protocol from [EK21] in Protocol 1.

Protocol 1 – The KH-PRF-PSA scheme in [EK21].

```
Setup($\lambda, n$):
    $\forall i \in [n] : \mathsf{ek}_i \xleftarrow{\$} \mathbb{Z}_R^\lambda$
    $\mathsf{ek}_a = \sum_{i \in [n]} \mathsf{ek}_i$
    return $\mathsf{ek}_a, \{\mathsf{ek}_i\}_{i \in [n]}$

Enc($\mathsf{ek}_i, m_i, l$):
    $t_i = \mathsf{KH\text{-}PRF}_{\mathsf{ek}_i}(l)$
    $c_i = (t_i + m_i) \pmod{R}$
    return $c_i$

Aggr($\mathsf{ek}_a, \{c_i\}_{i \in [n]}, l$):
    $m_a = \sum_{i \in n} c_i - \mathsf{KH\text{-}PRF}_{\mathsf{ek}_a}(l) \pmod{R}$
    return $m_a$
```

We here implement the KH-PRF-PSA realization in [EK21, Sec. 4], which uses a KH-PRF secure in the Random Oracle Model (ROM) to mask the message. The realization uses parameter $R \in \mathbb{N}$ and security parameters $\lambda = 2096$, $q = 2^{128}$ and $p = 2^{85}$. The encryption key ek is a vector of λ elements from $\mathbb{Z}_q$ and thus has size $\lambda \cdot q = 33536$ bytes.

The KH-PRF is defined as the inner product of the ek and the output of the function $H'(l)$ (where l is the given label):

$$\mathsf{KH\text{-}PRF}_{\mathsf{ek}}(l) = \lfloor \langle H'(l), \mathsf{ek} \rangle \rfloor_p \tag{1}$$

Where:

$$\lfloor x \rfloor_p = \lfloor x \cdot p/q \rfloor$$

The function $H'(l)$ is in turn defined as a vector of λ hashes of the label concatenated with a counter, and reduced modulo q:

$$H'(l) = \begin{pmatrix} H(l||""||"1") \pmod q \\ \vdots \\ H(l||""||"\lambda") \pmod q \end{pmatrix} \tag{2}$$

In the instantiation in [EK21] SHA3-512 is used for the hash function H. We however select a more efficient hash function, SHA256, which is hardware accelerated on the CC1352 platform.

LaSS-PSA

We recall the LaSS-PSA scheme from [Wal+21], presented in Protocol 2. The LaSS-PSA realization is presented using the notation $(-1)^{(i<j)}$ from Section 2. The realization uses parameter $R \in \mathbb{N}$ and the security parameter $\lambda = 128$. Note that $K_{i,i}$ is left undefined. LaSS-PSA uses LaSS to mask the message. We here implement the version which instantiates the PRF using AES-128, since its the most efficient instantiation of LaSS in the measurements of [Wal+21], and is hardware accelerated on the CC1352 platform. The encryption key ek consists of a vector of n elements from $\mathbf{Z}_R$, where n is the number of users in the system, and thus has size $n \cdot log_2(R)$.

Protocol 2 – The LaSS-PSA scheme [Wal+21].

```
Setup($\lambda, n$):
  $\forall i \in [n+1], \forall j$ s.t. $n \leq j > i : K_{i,j} \xleftarrow{\$} \mathbb{Z}_R$
  $\forall i \in [n+1], \forall j$ s.t. $n \leq j < i : K_{i,j} = K_{j,i}$
  let $\mathsf{ek}_i = \vec{K}_i$ be the vector s.t. $\forall j \in [n] : \vec{K}_i[j] = K_{i,j}$
  return $\{\mathsf{ek}_i\}_{i \in [n+1]}$

Enc($\mathsf{ek}_i = \vec{K}_i, m_i, l$):
  $t_i \leftarrow \sum_{j \in [n+1] \setminus \{i\}} (-1)^{i<j} \cdot \mathsf{PRF}_{\vec{K}_i[j]}(l)$
  $c_i = (t_i + m_i) \pmod R$
  return $c_i$

Aggr($\mathsf{ek}_a = \vec{K}_{n+1}, \{c_i\}_{i \in [n]}$):
  $m_a = \sum_{i \in n} c_i + \sum_{j \in [n]} \mathsf{PRF}_{\vec{K}_{n+1}[j]}(l) \pmod R$
  return $m_a$
```

3.3 Results

The results of the experiments measuring the performance of the encryption in KH-PRF-PSA and LaSS-PSA are available in Figure 2.

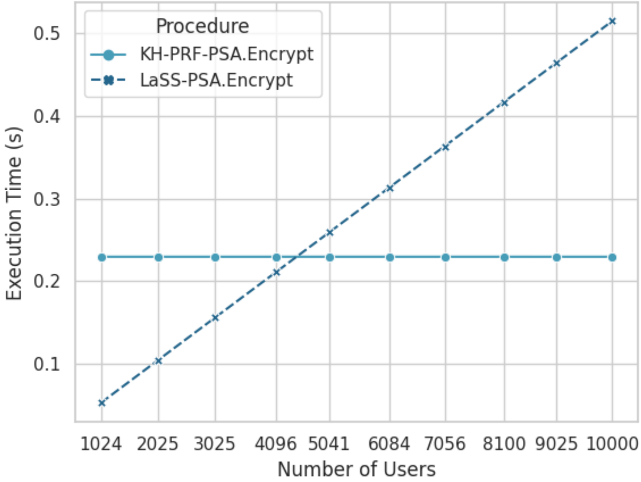


Figure 2: Execution time in seconds of the Enc procedure in KH-PRF-PSA and LaSS-PSA.

The KH-PRF construction used in KH-PRF-PSA has a execution time independent of the number of parties. Our measurements shows this constant execution time to be 230 ms, regardless of the numbers of parties. The performance of LaSS-PSA is linear with a coefficient of 0.05 ms per party in the system. The lines intersect at 4200 users. These trends correspond to the results from [EK21, Section 4.4], but the execution times for KH-PRF-PSA are around 200x longer in our measurements compared to the numbers presented in [EK21]. For LaSS-PSA the execution time per user is also around 200x longer in our measurements compared to the numbers presented in [EK21]. This discrepancy stems from the fact that our experiments are executed on a constrained devices whereas the experiments in [EK21] are executed on an Intel Core i5 CPU.

4 Evaluating the methods for a distributed setup proposed in [EK21; Wal+21]

Recall that all previous PSA schemes, including KH-PRF-PSA and LaSS-PSA, are presented with a trusted party for key distribution. Both KH-PRF-PSA [EK21] and LaSS-PSA [Wal+21], briefly discuss an approach to distribute the setup by negotiating a key between each party in the scheme, but does not give details on how to do this. In this section, we give details on how to construct the proposed solutions for distributed setup, implement the resulting schemes on the CC1352 platform with an ARM Cortex M4 processor, and evaluate the performance of the client side operations (*i.e.* both setup and encryption since the setup is now performed by the clients). This gives us an estimate for the performance of this approach to distributing the setup of PSA scheme.

It is not our intent to prove the security of distributed setup for PSA. We here only wish to show its (in)efficiency. Definitions and proofs for distributed setup are instead available for the scheme in Section 5.

4.1 A Distributed Setup for KH-PRF-PSA

Ernst and Koch propose a decentralized setup protocol in [EK21, Section 5.1] based on the sum-of-PRF technique. Let us here briefly recall this technique.

Sum-of-PRFs The sum-of-PRFs technique, first introduced as part of a scheme in [CC09] and later used in *e.g.* [Cho+20, Sec. 6.2], allows the parties $\{\mathcal{P}_1, \dots, \mathcal{P}_n\}$ to derive a sum of their respective inputs $\{m_1, \dots, m_n\}$ without revealing the individual m_i:s from honest users. That is, an adversary who corrupts $m < n - 2$ parties can learn the sum of the inputs of the honest users by subtracting the input from the corrupt users from the sum of all inputs. The technique assumes that each pair of users, $\mathcal{P}_i, \mathcal{P}_j$ has a shared secret $K_{i,j}$. To mask its message m_i, $\mathcal{P}_i$ derives $c_i \leftarrow m_i + \sum_{j \in [n] \setminus \{i\}} (-1)^{i<j} \cdot \mathsf{PRF}_{K_{i,j}}(x)$ (note the $(-1)^{i<j}$ notation from Section 2). Then, the sum of *all* m_i can be calculated as $\sum_{i=1}^{n} c_i = \sum_{i=1}^{n} m_i$. Summing any set smaller than n of c_i containing at least 2 ciphertexts from honest users will result in a random output.

