Mälardalen University Press Dissertations
No. 256

MONITORING FOR SECURING CLOCK SYNCHRONIZATION

Elena Lisova

2018

School of Innovation, Design and Engineering

ISBN 978-91-7485-378-0
ISSN 1651-4238
Printed by E-Print AB, Stockholm, Sweden

Mälardalen University Press Dissertations
No. 256

MONITORING FOR SECURING CLOCK SYNCHRONIZATION

Elena Lisova

Akademisk avhandling

som för avläggande av teknologie doktorsexamen i datavetenskap vid
Akademin för innovation, design och teknik kommer att offentligen försvaras
måndagen den 16 april 2018, 13.15 i Gamma, Mälardalens högskola, Västerås.

Fakultetsopponent: Professor Panos Papadimitratos, KTH Royal Institute of Technology

Akademin för innovation, design och teknik

Abstract

In today's society, more and more embedded computer systems are connecting. There are many different types of embedded systems including industrial networks, Internet of Things (IoT), distributed control systems, connected vehicles, etc. Most such cyber-physical systems (CPS), regardless of their specifics, have a communication part that enables data exchange between system entities and external entities. Today, many commercial systems adopt heterogeneous solutions including a combination of wired and wireless communication. Using both technologies together brings benefits in terms of flexibility and reliability, but it also imposes new challenges, such as maintaining system security. Security of connected CPS therefore becomes paramount to address.

One of the most critical properties of CPS is related to timing, as the vast majority of all CPS have real-time requirements due to interaction with a physical process, and communication therefore follows some kind of schedule with deadlines. In time-triggered networks, transmissions occur at pre-defined instants in time, but also in event-driven communication, data usefulness can be based on a timestamp, and consequently, to judge data validity and order of events, nodes need to interpret the received timestamp based on its own time. Both implementations make clock synchronization an essential network asset. Therefore, the first step in securing CPS is an investigation of ways to break clock synchronization. The next step is development of a solution that allows detection of malicious influence in the system and mitigates its consequences.

In this thesis, a threat model and a vulnerability analysis of clock synchronization is built upon IEEE 1588, a standard widely used in industry for establishing and maintaining clock synchronization. As a mitigation strategy, a distributed monitoring solution is proposed to detect if an adversary is influencing clock synchronization in the network. The monitor strategy is based on dynamic rules for switching between different network states: no adversary present, quarantine mode and attack detected. Next, game theory is used to investigate the interaction between an adversary and the monitor. Furthermore, the time chase between an adversary and the monitor is examined to see how the monitor strategy influences the outcome of the adversary actions. Safety and security interaction is also considered to see which implications the proposed security solution has on the safety domain. Finally, the monitoring approach is abstracted and analyzed for different estimations of channel reliability to investigate the applicability of the solution in different settings, and as a result a methodology for black channel state manager design is presented.

ISBN 978-91-7485-378-0
ISSN 1651-4238

Populärvetenskaplig sammanfattning

I dagens samhälle blir fler och fler datorsystem uppkopplade och sammankopplade. Det finns många typer av datorsystem, inklusive industriella nätverk, sakernas Internet (IoT), distribuerade kontrollsystem, uppkopplade fordon och mer därtill. De flesta system, oberoende av sin specifika användning, har en kommunikationsdel som möjliggör datautbyte mellan systemenheter och externa enheter. Dagens kommersiella system använder heterogena lösningar, vilket bland annat innebär kombinationer av trådbunden och trådlös kommunikation. Att använda av båda teknikerna ger fördelar i form av flexibilitet och tillförlitlighet, men det ger också nya utmaningar, såsom att upprätthålla systemets säkerhet. Säkerhet hos anslutna, uppkopplade och sammankopplade system blir därför viktig att lösa.