Decentralizing the Protocol We here introduce a detailed realization of the suggested approach to decentralize the protocol. Note that in order for this distributed setup to be compatible with Protocol 1, the evaluator must derive k_a as the sum of all aks_i, *i.e.* $k_a = \sum_{i \in [n]} \mathsf{aks}_i$.

Let us now describe our implementation of Protocol 3. During Setup each device generate λ random integers from $\mathbb{Z}_R$, they form the encryption-key ek_i. Aggregating all ek_i without revealing individual ek_i is done using the sum-of-PRF technique.

Each device generate a key-pair (pk_i, sk_i) using Non Interactive Key Exchange (NIKE), and post their identity, $\mathcal{P}_i$, and public key, pk_i, to a Public Key Infrastructure (PKI). The device then generate a shared secret with NIKE for all other parties in the system. This shared secret is used with sum-of-PRF to mask ek_i. When masking, ek_i is interpreted as a vector of length λ integers mod R. The device sends the masked sk_i to the aggregator.

In our implementation we have used ECDH P-256 as the building block to NIKE. ECDH P-256 is hardware accelerated on the CC1352 platform. The PRF

Protocol 3 – Distributed Setup for KH-PRF-PSA

```
$\mathsf{Setup}(\lambda, n, k, i)$:
let $\vec{E}_i$ be a vector where $\forall j \in \{1, \dots, \lambda\} : \vec{E}_i[j] \xleftarrow{\$} \mathbb{Z}_R$
$\mathsf{ek}_i \leftarrow \vec{E}_i[1] || \dots || \vec{E}_i[\lambda]$
$(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{NIKE.KeyGen}()$
Post $(\mathcal{P}_i, \mathsf{pk}_i)$ to the PKI
Wait until the PKI returns a $\mathsf{pk}_j$ for each $j \in [n]$
for $j \in [n] \setminus \{i\}$ do
    $K_j \leftarrow \mathsf{NIKE.SharedKey}(\mathsf{pk}_j, \mathsf{sk}_i)$
end for
for $\ell \in \{1, \dots, \lambda\}$ do
    $b_{i,\ell} \leftarrow \sum_{j \in [n] \setminus \{i\}} (-1)^{i<j} \cdot \mathsf{PRF}_{K_j}(\ell)$
    $\vec{A}_i[\ell] = \vec{E}_i[\ell] + b_{i,\ell} \pmod R$
end for
$aks_i \leftarrow \vec{A}_i[1] || \dots || \vec{A}_i[\lambda]$
return $\mathsf{ek}_i, \mathsf{aks}_i$
```

was instantiated using hardware accelerated AES-128. Keys and public keys were transmitted using CoAP [SHB14].

4.2 A Distributed Setup for LaSS-PSA

Waldner et al. propose a decentralized setup protocol in [Wal+21, Sec. 7], which we now describe. Note that to make this setup compatible with Protocol 2, the aggregator must also execute the setup protocol.

Protocol 4 – Distributed Setup for LaSS-PSA.

```
$\mathsf{Setup}(\lambda, k, n, i)$:
$(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{NIKE.KeyGen}()$
Post $(\mathcal{P}_i, \mathsf{pk}_i)$ to the PKI
Wait until the PKI returns a $\mathsf{pk}_j$ for each $\mathcal{P}_j \in \mathcal{P}$
for $j \in [n] \setminus \{i\}$ do
    $K_j \leftarrow \mathsf{NIKE.SharedKey}(\mathsf{pk}_j, \mathsf{sk}_i)$
end for
return $\mathsf{ek}_i = \vec{K}_i$
```

Let us now describe our implementation of Protocol 4. Each device generate a key-pair (pk_i, sk_i) using NIKE, and post their identity $\mathcal{P}_i$ and public key pk_i

to a PKI. The device then generates a shared secret, K_j, with NIKE for all parties keys in the system. All K_j are stored in a vector and form the encryption key ek_i.

In our implementation we have used ECDH P-256 as the building block to NIKE. ECDH P-256 is hardware accelerated on the CC1352 platform. The PRF was instantiated using hardware accelerated AES-128. Keys and public keys were transmitted using CoAP [SHB14].

4.3 Experiments and Results

The setup and experiments described in this section is the same as described in Section 3.1 and Appendix 7 with the following modifications. Instead of measuring Enc(), we measure the Setup() execution times. We measure the execution times from the start of the encryption process, until the ciphertext is ready to be transmitted over the network.

We show the results of the experiments described above are depicted in Figure 3.

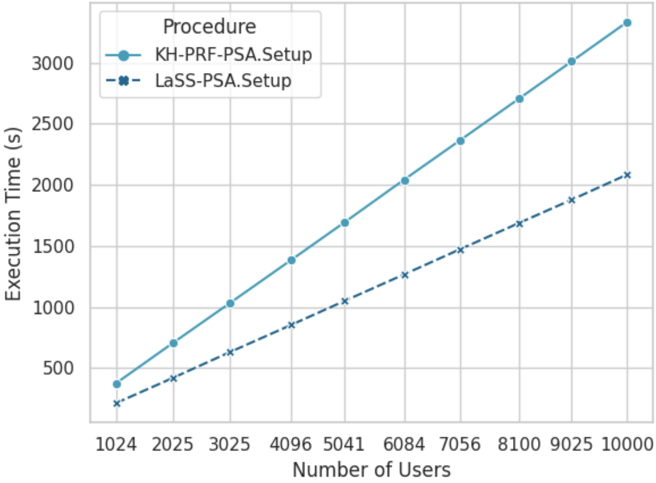


Figure 3: Execution time in seconds of the Setup procedure in KH-PRF-PSA and LaSS-PSA.

The figure shows that the execution times of the setup parts of both KH-PRF-PSA and LaSS-PSA grow linearly with the number of users in the system. The coefficient for KH-PRF-PSA is higher than that for LaSS-PSA. The reason for this difference in performance is that KH-PRF-PSA, in addition to deriving pairwise shared keys which is done in both Protocol 3 and Protocol 4, also generates a larger secret key ek_i (step 1-2 in Protocol 3) and mask ek_i before it is sent to the aggregator (step 9-13 in Protocol 3). Thus LaSS-PSA outperforms KH-PRF-PSA for all number of users in the system.

5 Efficient Distributed Setup PSA

The results in the previous section show that the existsing suggestions for obtaining a distributed setup for PSA schemes in [Wal+21; EK21] are infeasible in practice. To address this, we now present our protocol DIPSAUCE. It can be seen as a distributed setup variation of the protocol in [Wal+21, Section 4], modified so that the Setup procedure no longer generates keys centrally. Instead, we introduce a KeyGen procedure which each party executes independently.

Although we here present a specific protocol, our approach can be used to distribute the setup of other PSA schemes, for example the scheme in [EK21].

Approach The distributed setup procedures in Section 4 use the sum-of-PRFs technique, which works by each party evaluating a PRF once for each party in its *committee*. This committee consists of *all other parties*, and thus its size is $n - 1$. In these schemes, a targeted device is secure against an *adaptive* adversary which corrupts up to $n - 2$ of the committee parties (but not the targeted device itself). While this is a very strong security guarantee, we have shown in previous sections that the resulting protocol is rendered too inefficient for practical use. The main bottleneck giving rise to this inefficiency is the size of the committee.

How then to enable more efficient constructions by reducing the size of the committee, without sacrificing security? For example a committee of size $\sqrt{n}$ would be much more efficient, but if it is only secure against up to $\sqrt{n} - 1$ corruptions it cannot be said to be as secure. A key insight is that a *static* or *mobile* adversary cannot target devices in a committee for corruption (within an epoch) if it cannot predict what devices constitutes the committee. Using an unpredictable committee of size $k < n$ we can create a more efficient construction, secure in the presence of a *static* or a *mobile* adversary capable of corrupting up to t devices, where $k < t < n$.

The technical novelty of the protocol lays in how it uses a k-regular graph and a randomness beacon to efficiently establish unpredictable committees. The protocol defines each committee using the output of a public randomness beacon. However, an efficient protocol cannot directly use the output of the beacon to determine the committees. Sampling n committees of size k and transferring this data to the devices would mean transferring nk group elements to each device, which is not feasible in scenarios with constrained devices or networks. Instead, we first let each device be represented as a vertex in a k-regular graph which is part of the system configuration. Then, a *single* output of the beacon is used to determine a pseudorandom permutation of this graph. The committee of each party is then determined by the k neighbours in the randomly permuted graph. This committee is then used in a *threshold* sum-of-PRFs where each party evaluates a PRF for k other parties.

Aggregation output In line with [EK21; Wal+21], we consider a definition for PSA which outputs the sum of all plaintexts to the aggregator, *i.e.* we do not strive to achieve differential privacy. In contrast to existing definitions of PSA, no secret key is needed to aggregate the sum of plaintexts. This is a more general definition. If it is a desired system property to specifically allow only one particular aggregator party to aggregate, then this property can be obtained by sending the ciphertexts over an encrypted channel to the aggregator party, or alternatively by including the aggregator among the encrypting parties and letting it encrypt zero without publishing the ciphertext.

5.1 Definition

Assumptions We assume that all parties have access to a Randomness Beacon (RB) and a Public Key Infrastructure (PKI). We presuppose that each node in G is assigned an index to indicate its corresponding vertex in the graph during setup.

Corruptions We consider an adversary $\mathcal{A}$ capable of corrupting any party $\mathcal{P}_i$, up to a threshold of t parties. Once a party is corrupt, $\mathcal{A}$ takes full control of the execution of that party, meaning that it controls the actions and learns the internal state of a corrupt party throughout the execution of the protocol. The set of corrupt parties is denoted $\mathcal{C}$.

Definition 5.1 (Distributed Setup Private Stream Aggregation). A Distributed Setup Private Stream Aggregation (DS-PSA) scheme over $\mathbb{Z}_R$, where $R \in \mathbb{N}$, is defined for a set of parties $\mathcal{P} = \{\mathcal{P}_1, \dots, \mathcal{P}_n\}$ and a special party called the evaluator $\mathcal{E}$, and consists of the following procedures:

- $\mathsf{Setup}(\lambda, conf)$: On input a security parameter λ and optional configuration parameters $conf$, the procedure outputs the system parameters pp.

- $\mathsf{KeyGen}(\mathsf{pp}, i)$ On input the system parameters pp and the users index in the system i, the procedure outputs an encryption key ek_i.

- $\mathsf{Enc}(\mathsf{pp}, \mathsf{ek}_i, m_i, l)$: On input the system parameters pp, an encryption key ek_i, a message m_i and a label l, the procedure outputs an encryption c_i of m_i under ek_i.

- $\mathsf{Aggr}(\mathsf{pp}, \{c_i\}_{i\in[n]})$: On input the system parameters pp, a set of n ciphertexts $\{c_i\}_{i\in[n]}$ and a label l, the procedure outputs the sum of all plaintexts, $M \pmod{R}$.

Note that, as is often the case in PSA, our scheme returns the sum of the encrypted values modulo R, where R is a system parameter.

We say that a Distributed Setup PSA scheme is *correct* if for all $\mathsf{pp} \leftarrow \mathsf{Setup}(\lambda, conf)$, m_i, l, $\{\mathsf{ek}_i \leftarrow \mathsf{KeyGen}(\mathsf{pp}, i)\}_{i\in[n]}$, we have;

$$\Pr\left[\mathsf{Aggr}\left(\{\mathsf{Enc}(\mathsf{pp}, \mathsf{ek}_i, m_i, l)\}_{i\in[n]}\right) = \sum_{i=1}^{n} m_i\right] = 1$$

A DS-PSA scheme is secure if an adversary has a negligible probability of winning the game for Aggregator Obliviousness (AO) in Definition 5.2.

```
$\mathsf{AO}_b(\lambda, n, \mathcal{A})$

$L \leftarrow \emptyset$
$\mathsf{pp} \leftarrow \mathsf{Setup}(\lambda, conf)$
for $i \in [n]$ do
    $\mathsf{ek}_i \leftarrow \mathsf{KeyGen}(\mathsf{pp}, i)$
end for
$\gamma \leftarrow \mathcal{A}^{\mathsf{QEnc},\mathsf{QLeftRight}}$
return $\gamma \overset{?}{=} b$
```

Figure 4: The aggregator obliviousness experiment defining security for a distributed setup PSA scheme.

Definition 5.2 (Aggregator Obliviousness (AO))**.** Security is defined via the game of Aggregator Obliviousness $\mathsf{AO}_b(\lambda, n, \mathcal{A})$, $b \in \{0, 1\}$ in Figure 4. $\mathcal{A}$ denotes the adversary with access to the following oracles:

- $\mathsf{QEnc}(i, m_i, l^*)$: Given a user index i, a message m_i and a label l^*, if $(i, l^*) \notin L$ then it lets $L \leftarrow L \cup \{(i, l^*)\}$ and answers the query with $c_i = \mathsf{Enc}(\mathsf{ek}_i, m_i, l^*)$.
- $\mathsf{QLeftRight}(\mathcal{U}, \{m_i^0\}_{i\in\mathcal{U}}, \{m_i^1\}_{i\in\mathcal{U}}, l^*)$: Given a set $\mathcal{U}$ of user indices, two sets $\{m_i^0\}_{i\in\mathcal{U}}$ and $\{m_i^1\}_{i\in\mathcal{U}}$, and a label l^*, it checks if $\forall i \in \mathcal{U} : (i, l^*) \notin L$ and $\{\mathcal{P}_i\}_{i\in\mathcal{U}} \cap \mathcal{C} = \emptyset$ and no previous calls has been made to QLeftRight. If further $\{\mathcal{P}_i\}_{i\in\mathcal{U}} \cup \mathcal{C} = \{\mathcal{P}_i\}_{i\in[n]}$ it also checks if $\sum_{i\in\mathcal{U}} m_i^0 = \sum_{i\in\mathcal{U}} m_i^1$. If all checks return true, it lets $L \leftarrow L \cup \{(i, l^*)\}_{i\in\mathcal{U}}$ and answers the query with $\{c_i\}_{i\in\mathcal{U}}$ s.t. $c_i = \mathsf{Enc}(\mathsf{ek}_i, m_i^b, l^*)$.

At the end of the game, $\mathcal{A}$ outputs a guess γ of whether b equals 0 or 1.

This security definition models *encrypt-once* security, i.e. the restriction that each party only encrypts a single message per label (which is the natural usage of the scheme). This is enforced by both QEnc and QLeftRight maintaining the set L, where they store which label has been used for each user and ignores any requests of label reuse. Further, since any party has the ability to aggregate in Definition 5.1, the $\mathsf{QLeftRight}$ enforces that $\sum_{i \in \mathcal{U}} m_i^0 = \sum_{i \in \mathcal{U}} m_i^1$ when all honest users are part of the $\mathsf{QLeftRight}$ call. This prevents $\mathcal{A}$ from trivially winning the game by receiving a ciphertext for each honest user and then checking whether the output of Aggr contains $\{m_i^0\}_{i \in \mathcal{U}}$ or $\{m_i^1\}_{i \in \mathcal{U}}$.

We note that this AO-game is similar to the AO-games in [EK21; Wal+21]. The main differences are that we model corruptions as a full party takeover rather than as a key leaking oracle, and the lack of a dedicated key for the aggregator.

5.2 Construction

The protocol is described in Protocol 5.

Correctness By definition we have

$$\mathsf{DIPSAUCE.Aggr}\left(\{c_i\}_{i \in n}\right) = \sum_{i \in n} c_i = \sum_{i \in n} m_i + t_i = \sum_{i \in n} m_i + \sum_{i \in n} t_i.$$

Since G is k-regular and there exists a one-to-one mapping (bijection) between every vertex v_i and its neighbour set $N(v_i)$, there exist unique indices $i_1, \ldots, i_k$ with $i_j \neq i$ for $j = 1 \ldots, k$ such that

$$i \in \vec{J}_{i_j} \text{ for } j = 1, \ldots, k.$$

For simplicity we let i' denote any one of the indices i_j above. Furthermore, since NIKE is correct – that is, since

$$\mathsf{NIKE.SharedKey}(\mathsf{pk}_i, \mathsf{sk}_{i'}) = \mathsf{NIKE.SharedKey}(\mathsf{pk}_{i'}, \mathsf{sk}_i),$$

we also have

$$\forall K_i[\ell] : \exists K_{i'}[\ell'] \text{ s.t. } K_i[\ell] = K_{i'}[\ell']$$

Thus $\mathsf{DIPSAUCE}$ is correct as long as NIKE is correct and G is k-regular, since then all $K_i[\ell]$ will cancel out during aggregation s.t. $\sum_{i \in n} t_i = 0$.

5.3 Security Analysis

Since $\mathsf{DIPSAUCE}$ is a variation of [Wal+21], it can be proven using the same proof strategy (originating from [ABG19] and also used in [EK21]), which consists of a series of games forming a hybrid argument, and where each game changes the definition of the $\mathsf{QLeftRight}$-oracle. We recall this strategy in Table 1.