En av de mest kritiska egenskaperna hos denna typ av system är relaterad till tid. Majoriteten av systemen har realtidskrav och kommunikation som följer någon form av tidsschema. I tidsstyrda nätverk sker all kommunikation vid givna tidpunkter, men även i händelsestyrd kommunikation kan datas färskhet baseras på en tidsstämpel, och följaktligen, för att bedöma datas giltighet och ordning på olika händelser, måste noder kunna tolka den mottagna tidstämpeln baserat på sin egen tid. Båda förhållandena gör klocksynkronisering till en viktig egenskap i nätverket. Därför är det första steget mot att göra sådana system säkra, en undersökning av olika sätt att bryta klocksynkronisering. Nästa steg är att utveckla en lösning som möjliggör upptäckt av skadligt inflytande i systemet och mildrar dess konsekvenser.

I denna avhandling görs därför först en hotmodell och en sårbarhetsanalys av klocksynkronisering, baserad på en standard som används i industriella nätverk. Därefter föreslås en metod för distribuerad övervakning som ett sätt

att upptäcka om en motståndare påverkar klocksynkroniseringen i nätverket. Övervakningsstrategin bygger på dynamiska regler för växling mellan olika nätverkstillstånd. Spelteori används för att undersöka samspelet mellan en motståndare och övervakaren. Vidare undersöks tidsförloppet från det att en motståndare bryter klocksynkroniseringen till dess att säkerheten äventyras, alternativt till dess att övervakaren upptäcker motståndaren, för att se hur övervakningsstrategin påverkar systemets tidsaspekter. Därefter undersöks informationssäkerhets- och personsäkerhetsinteraktionen för att se vilka konsekvenser den föreslagna lösningen kan ha på personsäkerhetsområdet. Tillvägagångssättet abstraheras och analyseras för att undersöka lösningens användbarhet i olika tillämpningar, och som ett resultat presenteras en metod för att hantera otillförlitliga kommunikationskanaler.

Abstract

In today's society, more and more embedded computer systems are connecting. There are many different types of embedded systems including industrial networks, Internet of Things (IoT), distributed control systems, connected vehicles, etc. Most such cyber-physical systems (CPS), regardless of their specifics, have a communication part that enables data exchange between system entities and external entities. Today, many commercial systems adopt heterogeneous solutions including a combination of wired and wireless communication. Using both technologies together brings benefits in terms of flexibility and reliability, but it also imposes new challenges, such as maintaining system security. Security of connected CPS therefore becomes paramount to address.

One of the most critical properties of CPS is related to timing, as the vast majority of all CPS have real-time requirements due to interaction with a physical process, and communication therefore follows some kind of schedule with deadlines. In time-triggered networks, transmissions occur at pre-defined instants in time, but also in event-driven communication, data usefulness can be based on a timestamp, and consequently, to judge data validity and order of events, nodes need to interpret the received timestamp based on its own time. Both implementations make clock synchronization an essential network asset. Therefore, the first step in securing CPS is an investigation of ways to break clock synchronization. The next step is development of a solution that allows detection of malicious influence in the system and mitigates its consequences.

In this thesis, a threat model and a vulnerability analysis of clock synchronization is built upon IEEE 1588, a standard widely used in industry for establishing and maintaining clock synchronization. As a mitigation strategy, a distributed monitoring solution is proposed to detect if an adversary is influencing clock synchronization in the network. The monitor strategy is based on dynamic rules for switching between different network states: no adversary present, quarantine mode and attack detected. Next, game theory is used

to investigate the interaction between an adversary and the monitor. Furthermore, the time chase between an adversary and the monitor is examined to see how the monitor strategy influences the outcome of the adversary actions. Safety and security interaction is also considered to see which implications the proposed security solution has on the safety domain. Finally, the monitoring approach is abstracted and analyzed for different estimations of channel reliability to investigate the applicability of the solution in different settings, and as a result a methodology for black channel state manager design is presented.