The first game G_0 corresponds to the AO_0-game where $\mathsf{QLeftRight}$ queries are answered with the encryption of m_i^0. G_3 corresponds to the AO_1-game where $\mathsf{QLeftRight}$ queries are answered with the encryption of m_i^1. Thus, if the security

Protocol 5 – DIPSAUCE

```
Setup($\lambda, conf = \{n, k, time\}$):
Generate a $k$-regular graph $G = (V, E)$ where $|G| = n$
$r \leftarrow \mathsf{Beacon}(time)$
return $\mathsf{pp} = \{n, k, G, r, \lambda, R\}$

KeyGen($\mathsf{pp}, i$):
$\mathsf{npp} \leftarrow \mathsf{NIKE.Setup}(\lambda)$
$(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{NIKE.KeyGen}(\mathsf{npp})$
Post $(\mathcal{P}_i, \mathsf{pk}_i)$ to the PKI
$r \leftarrow \mathsf{Beacon}(time)$
$\rho \leftarrow \mathsf{Perm}_r(n)$
Let $\vec{J_i}$ be the vector s.t $\forall \vec{J_i}[\ell] = j : v_j \in N(v_{\rho(i)})$, (i.e. the indices of $\mathcal{P}_i$:s neighbors in the permuted graph)
for $\ell \in \{1, \ldots, k\}$ do
    $\ell' = \vec{J_i}[\ell]$
    Wait until the PKI returns an entry $\mathsf{pk}_{\ell'}$ for $\mathcal{P}_{\ell'}$
    $\vec{K}_i[\ell] \leftarrow \mathsf{NIKE.SharedKey}(\mathsf{pk}_{\ell'}, \mathsf{sk}_i)$
end for
return $\mathsf{ek}_i = (\vec{K}_i, \vec{J_i})$

Enc($\mathsf{pp}, \mathsf{ek}_i = (\vec{K}_i, \vec{J_i}), m_i, l$):
$t_i \leftarrow \sum_{\ell=1}^{k} (-1)^{i < \vec{J_i}[\ell]} \cdot \mathsf{PRF}_{\vec{K}_i[\ell]}(l)$
$c_i = (t_i + m_i) \pmod{R}$
return $c_i$

Aggr($\mathsf{pp}, \{c_i\}_{i \in [n]}$):
$M = \sum_{i \in n} c_i \pmod{R}$
return $M$
```

of the transitions between the games hold, the adversary cannot tell the AO_0-game from the AO_1-game.

The transition from G_0 to G_1 consists of adding a *perfect* secret sharing (denoted PSS in Table 1) of zero to the threshold-sum-of-PRFs, so that all t_i are perfectly random without destroying the correctness of the scheme. Thus this transition is justified if the threshold-sum-of-PRFs produces t_i so that it is indistinguishable from randomness. Next, consider the transition from G_1 to G_2, where c_i now encrypts m_i^1 instead of m_i^0. This transition is justified since t_i is perfectly random, and thus an adversary cannot distinguish whether c_i is an encryption of m_i^0 or m_i^1. Finally, the transition from G_2 to G_3 consists of undoing the change made in G_1 (with the same security argument). We arrive at the following theorem.

Theorem 5.1. If t_i is indistinguishable from randomness for a computationally bounded adversary except with a negligible advantage, then $\mathsf{DIPSAUCE}$ is AO-secure.

Game	Definition of $\mathsf{QLeftRight}$-oracle	Argument
G_0	$t_i \leftarrow \sum_{\ell=1}^{k}(-1)^{i<\vec{J}_i[\ell]} \cdot \mathsf{PRF}_{\vec{K}_i[\ell]}(l)$ $c_i \leftarrow m_i^0 + t_i$	
G_1	$\boxed{t'_i \leftarrow \sum_{\ell=1}^{k}(-1)^{i<\vec{J}_i[\ell]} \cdot \mathsf{PRF}_{\vec{K}_i[\ell]}(l)}$ $\boxed{t_i \leftarrow t'_i + \mathsf{PSS}(0, i, n - \lvert\mathcal{C}\rvert)}$ $c_i \leftarrow m_i^0 + t_i$	t_i indistiguish-able from randomness
G_2	$t'_i \leftarrow \sum_{\ell=1}^{k}(-1)^{i<\vec{J}_i[\ell]} \cdot \mathsf{PRF}_{\vec{K}_i[\ell]}(l)$ $t_i \leftarrow t'_i + \mathsf{PSS}(0, i, n - \lvert\mathcal{C}\rvert)$ $c_i \leftarrow \boxed{m_i^1} + t_i$	one-time-pad info. theo. secure
G_3	$\boxed{t_i \leftarrow \sum_{\ell=1}^{k}(-1)^{i<\vec{J}_i[\ell]} \cdot \mathsf{PRF}_{\vec{K}_i[\ell]}(l)}$ $c_i \leftarrow m_i^1 + t_i$	t_i indisting. from rand.

Table 1: Strategy for proving AO-Security of DIPSAUCE. The change in each game is highlighted by boxing.

Proving the threshold sum-of-PRFs technique

We now prove that t_i is indistinguishable from randomness in the presence of a *static* malicious adversary, *i.e.* an adversary limited to corrupting parties only *before* the start of protocol execution. In $\mathsf{DIPSAUCE}$ t_i is generated with a threshold version of the Sum-of-PRFs technique, where for each $\mathcal{P}_j$, the selection of which PRFs to include in its sum is determined by its k-sized committee, equal to the set of neighbours to the users vertex in a random permutation of the graph G.

Proof Outline The outline of the proof is as follows. We first formalize the security of our known building blocks of NIKE and sum-of-PRFs in the context of our scheme in Lemma 5.2 and Lemma 5.3. Intuitively Lemma 5.2 states that all NIKE derived keys are private to the negotiating parties, and Lemma 5.3 states that the sum-of-PRF output, t_i, is secret to an adversary which corrupts all but one out of the parties in a sum-of-PRFs committee.

We are then ready to consider the DIPSAUCE method, where k-sized committees are selected at random from a population of n parties with a threshold t of corrupt parties. This is formalized in Theorem 5.4.

We then conclude by formalizing the indistinguishably of t_i as a consequence of the previous theorem and lemmas in Theorem 5.5.

Details of the proof Let us now detail the different parts, beginning with restating the security of NIKE in the context of our scheme, *i.e.* that each NIKE derived key derived for a committee member, is does not leak anything to the adversary for all *honest* parties in the users committee. As a direct consequence of the security of NIKE, Lemma 5.2 is true.

Lemma 5.2 (Pseudo-Random Shared Keys). DIPSAUCE.KeyGen outputs $\mathsf{ek}_i = (\vec{K}_i, \vec{J}_i)$ s.t each key $\vec{K}_i[\ell]$ is indistinguishable from randomness to a computationally bounded adversary when $\mathcal{P}_i$ and the committee counterparty $\mathcal{P}_{\vec{J}[\ell]}$ (whose index is defined in $\vec{J}[\ell]$) are both honest.

Let us also briefly restate the security of the sum-of-PRFs technique in our setting. If a key $\vec{K}_i[\ell]$ is (pseudo)-random (*i.e.* when $\mathcal{P}_{\vec{J}[\ell]}$ is honest), the output of $\mathsf{PRF}_{\vec{K}_i[\ell]}(l)$ is also (pseudo)-random. Then since t_i is the sum of all such values, a single honest $\mathcal{P}_j$ renders t_i (pseudo)-random. Thus, an adversary must corrupt *all* k parties in the committee to learn anything about t_i for the label l. We get *Lemma* 5.3.

Lemma 5.3 (Sum-of-PRFs). An adversary given l and up to $k-1$ entries in $\vec{K}_i$ has a negligible advantage in distinguishing

$$t_i = \sum_{\ell=1}^{k} (-1)^{i<\vec{J}_i[\ell]} \cdot \mathsf{PRF}_{\vec{K}_i[\ell]}(l)$$

from randomness.

By relying on just Lemma 5.3, we can only say that the protocol is secure against an adversary corrupting up to $t = k - 1$ parties. Let us therefore transfer from the standard sum-of-PRFs technique to our threshold version.

Theorem 5.4 informally states that if we randomize the committee members, an adversary corrupting up to t parties will have a negligible chance to corrupt *all* k committee members of a user with these t corruptions.

In the proof of this theorem, we first argue that the permutation of the graph is pseudorandom.