To my beloved mother,
Lisova Irina

Моей любимой маме,
Лисовой Ирине Васильевне

Acknowledgments

First and foremost I would like to thank my supervisors for their continuous support. I learnt a lot through discussions with all of you and I am very grateful for your patience. Elisabeth Uhlemann, you are so generous to your students, you share experience and tips, and you are always there to support, even if it is late at night. You taught me an important criterion to navigate in life: ”it should be fun!” I want to be inspired, at least from time to time, by what I am doing and this I learnt from you. Mats Björkman, your experience is tremendous and it is amazing how much freedom you give to your students at the same time guiding them in the right direction. Thank you so much for always supporting my ideas, even when I say that in a week we submit a paper that has not been started yet. I know it was not so easy! Johan Åkerberg, thank you for your feedback and comments, they are always to the point. Thank you for assuring that tricky points in my work are covered and for being immediately ready to take over a presentation once visa issues arise.

Thanks to all lecturers whose courses I took at MDH, I learnt a lot. I consider myself lucky to encounter so many amazing people on my way. Thank you to all with whom I had luck to work together with! Marina, you always know which questions to ask to bring the trickiest point up, work with you was fun and productive. Svetlana, talking to you in Russian about work kills me, but working with you is great. You are always there to provide feedback and help, even if we re-write the whole paper in three days and two of them are a weekend. Sara, you thought me how to do things in an organized fashion, the difference between stuff and staff, and made me learn some ethics! Omar, you have such a systematic and well-based approach to everything, I learnt a lot from discussions with you, I don’t promise to list contributions as a bullet list in my papers, but I definitely do it in my head. Irfan, other people freak out hearing us discussing things as they think we are going to kill each other, well, they do not understand how to create a well-founded solution. I learnt from

you a lot: starting from how much a particular word could mean and criticality of synchronization with the example of people trying to maneuver a canon, and till how to think big and set up the right priorities. Aida, you are just amazing and I cannot find enough words to thank you, you are a continuous generator of ideas and you are always unconditionally ready to help and support. RetNet guys! We had a cool journey together and we had so much fun on the way: travel check, Bulgarian border, loosing Pablo in Moscow, guessing a normal distribution, SkyNet project and yes, of course, "Triangle!!!" It doesn't matter where life places us, I am sure we are not going to be lost.

A great thank you goes to the administrative staff: one is never lost here and always knows where to find help with any matter! People are the best part of MDH. Fikas, conference trips, badminton, and running competitions all those are treasured memories. Thank you all for being part of this wonderful team!

A bit more than four years has passed since I arrived to Sweden. I changed a lot since then, hopefully for the best. "Love hard, run fast and be kind" . . . still working on the last one. I don't regret my decision to come to Sweden even for a minute. It was not easy and straightforward and often I got caught up with routine and work, but I did enjoy the journey and I know it will make a new turn and continue further. I managed to set up my life in Sweden and here a huge thank you goes to my family back home in Russia. My mother and grandfather, aunt and cousins are always having my back regardless of which crazy idea I have in my head. I already told it before and I say it again — I dare to fly high only because of my dependable home ground! Margo and Kate, you are part of the home ground too, I am so grateful that I met you and we managed to keep our friendship through so many years. And I was extremely lucky to find a family here too, the shotgun wedding family with a traditional big chicken. Some people say I am a "baksuz", quite possible, there is always a story to tell about what has happened to me. However, I don't mind to encounter as many problems as needed to compensate the luck of having you all in my life.

This chapter is closed, the new one awaits its beginning.

Engage.

Elena Lisova
March, 2018
Västerås, Sweden

This work has been supported by the European Union's Seventh Framework Programme within the RetNet project.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

We live in an era of digitalization – both hardware and software technologies have advanced significantly and as a result, devices around us have more and more intelligence inside, they become ”smarter” and have more capabilities. A smart coffee machine can be set to prepare coffee for the moment one is getting up, order coffee beans when there is almost none left and call maintenance once it needs to be repaired. A smart car can assist its driver, make an optimal route and dynamically change it according to the current traffic situation. Among many other factors, the progress is also enabled by advances in communication technologies in the information technology (IT) field. Technologies such as Ethernet and WiFi in a combination with low cost hardware, boost the usage of network applications. However, many IT technologies do not support real-time requirements, implying that they cannot guarantee that an action is performed within a specific time range – not earlier nor later. Applications with such requirements traditionally come from the industrial domain. For example, the control process regulates humidity in a paper machine, or the conveyor belt in a discrete manufactoring plant. In industrial settings, networks with real-time requirements are supported by operational technologies (OTs), e.g, PROFIBUS [2], a standard for fieldbus communications in industrial automation. With advances in communication technologies from the OT domain, distributed networks have also been wider adopted for safety-critical domains, i.e., domains where an application failure may lead to significant harm. Thus, OTs today also include automotive, avionics and robotics applications. Some examples of real-time technologies from the automotive domain are the Controller Area Network (CAN) [3] and Time-Triggered Ethernet

(TTEthernet) [4]. The first one is broadly used for in-vehicle embedded systems. CAN is implemented on a bus, where all nodes can hear each other and there is no possibility to send a message only to one specific node, as the nature of the protocol is broadcasting. TTEthernet is a platform that complements Ethernet with time-triggered channel access so that it can be used for safety-critical applications with real-time requirements. The application area of this protocol is quite large and includes not only automotive, but also avionics, space, railway, energy etc. OT solutions are more expensive, e.g., as they often require costly hardware. However, for safety-critical applications high cost is acceptable as long a ssufficient level of safety is guaranteed.

IT and OT have developed side by side for different domains and today we observe the beginning of a technological revolution accelerated by integration of IT and OT into solutions where the best from both domains are taken. Development of Industry 4.0, smart cities, intelligent transportation and Internet of Things is enabled by OT and IT integration. For example, Industry 4.0 uses OT infrastructure within a factory and IT solutions to place its data in a cloud. This integration also enables a cooperation of geographically distributed devices. Such that a factory with OT running inside can be remotely controlled though IT. Wireless communications are gaining more and more attention in such circumstances, as they bring benefits of reducing complexity, increasing flexibility, higher mobility, etc. [5]. However, they also bring new challenges in providing the necessary security level to the system [6], as wireless links are open by their nature and require new solutions. Often in systems with wired connections, security is not considered from the very beginning, as traditionally they are designed under the assumption of being isolated. However, this assumption does not hold in emerging systems, as they tend to be more and more connected to the outside, e.g, through Internet, and more interconnected for performing more complex tasks. Even though wired communications also require security solutions, it is still considered to be more reliable. Therefore, heterogeneous communications combining both technologies are both an appealing as well as a challenging solution for today's industry [7].

Inevitably, when making systems and devices around us smarter and smarter, we shift responsibilities from us to the devices. Thus, trust in systems being dependable needs to be established between the systems and their users. Given system interconnectivity and complexity growing every day, and in order to comply with market demands as well as providing required functionality, appropriate safety and security levels need to be guaranteed.

1.1 Why Cyber-Physical Systems?

A cyber-physical system (CPS) can be defined as *"a new generation of systems with integrated computational and physical capabilities that can interact with humans through many new modalities"* [8]. In other words, a CPS is an embedded computer system which interacts with its physical surrounding. In CPSs, computations and physical processes are combined in order to provide new services enabled by the control loops between these two instances. CPSs include a wide range of applications: smart grid, advanced process control systems, distributed robotics, health systems, etc. Such systems open up new possibilities, e.g., for the next generation transportation systems they can enable vehicle remote diagnostics and management, as well as (semi)-autonomous functionalities [8]. Also the economical impact of CPSs development is significant.

Since CPSs interact with their surroundings, real-time requirements and timeliness are of essence, as there are deadlines to meet for system responses. Using an airplane as an example, its landing gear should unfold within the allowed time range, i.e., not in mid-air, before the landing procedure has been initiated, and not several minutes after the plane has landed. A physical process cannot be completetly predicted in advance and, consequently, there is a need to provide support for real-time requirements even in presence of unexpected conditions. Real-time requirements are based on correct timing in the system, meaning that the notion of time must be shared by the system components and communication needs to be deterministic from a time perspective. Many CPSs are safety-critical systems as they have a human in the loop. Therefore, one of the challenges for CPSs is assuring that timing properties of the system are dependable, i.e., acceptably safe and secure. Security, along with reliability and safety, against malicious as well as natural attacks and conditions, can be identified as one of the core challenges in CPS design [9].