Then, as a stepping stone, we first consider the advantage of the adversary in guessing a *specific* random committee. Intuitively, if we randomize the committees for each user, a static adversary has no better strategy than to randomly guess the k users in the committee. To then put an upper bound on the advantage when attempting to guess the committee of *any* honest user, and fully prove the security of the scheme, we then finally consider an adversary which attempts to learn *any* t_i.

Theorem 5.4 (Incorruptible Committee). $\mathsf{DIPSAUCE.KeyGen}$ outputs $\mathsf{ek}_i = (\cdot, \vec{J_i})$ s.t a static adversary allowed to corrupt up to t parties, $k < t < n$, has a negligible probability in guessing $\vec{J'}$ s.t. $|\vec{J'}| = k$ and $\forall j \in \vec{J'} : j \in \vec{J_i}$, for *some* i.

Proof. The permutation ρ is determined by the output r of the randomness beacon. Since r is thus *unbiased* and *unpredictable* to a static $\mathcal{A}$, it cannot predict anything about ρ except with a negligible advantage. Then, since $|G| = |\rho|$, the adversary has a negligible advantage in determining which $\mathcal{P}_i$ is associated with which $v_j \in G$.

Consider the number of possible k-sized committees and the number of k-sized committees an adversary can form from t random corruptions. The total number of possible unordered sets of size k within the n parties is $\binom{n}{k}$. An adversary allowed to corrupt up to t out of n parties can form $\binom{t}{k}$ sets of k corrupt parties. Thus, the probability of obtaining a *specific* k-sized committee of a specific party when corrupting t out of n parties is $\frac{\binom{t}{k}}{\binom{n}{k}}$.

An upper bound on the capability to corrupt *all* members in the committee of *any* honest party for a static adversary allowed to corrupt up to t out of n parties can thus be calculated as $n \cdot \frac{\binom{t}{k}}{\binom{n}{k}}$.

In conclusion, the advantage to corrupt all committee members of some party is at most $Adv_{beacon} + n \cdot \frac{\binom{t}{k}}{\binom{n}{k}}$, which is negligible for realistic values of n, t, k (see Section 5.5 for a discussion on the values of n, t, k).

□

Now, since a static adversary cannot corrupt all nodes in the committee of any honest party (Theorem 5.4), and the sum-of-PRFs technique is secure if there is at least one honest committee member (Lemma 5.3), t_i is indistinguishable from randomness.

Theorem 5.5 (t_i Indistinguishability). Each t_i in $\mathsf{DIPSAUCE.Enc}$ is indistinguishable from randomness to a static adversary allowed to corrupt up to t parties except with a negligible advantage.

5.4 Proactively secure DIPSAUCE

The above section proves the DIPSAUCE protocol secure in the static security setting. In this section we will sketch how to construct a proactively secure version of DIPSAUCE, *i.e.* a version secure in the presence of a *mobile* adversary. Trivially, by re-running the Setup and KeyGen procedures at the beginning of each epoch, proactive security is achieved. Since the Setup and KeyGen procedures are efficient in DIPSAUCE, this modification is feasible in practice.

Modelling proactive security

We model the proactive security property according to [OY91], allowing corruptions and uncorruptions at epoch changes as follows.

Epochs Time is divided into consecutive *epochs*, where each epoch is indexed by an incrementing epoch counter.

Corruptions A mobile adversary $\mathcal{A}$ is allowed to corrupt any party $\mathcal{P}_i$.The adversary must make its selection of corrupt parties *before* an epoch is started, but will gain no information from the corrupt parties *until* that epoch is started. An adversary can additionally *uncorrupt* (leave) a corrupted party. When doing so, the adversary retains all knowledge of secrets it previously learned from that party, but has no further control of the execution of that party and learns no further secrets. The total number of corrupt parties at the start of an epoch can never exceed t. As a consequence, all parties can be corrupt during some stage of the protocol execution, but the adversary learns secrets from at most t parties during each individual epoch.

Achieving proactive security for DIPSAUCE

By discarding all secrets and starting an epoch with fresh secrets, we can achieve proactive security. For brevity, we have so far omitted how the PKI trust relation is achieved, *i.e.* how the PKI verifies that a posted public key actually belongs to the claimed identity. The caveat to discarding all secrets is how to maintain this PKI trust relation, in order to to prevent impersonations, over epochs. This problem has been studied in the literature before [OY91].Let us go into some detail of the known solutions.

When the adversary leaves a party, it still retains all variables learnt during corruptions, including any secret used to establish the trust relation with the PKI. Thus, in the mobile scenario, we must additionally prevent the adversary from using this knowledge to impersonate previously corrupt parties, during subsequent epochs whenever the party is honest. Otherwise, another honest party might derive a shared key by using a public key posted to the PKI by the adversary, believing it to be the public key of an honest party. When secrets are deleted at the end of an epoch, this includes any secret related to the trust relation with the PKI, and the trust relation is then destroyed. The challenge of achieving proactive security for

DIPSAUCE thus hinges on maintaining a trust relation with the PKI in between epoch changes.

In [OY91] two methods of maintaining such trust relations are described. In the first method, the device is assumed to be able to store a secret key that cannot be learned by an adversary corrupting the device. This can be realized using a Trusted Platform Module (TPM) [Gro11] or trusted execution techniques that provide secure storage [PS19] for a PKI relation root key.

The second method consists of updating keys by generating a new key-pair and posting the new public key signed with the previous secret key. An adversary can of course also post a new key signed with the previous key. However, in that case, since an honest party will also post a new key, the system will notice that two public keys have been published, signed with the same secret key. The system can then deduce that the corresponding device has been compromised. This assumes that the adversary cannot suppress legitimate messages reaching their destination.

Details on how to implement this in DIPSAUCE We divide the execution of the protocol into a *setup* phase comprised of the Setup and KeyGen procedures, and an *operational* phase comprised of any number of Enc and Aggr procedures.

When an epoch ends, each party erases all secrets (except the PKI relation secret), and then enters the setup phase once the next epoch begins. In this phase, it awaits the system parameters pp as output of the Setup procedure. It then calls the KeyGen procedure (using one of the PKI relation maintaining methods described above) to generate new secrets. This concludes the setup phase, and initiates the operational phase.

We arrive at the following informal theorem:

Theorem 5.6 (informal). Let there be a scheme such that the PKI will not accept more than one $(\mathcal{P}_i, pk_i)$ for each $\mathcal{P}_i$. Further, let there be at least one fresh output from the randomness beacon every epoch. Then the above transformation of DIPSAUCE is secure against a mobile adversary, corrupting up to t parties.

5.5 Parameter Selection for n, t and k

The adversary advantage (excluding the potential advantage resulting from the beacon) is calculated as $n \cdot \frac{\binom{t}{k}}{\binom{n}{k}}$ in Theorem 5.4. Table 2 shows this advantage for realistic n, t and k, where $t = n/2$ and $k = 2\sqrt{n} - 2$) in a rook's graph which is the k-regular graph which was used in our implementation. Code to calculate this advantage for different values of is available in Appendix 8.

6 Experimental Evaluation

In this section we describe our implementation of DIPSAUCE and perform a comparative evaluation of DIPSAUCE against the state of the art protocols

n	k	t	Advantage
1024	62	512	2^{-55}
2025	88	1012	2^{-78}
3025	108	1512	2^{-99}
4096	126	2048	2^{-117}
5041	140	2520	2^{-131}
6084	154	3042	2^{-144}
7056	166	3528	2^{-156}
8100	178	4050	2^{-168}
9025	188	4512	2^{-178}
10000	198	5000	2^{-188}

Table 2: Adversary advantage in DIPSAUCE with a rook's graph given by $n \cdot \frac{\binom{t}{k}}{\binom{n}{k}}$ for different values of n and a corruption ratio of 0.5.

KH-PRF-PSA and LaSS-PSA modified to utilize distributed setup as described in Section 4.

6.1 Implementation of DIPSAUCE

Let us now describe our implementation of Protocol 5. The Setup procedure is hard-coded into the source code. Here we have implemented the graph G as a rook's graph. As a consequence all n must be square numbers and $k = 2\sqrt{n-1}$. We remark that this is an implementation property, and that regular graphs for other k, n can be generated [Mer99]. The KeyGen procedure is straightforwardly implemented according to Protocol 5, using a python based PKI with a CoAP [SHB14] interface where all keys of other parties are registered, using the Drand public randomness beacon [Org22], and instantiating NIKE as ECDSA on the P-256 curve. We have implemented the Enc procedure by instantiating the PRF using AES-128. Both AES-128 and ECDSA P-256 utilizes the hardware acceleration of the CC1352 platform.