In this thesis, integration of OT and IT is considered under the umbrella of CPSs; connected systems that also has outer connections and dependencies due to IT solutions.

1.2 Why Security?

Dealing with unreasonable risk of harm due to malfunctioning behaviour of technological systems is addressed under the umbrella of functional safety [10]. System functions should be designed and performed with an acceptable safety level. However, depending on a particular application, safety techniques can

vary. For example, a robot in a factory should have a fail-safe state, i.e., stop button that shuts it down, and an airplane should have safe operational state, i.e., it shall to continue to operate even in a presence of a system failure. However, since systems are not isolated anymore, but interconnected, security risks arise. A system cannot be safe if it is not secure, e.g., if someone can tamper with control commands of an autonomous car, it can start accelerating instead of braking. Safety engineering and security engineering addressing safety and security problems respectively have traditionally been developed separately as most safety-critical systems have been isolated. With more and more CPSs connecting to the surrounding ecosystem, the risk to harm that caused intentionally is increasing. Hence, there is a need to synchronize safety and security engineering such that the unreasonable risk of harm due to either malfunctioning or malicious intent is adequately addressed. More and more research is done attempting to bring together of safety and security. From a security perspective, the adoption of IT technologies for OT is a double-edged sword. On the one hand, such a mix of IT and OT technologies allows industrial applications to keep up with the growing level of network complexity and subnetwork interconnections. On the other hand, the introduction of IT technologies in OT environments significantly increases the security risks, and novel security frameworks need to be developed to meet industrial requirements. Among these, timeliness, availability, reliability and heterogeneity can be defined as the main requirements and related to the vast majority of networks, especially considering safety-critical applications [11]. Adoption of wireless solutions for OT applications is just starting. Because of the open nature of communication, security aspects of wireless links are usually considered from the beginning of the development process. However, wireless solutions are traditionally considered not reliable enough. One of the targets of the security research in an industrial context is to bring safety and security to acceptable levels for networks containing both wired and wireless solutions.

Given safety as the main requirement for today's systems that tend to have increasing complexity and interconnections with other systems, security becomes a necessary property that needs to be not only provided but also structured and argued in a clear way. It is especially the case when humans are involved in the loop and there are direct physical interactions between humans and systems, as such a setting makes the system safety-critical. For instance, an autonomous car needs to be secure, i.e., there should be enough confidence that its security level is adequate and in line with the current state-of-the-art of existing threats, attacks and system vulnerabilities. Condifence can be achieved by building a corresponding assurance case. An assurance case is documented

as a clear and well-structured argument to assure the particular property of the system [12]. Such assurance case is then reviewed by the regulation bodies to establish its validity and whether it communicates sufficient confidence that the system exhibits the specific property. Assurance case can be built for a system property such as safety, security or ethics.A systematic way to build an acceptable level of confidence in system security is to collect evidences and arguments over adequacy of security measures in a security assurance case [13].

Security can be presented through security objectives, such as confidentiality, availability, reliability, integrity etc. depending on a concrete use case [14]. For safety-critical applications, availability and reliability is of great importance, as their failure can cause an application failure and consequently lead to a hazard. There are many aspects to consider when addressing system security. Following a top-down approach, relevant system assets, i.e., things that need to be protected, are identified first. Next, the system is analyzed for its vulnerabilities, i.e., weak spots, that can be exploited to get to the assets. Then, relevant threats that can exploit the vulnerabilites are identified. Later, security objectives that prevent or mitigate threats need to be formulated and assured to guarantee a certain system security level.

Along with new capabilities CPSs also impose new challenges and this thesis considers security of CPSs as an emerging challenge to address, as security risks are becoming a showstopper for deployment of CPSs, and deployment of appropriate security solutions is considered an increasingly important requirement [15].