6.2 Experimental Setup

We have used the same experimental setup as described in Section 3.1 with the same suite of experiments, *i.e.* measuring the setup, (including keygen for DIPSAUCE) and encrypt procedures described in Section 3.1 and Section 4.3.

Execution times for the setup procedure are measured from the start of the process, including the time needed to transfer data, such as keys, over the network. For the encryption procedure, execution times excludes the time needed to transfer the encrypted message.

6.3 Results

Setup and KeyGen

Our evaluation shows that DIPSAUCE significantly outperforms both KH-PRF-PSA and LaSS-PSA in terms of execution time for the setup (and keygen) procedure. We show a plot of the execution times of these procedures in Figure 5. The slope of the graphs mean that DIPSAUCE will have the shortest execution time of the protocols for all number of users in the system. The execution time of DIPSAUCE grows with the number of users at rate of 3.2 ms per user, a lower rate than KH-PRF-PSA which grows with 330 ms per user and LaSS-PSA which grows with 210 ms per user.

This is due to DIPSAUCE only generating $k = 2\sqrt{n-1}$ NIKE shared secrets for n users, rather than n derived secrets as in LaSS-PSA, and LaSS-PSA in turn, as explained in Section 4.3, being more efficient than KH-PRF-PSA. Compared to LaSS-PSA, DIPSAUCE shows a speedup of 66x.

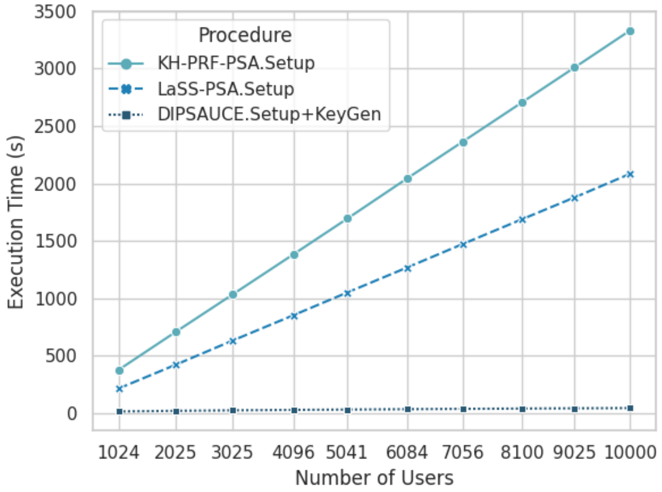


Figure 5: Execution time in seconds of the Setup procedure of KH-PRF-PSA and LaSS-PSA and the Setup and KeyGen procedure of DIPSAUCE

.

Encrypt

Our evaluation of the Enc procedures show that DIPSAUCE outperform KH-PRF-PSA and LaSS-PSA for all measured number of users in the system. We show the measured execution times of the encrypt procedure in Figure 6. LaSS-PSA and DIPSAUCE show a linear performance, depending on the number of users. The execution time of the Enc procedure grows with 0.052 ms per user for LaSS-PSA and with 0.00075 ms per user for DIPSAUCE. The speedup per user of DIPSAUCE compared to LaSS-PSA is 69x.

KH-PRF-PSA shows a constant execution time of 230 ms for any number of users in the system. Thus, it will eventually outperform DIPSAUCE. Extrapolating from the measured execution times, this will be the case when $n \approx 300000$.

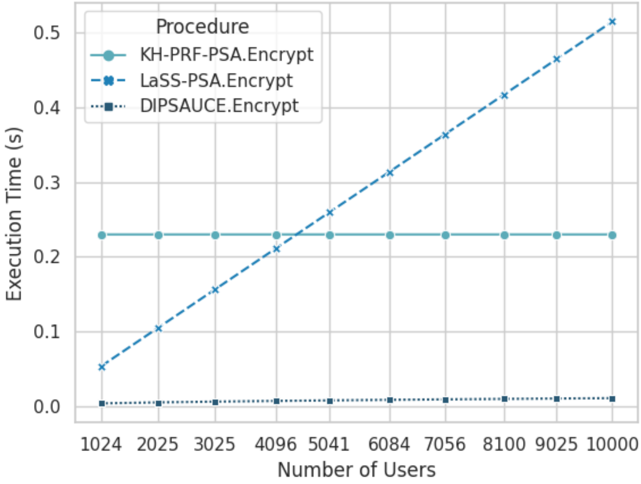


Figure 6: Execution time in seconds of the Encrypt procedure of of KH-PRF-PSA, LaSS-PSA, and DIPSAUCE.

In this paper we have evaluated two state-of-the-art PSA schemes that rely on a *centralized* setup. A centralized setup is a problem from a privacy standpoint. We have experimentally evaluated proposed ways to bypass the centralized setup, but found all solutions infeasible in a practical environment. The reason for this is the complexity of computation that grows with the number of users. We provide a formal definition of PSA with distributed setup. We suggest a new PSA scheme adhering to this definition and prove it secure. Further, we have implemented it on realistic hardware and found its performance sufficient to be deployed in practice.

Let us further elaborate on some discussion points.

Client Failures If a single ciphertext is missing at the aggregator, the security definition requires that the aggregator learns nothing. This is the point of a PSA scheme and considered a feature. However this feature can be a problem in practice if ciphertexts are lost due to client failures. This practical problem is dealt with in [CSS12], which proposes a general solution for dealing with client errors applicable to all PSA schemes, including ours. Since the setup in DIPSAUCE is efficient, another alternative to deal with client failures can be to re-execute the setup to exclude failing clients from the protocol if the failures are fairly infrequent.

Relying on an external PKI and Randomness Beacon The distributed KeyGen procedure in DIPSAUCE relies on a PKI to supply the correct public key for each user, which is often implemented as a central entity. While this is a standard assumption, we note that it is possible to *distributively audit* a PKI for correct behaviour [Mel+15; MR17; Lau14].

Analogously, DIPSAUCE relies on a randomness beacon. We therefore remark that one must be careful when realizing the beacon, in order to not introduce a trusted party. Multiple solutions for beacons which do not rely on a trusted party exist, for example beacons based on multiparty randomness generation protocols [CD20] or beacons utilizing the existing distributed security of Bitcoin [BCG15].

References

[Abd+19] M. Abdalla et al. "Decentralizing inner-product functional encryption". In: *IACR International Workshop on Public Key Cryptography*. Berlin, Germany: Springer, 2019, pp. 128–157.

[ABG19] M. Abdalla, F. Benhamouda, and R. Gay. "From single-input to multi-client inner-product functional encryption". In: *International Conference on the Theory and Application of Cryptology and Information Security*. Berlin, Germany: Springer, 2019, pp. 552–582.

[BCG15] J. Bonneau, J. Clark, and S. Goldfeder. *On Bitcoin as a public randomness source*. Cryptology ePrint Archive, Paper 2015/1015. https://eprint.iacr.org/2015/1015. 2015.

[BEK14b] C. Bormann, M. Ersue, and A. Keränen. *Terminology for Constrained-Node Networks*. RFC 7228. May 2014.

[BGZ18] D. Becker, J. Guajardo, and K.-H. Zimmermann. "Revisiting Private Stream Aggregation: Lattice-Based PSA." In: *NDSS*. Reston, VA, USA: Internet Society, 2018.

[BJL16] F. Benhamouda, M. Joye, and B. Libert. "A new framework for privacy-preserving aggregation of time-series data". In: *ACM Transactions on Information and System Security (TISSEC)* 18.3 (2016), pp. 1–21.

[Bor+22] C. Bormann et al. *Terminology for Constrained-Node Networks*. Internet-Draft draft-ietf-lwig-7228bis-00. Work in Progress. Internet Engineering Task Force, June 2022. 27 pp.

[BSW11] D. Boneh, A. Sahai, and B. Waters. “Functional encryption: Definitions and challenges”. In: *Theory of Cryptography Conference*. Berlin, Germany: Springer, 2011, pp. 253–273.

[Cas13] R. Castellucci. *libtprpg*. `https://github.com/ryancdotorg/libtprpg`. 2013.

[CC09] M. Chase and S. S. Chow. “Improving privacy and security in multi-authority attribute-based encryption”. In: *Proceedings of the 16th ACM conference on Computer and communications security*. New York, NY, USA: ACM, 2009, pp. 121–130.

[CD20] I. Cascudo and B. David. “ALBATROSS: publicly attestable batched randomness based on secret sharing”. In: *Advances in Cryptology–ASIACRYPT 2020: 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7–11, 2020, Proceedings, Part III 26*. Springer. 2020, pp. 311–341.