1.3 Why Timing as an Asset?

Is it important to have a precise clock or to know how long a specific action takes or to be able to determine data age? For someone on an uninhabited island, clock precision is not important at all. A person in hurry to catch a train requires higher precision between his or her clock and the train's time, together with a good estimate on the time needed to get to the train station and the latest updated traffic information. Thus, the answer depends on how critical the situation is, how many actors are involved and if there is a necessity to get information form outer sources. Coming back to systems and networks, timing properties includes latency, i.e., the time needed to transmit a message between nodes; clock synchronization precision, i.e., the difference in time between different nodes' clocks; data freshness, i.e., its age.

Timing properties are important for time-based cooperation of system nodes,

e.g., following a time schedule for sending messages; time-based mechanisms, e.g., time based access to a communication medium; and for handling data freshness. The latter implies establishing of a correspondence between data coming from within the system or from outer connections at a particular point in time. Two paradigms of network communication are time-triggered [16] and event-driven [17, 18] communications. The first one requires scheduling and performing certain actions at certain points in time. For instance, a robot that needs to preform actions according to time points set by a user, i.e., the robot's time should be the same as user's, or Time Division Multiple Access (TDMA) used as a medium access method. For event-driven networks, there is no schedule to follow, however the notion of time is still important, as for some applications data freshness and event ordering are important. For example, a controller may need to align its decisions not only with data from sensors, but also with this data timestamp, as in dynamic systems data freshness is an input for decision making.

One of the existing visions today for future technologies is Time-Sensitive Networking (TSN). The task group of IEEE 802.1 working on TSN develops standards that complement existing ones to enable real-time capabilities of Ethernet switches by introducing such features as system-wide clock synchronization, redundancy, and traffic preemption. TSN targets to support real-time requirements with minimal latency. For such solutions, clock synchronization is a fundamental building block.

To be able to meet deadlines and support real-time requirements, to cooperate and follow schedules, network participants need to share the same notion of time, i.e., they need to be synchronized. Being synchronized means being able to determine the age of data coming from outer dependencies or systems and to perform an action at a particular moment of time. In cooperative systems, systems synchronization facilitates a common perception of the current operational situation. Thus, security aspects of clock synchronization in a CPS need to be considered to be able to claim acceptable security and safety levels for users interacting with the CPS.

Let us assume clock synchronization was broken, i.e., nodes in the network have significantly different opinions about time, and assume further that this leads to a system failure, e.g., the system has a time-triggered architecture and a control command was not forwarded in time due to a clock synchronization failure. Would a clock synchronization failure be considered as a possible cause when analyzing the incident? Is it easy to trace it back? If the right cause of a failure is not identified, no countermeasure would be applied to prevent it in the future. One possible way to answer positively to these questions would

be to monitor clock synchronization parameters in run-time.

1.4 Why Monitoring?

Having a monitor in the network of embedded systems has several benefits; it can increase the level of safety in the system, e.g., by detecting component failures, and/or the level of security in the system, e.g., by detecting an attack being deployed. A monitor used for safety reasons detects system states leading to possible hazards, with the goal to prevent or at least mitigate the consequences. A monitor used for security purposes detects malicious actions in the network with the goal to suppress adversary actions and their consequences. Thus, the actions required from a monitor to cover both domains are similar and thus a monitor can be used to increase the dependability for safety-critical embedded systems.

Monitoring is not a new idea in regards to security, it was proposed to be used already around 30 years ago [19]. However, the approach is still appealing and promising as security solutions must be dynamic with respect to continuous updates and further refinements. More importantly, a system cannot be assessed as being secure once and for all. Security as a system property can be provided and guaranteed to a specified extent only for the current state-of-the-art, as new system vulnerabilities are constantly exposed and new attack techniques are continuously being developed [20]. Security is dynamic by its nature and requires run-time updates and refinements. To this end, a tool such as a monitor can provide a run-time assessment of the current system state.