[Cho+18] J. Chotard et al. “Decentralized multi-client functional encryption for inner product”. In: *International Conference on the Theory and Application of Cryptology and Information Security*. Berlin, Germany: Springer, 2018, pp. 703–732.

[Cho+20] J. Chotard et al. “Dynamic decentralized functional encryption”. In: *Annual International Cryptology Conference*. Berlin, Germany: Springer, 2020, pp. 747–775.

[CM19] J. Chen and S. Micali. “Algorand: A secure and efficient distributed ledger”. In: *Theoretical Computer Science* 777 (2019), pp. 155–183.

[CSS12] T.-H. H. Chan, E. Shi, and D. Song. “Privacy-preserving stream aggregation with fault tolerance”. In: *International Conference on Financial Cryptography and Data Security*. Berlin, Germany: Springer, 2012, pp. 200–214.

[Dav+21] B. David et al. “GearBox: An Efficient UC Sharded Ledger Leveraging the Safety-Liveness Dichotomy.” In: *IACR Cryptol. ePrint Arch.* 2021 (2021), p. 211.

[EK21] J. Ernst and A. Koch. “Private Stream Aggregation with Labels in the Standard Model.” In: *Proc. Priv. Enhancing Technol.* 2021.4 (2021), pp. 117–138.

[Emu17] K. Emura. “Privacy-preserving aggregation of time-series data with public verifiability from simple assumptions”. In: *Australasian Conference on Information Security and Privacy*. Berlin, Germany: Springer, 2017, pp. 193–213.

[Gro11] T. C. Group. *TCG TPM Specification Version 1.2 - Part 1 Design Principles*. Tech. rep. Beaverton, OR, United States: Trusted Computing Group, Mar. 2011.

[GS18] P. Gope and B. Sikdar. "Lightweight and privacy-friendly spatial data aggregation for secure power supply and demand management in smart grids". In: *IEEE Transactions on Information Forensics and Security* 14.6 (2018), pp. 1554–1566.

[IEE11] IEEE Computer Society. *IEEE Standard for Local and Metropolitan Area Networks, Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)*. Oct. 2011.

[Ins21] T. Instruments. *LAUNCHXL-CC1352R1 SimpleLink™ Multi-Band CC1352R Wireless MCU LaunchPad™ Development Kit*. `https://www.ti.com/tool/LAUNCHXL-CC1352R1`. Accessed: 2022-01-21. 2021.

[JL13] M. Joye and B. Libert. "A scalable scheme for privacy-preserving aggregation of time-series data". In: *International Conference on Financial Cryptography and Data Security*. Berlin, Germany: Springer, 2013, pp. 111–125.

[Kab16] Y. Kabalci. "A survey on smart metering and smart grid communication". In: *Renewable and Sustainable Energy Reviews* 57 (2016), pp. 302–318.

[KDK11] K. Kursawe, G. Danezis, and M. Kohlweiss. "Privacy-friendly aggregation for the smart-grid". In: *International Symposium on Privacy Enhancing Technologies Symposium*. Berlin, Germany: Springer, 2011, pp. 175–191.

[Lau14] B. Laurie. "Certificate transparency". In: *Communications of the ACM* 57.10 (2014), pp. 40–46.

[LEM14] I. Leontiadis, K. Elkhiyaoui, and R. Molva. "Private and dynamic time-series data aggregation with trust relaxation". In: *International Conference on Cryptology and Network Security*. Berlin, Germany: Springer, 2014, pp. 305–320.

[Lyu+18] L. Lyu et al. "PPFA: Privacy preserving fog-enabled aggregation in smart grid". In: *IEEE Transactions on Industrial Informatics* 14.8 (2018), pp. 3733–3744.

[Mel+15] M. S. Melara et al. "{CONIKS}: Bringing Key Transparency to End Users". In: *24th USENIX Security Symposium (USENIX Security 15)*. Berkeley, CA, United States: USENIX Association, 2015, pp. 383–398.

[Mer99] M. Meringer. "Fast generation of regular graphs and construction of cages". In: *Journal of Graph Theory* 30.2 (1999), pp. 137–146.

[Mol+10] A. Molina-Markham et al. “Private memoirs of a smart meter”. In: *Proceedings of the 2nd ACM workshop on embedded sensing systems for energy-efficiency in building*. New York, NY, USA: ACM, 2010, pp. 61–66.

[MR17] S. Matsumoto and R. M. Reischuk. “IKP: turning a PKI around with decentralized automated incentives”. In: *2017 IEEE Symposium on Security and Privacy (SP)*. New York, NY, USA: IEEE, 2017, pp. 410–426.

[Oik+22] G. Oikonomou et al. “The Contiki-NG open source operating system for next generation IoT devices”. In: *SoftwareX* 18 (2022), p. 101089.

[Org22] D. Organization. *Drand - A Distributed Randomness Beacon Daemon*. `https://github.com/drand/drand`. 2022.

[OY91] R. Ostrovsky and M. Yung. “How to withstand mobile virus attacks”. In: *Proceedings of the tenth annual ACM symposium on Principles of distributed computing*. New York, NY, USA: ACM, 1991, pp. 51–59.

[PS19] S. Pinto and N. Santos. “Demystifying arm trustzone: A comprehensive survey”. In: *ACM computing surveys (CSUR)* 51.6 (2019), pp. 1–36.

[SHB14] Z. Shelby, K. Hartke, and C. Bormann. *The Constrained Application Protocol (CoAP)*. RFC 7252. June 2014.

[Shi+11] E. Shi et al. “Privacy-Preserving Aggregation of Time-Series Data”. In: *Network and Distributed System Security Symposium, NDSS* 1 (2011), p. 17.

[Szi22] J. Szigeti. *Big Unsigned Integers*. `https://github.com/SzigetiJ/biguint`. 2022.

[Tak+23] J. Takeshita et al. “TERSE: Tiny Encryptions and Really Speedy Execution for Post-Quantum Private Stream Aggregation”. In: *Security and Privacy in Communication Networks*. Ed. by F. Li et al. Cham: Springer Nature Switzerland, 2023, pp. 331–352.

[Wal+21] H. Waldner et al. *Private Stream Aggregation from Labeled Secret Sharing Schemes*. Cryptology ePrint Archive, Paper 2021/081. `https://eprint.iacr.org/2021/081`. 2021.

7 Experimental Setup

7.1 CC1352R SimpleLink

The CC1352R SimpleLink [Ins21] is a series of micro controllers (MCU) sold by Texas Instruments. Its intended application areas include: building automation, grid infrastructure, water meters, *electricity meters*, gas meters, and personal electronics. It features a 48 MHz ARM Cortex-M4F CPU, with 88KB of RAM and 602KB of ROM. It also features a wide variety of peripherals. Of special interest in this work are the hardware accelerated cryptography peripherals for AES-128, SHA256, ECC, and a TRNG. Elliptic Curves on Short Weierstrass form are fully supported and include NIST-P224, NIST-P256, NIST-P384, and NIST-P512, Brainpool-256R1, Brainpool-384RR1, and Brainpool-512R1. Elliptic curves on Montgomery form such as Curve25519 have limited hardware support. The built in TRNG has a self test required by FIPS 140.

7.2 Operating System and Software

The experiments are implemented on the Contiki-NG operating system [Oik+22], designed for constrained devices, with a its built in network stack. All hardware accelerated cryptographic operations were performed using the default drivers included in Contiki-NG. Furthermore we used the BigUint128 library [Szi22] to perform 128-bit arithmetics, and the libtprpg [Cas13] library to generate the pseudo-random permutation used in DIPSAUCE.

7.3 Communication

In our experiments we have used the IEEE 802.15.4 [IEE11] physical layer operating on the 2.4 GHz band. The network stack is the Contiki-NG networking stack with IPv6, UDP and CoAP with default settings. An RPL-border-router is required, since IEEE 802.15.4 is not supported on the laptop we used in the experiments. The RPL-border-router was run on another CC1352R device with the standard RPL-border-router application provided in Contiki-NG.

7.4 Experimental Setup

We executed the protocols on a CC1352R device, which we denote as the *Client*. The *Client* communicates with a *Server* running on a laptop. The RPL-border-router is connected to the laptop with a USB cable. The *Client* can then communicate with the *Server* running on the laptop via the border-router. The *Client* and the RPL-border-router were placed close to eachother, with their antennas facing each other to minimize packet-loss.We illustrate this setup in Figure 7.

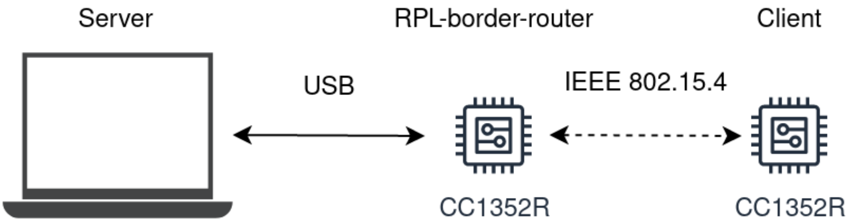


Figure 7: An illustration of the experimental setup.

.

8 Python Code for Calculating Advantage for different n,t,k

```
#rook's graph adversary advantage
import math
from decimal import Decimal

for n in [1024, 2025, 3025, 4096, 5041, 6084, 7056, 8100, 9025,
    10000]:

    # number of columns/rows
    x=float(math.sqrt(n))
    #corruption threshold
    thresh = 0.5

    k = 2*x-2
    t = n*thresh

    nc = Decimal(math.comb(int(t),int(k)))
    npc = Decimal(math.comb(int(n),int(k)))

    print("all: n =", int(n), ", k =", int(k), ", t =", int(t), ",
        advantage =", Decimal(n) * nc/npc)
```

Popular Science Summary

Popular summary in English

Connected digital computing devices have spread to virtually all aspects of society. As of 2021, there are more connected *things*, i.e. small computers, on the internet than people. These small computers power a wide variety of things in our society, from household appliances and vehicles, to power plants. One important sector that is becoming increasingly connected to the internet is manufacturing.

Industry 4.0, for example, predicts and outlines a more flexible future of manufacturing. Smaller series of custom products can be produced efficiently with distributed connected control systems without requiring complex and time-consuming retooling. Such trends in manufacturing point to a more connected, decentralized, and *agile* future.

The future of connected devices is scale and decentralization. Since devices are deployed at scale, they must be cheap to manufacture, deploy, and run. Wireless and battery-powered devices can decrease the cost of installation by 30-60%. Many of the devices being added to networks today are *constrained devices*, that is, devices with limited computational power, memory, and network bandwidth. Many constrained devices are also battery-powered and need to preserve energy.

Often the public is only made aware that a device is connected when a cyber attack is disrupting that device's operation. The move from connected computers, servers, and networking equipment to connected Smart Manufacturing, Smart Grid, and other cyber-physical systems has moved the risks of cyber attacks from loss of capability and data to the risk of physical harm, loss of property or even life. Technologies exist that mitigate the risk of connected IT infrastructures. These technologies might not, however, be suited to deployment in connected constrained devices. The limited performance of the constrained device can make such technologies too resource-intensive to be feasible. The number of sensors and actuators in a factory or a wireless sensor network can be thousands of devices. Solutions must be able to handle a large number of deployed devices. This thesis addresses the lack of efficient security protocols for new decentralized connected systems.

We have studied the efficiency of the protocols OSCORE and Group OSCORE. These protocols have been proposed to enable secure communication for constrained devices with untrusted intermediaries. We show that the protocols can be implemented efficiently, and that they are suitable for use in constrained connected devices.

Next, we used formal verification to evaluate the security properties of the WirelessHART protocol. WirelessHART has been adopted and deployed for industrial control systems. We have found that WirelessHART is secure as long as no device in a network is compromised. Furthermore, we found that a single compromised device can be used to cause further damage to a WirelessHART network. This analysis can inform WirelessHART users of the security properties of the protocol.

Furthermore, we have studied how the concept of Digital Twins can be used to create a security architecture for industrial control systems. A Digital Twin can be seen as a digital replica of a physical entity.

Next, we identified the problem of *ownership transfer* for constrained connected devices. We have designed and evaluated a protocol that enable one owner of a constrained-device network to transfer the network to another entity. This can allow a network of constrained devices to be shared between operators, without anyone having to reprogram each device that can be inaccessible.

Finally, we have proposed a novel protocol for privacy-preserving data collection and analytics. Our proposed protocol allows values, such as measurements, to be sent encrypted to an aggregator. The aggregator can then compute the sum of the encrypted values, learning the sum, but not the individual values. Our protocol allow measurements while preserving privacy.

The main finding of this thesis is that it is possible to implement and deploy secure and efficient protocols in the setting of constrained devices.

Populärvetenskaplig sammanfattning på svenska

Digitala uppkopplade enheter finns nästan överallt i samhället. År 2021 fanns det fler *uppkopplade enheter* anslutna till internet än vad det fanns människor i världen. Internet of Things (IoT) eller *sakernas internet* är ett samlingsnamn för uppkopplade *enheter* med inbyggd elektronik och ofta trådlös kommunikation. Sådana små datorer styr ett brett spektrum av saker i vårt samhälle, från hushållsmaskiner och fordon till kraftverk. En viktig sektor som blir mer och mer uppkopplad till internet är tillverkningsindustrin.

Industry 4.0 är ett koncept som förutser en framtida tillverkningsindustri som är mer flexibel än dagens. Konceptet möjliggör mindre tillverkningsserier av specialiserade produkter som effektivt kan tillverkas utan tidsödande omställning av tillverkningsmaskinerna. Denna trend inom tillverkning går mot en mer decentraliserad och lättrörlig framtid.

Många enheter som driftsätts i sådana system är så kallade *resursbegränsade enheter*. I framtiden kommer dessa uppkopplade enheter vara decentraliserade och många. Därför måste de vara billiga att tillverka, driftsätta och underhålla. Resursbegränsade enheter kan ha en eller flera av följande begränsningar: svag processor, lite minne och begränsade kommunikationsmöjligheter. Många resursbegränsade enheter drivs dessutom med batteri och behöver därför hushålla med strömförsörjningen.

Ofta blir allmänheten medveten om att ett föremål är uppkopplat först när en cyberattack stör dess funktion. Få tänkte på att kassorna på Coop egentligen är uppkopplade datorer innan utpressningsvirus gjorde dessa obrukbara. När datorer och system som styr fysiska processer såsom fabriker och energiinfrastruktur kopplas upp så introduceras nya risker för cyberangrepp. Flera uppmärksammade cyberattacker har genomförts mot industriella styrsystem och engergiinfrastruktur. STUXNET år 2012, attacken mot ett tyskt stålverk år 2014, och Triton år 2017 är bara några exempel.

Denna avhandling presenterar forskning på effektiva säkerhetsprotokoll för resursbegränsade enheter. Säkerhetsmekanismer innebär alltid effektivitetskostnader. Vi har därför undersökt prestandan för protokollen OSCORE och Group OSCORE som möjliggör säker kommunikation mellan resursbegränsade enheter.

Vår undersökning visar att protokollen är tillräckligt effektiva för användning i resursbegränsade enheter.

Vårt nästa arbete var en detaljstudie av protokollet WirelessHART. WirelessHART är ett protokoll för trådlös kommunikation i en fabriksmiljö. Vår analys visar att WirelessHART är säkert så länge ingen enhet i ett nätverk har blivit hackad. Denna analys ger användare av WirelessHART konkreta bevis för säkerheten i protokollet.

Vidare har vi undersökt konceptet *digitala tvillingar* och hur dessa kan användas för att bygga en säkerhetsarkitektur för industriella styrsystem. En digital tvilling kan ses som en digital kopia av en fysisk enhet. Vi föreslår ett sätt att konstruera en digital tvilling och att hålla den updaterad med den fysiska enheten. Vår arkitektur presenterar ett nytt sätt att designa säkra industriella styrsystem.

Vi har även undersökt säkra ägarbyten av resursbegränsade enheter. Detta är ett scenario där trådlösa resursbegränsade enheter, till exempel koldioxidmätare utplacerade i en stad, skall byta ägare. Vi har tagit fram ett protokoll som tillåter en ägare av ett sådant nätverk att överföra kontrollen till en ny ägare. Vi har bevisat säkerheten för protokollet med en teknik som kallas formell protokollverifiering och implementerat protokollet för att undersöka dess prestanda. Vårt protokoll kan möjliggöra överföringen av trådlösa nätverk av sensorer och andra uppkopplade enheter mellan operatörer.

Slutligen har vi studerat integritetskyddad datainsamling och analys för uppkopplade resursbegränsade enheter, där ett vanligt behov är central insamling av data. Data som samlas in kan avslöja information som affärshemligheter om processer i en fabrik. Vi har designat ett effektivt protokoll som tillåter krypterade mätvärden att samlas in och summeras centralt utan att individuella mätvärden avslöjas.

Avhandlingens huvudsakliga bidrag är att det är möjligt att utveckla och driftsätta säkra och effektiva protokoll för resursbegränsade enheter.