Since developing a security case from scratch every time there is an update is not feasible, the challenge of handling updates in a smart way within a security case needs to be addressed. The notion of a dynamic assurance case has already been proposed in the safety domain [21], e.g., the introduction of a set of rules for updates and a set of monitors for establishing a link between the system and a confidence structure within the safety case. It is likely that such techniques can be adopted, further developed and complemented with other relevant solutions to handle a dynamic security assurance case. Thus, even though monitoring of clock synchronization is a small piece of a possible monitor of a system operational state, such reasoning and its possible extension is a motivation to consider monitoring further.

1.5 Problem Formulation

Reliability and timeliness are usually considered as necessary properties of CPSs. Message transactions have deadlines even for non-critical applications, as after some point in time the carried information loses its value. To be able to cope with such deadlines, network participants need to share the same notion of time, i.e., be synchronized. It makes clock synchronization a common essential asset for the vast majority of CPSs. There are several broadly used standards for establishing and maintaining clock synchronization in CPSs and most of them do not have the necessary level of protection or do not have any security solutions at all. Therefore, protecting clock synchronization can be considered as an important step on the way to securing heterogeneous CPSs. Given the motivation and the problem described above, the following goal is stated for the research:

Goal: To enhance the security of cyber-physical systems, while ensuring that the adopted security solutions are selected and designed according to the specific requirements of safety-critical real-time applications, in terms of timeliness an reliability.

To reach the goal, the following research questions must be addressed:

Research questions:

- RQ1: What are the main system assets and security objectives of cyber-physical systems?
- RQ2: How can the main system assets be protected and the needed security objectives provided in the presence of real-time deadlines?
- RQ3: How will protection of the system assets influence system safety and dependability?

1.6 Thesis Scope

Having the thesis goal and corresponding research questions formulated, this section demonstrates how the thesis scope was narrowed down. The general rationale is presented in Fig. 1.1 and below are the steps for specifying the topics which are considered in more detail.

The first question presented in the upper left in Fig. 1.1 is to answer which area that is considered in the work, what exactly this work targets to secure.

There are many groups of systems with specific requirements, architectures and application areas: industrial networks require high reliability and are used in plants; control systems can be a layer within industrial systems with higher security requirements; supervisory control and data acquisition (SCADA) systems are similar to control systems but with wider variety of means to interact with a factory; systems with incorporated cloud computing enable usage on demand of storage place and computation power; software-defined networking (SDN) systems have data and control planes separated; Internet of Things (IoT) systems consist of devices that communicate and exchange data; services systems that use system representation into services for their analysis; distributed control systems have distributed resources and intelligence; etc. These groups are partly overlapping and sometimes it is tricky to explicitly define the border. In this thesis, networked systems with outer dependencies, with a human in the loop and thus safety-critical, with physical process considered along with computation capabilities, and with real-time requirement are considered and the term CPSs was chosen to best describe their properties.

Given CPSs as a considered area, the next step depicted in the upper right in Fig. 1.1, is to specify which security aspect of CPSs the work is focused on. Access control allows limiting the amount of users and specifies their rights with regard to interacting with CPSs. Considering industries, physical security, e.g., security guards can be of importance. Having hardware components in a system, one can consider means of protection on the hardware level, e.g., digital systems design flows and physical attacks. Software security in its turn deals with cyber-attacks. Computer forensic is related to derivation of evidences in computers and digital storage media. Communication security addresses protection of communications between network nodes. Communications is a crucial part of CPSs as it enables information exchange between system entities and outer dependencies. Therefore this aspect was chosen as being critical to consider for the security of CPSs.

However, CPSs communications still incorporates a lot of aspects as shown in the lower right in Fig. 1.1. Communication channel availability is important to be able to communicate, i.e., to access the channel. Communicational channel reliability is addressed in safety-critical applications, as its analysis is required to guarantee timely deliver of data including control information. For transmitted data, its integrity is of importance for system safety. Real-time requirements are tightly connected with communications properties. However, in this thesis only one aspect of CPSs communications is considered and it is timing properties, as a necessary aspect to secure to enable real-time requirements support which is not traditionally considered in security as real-time systems

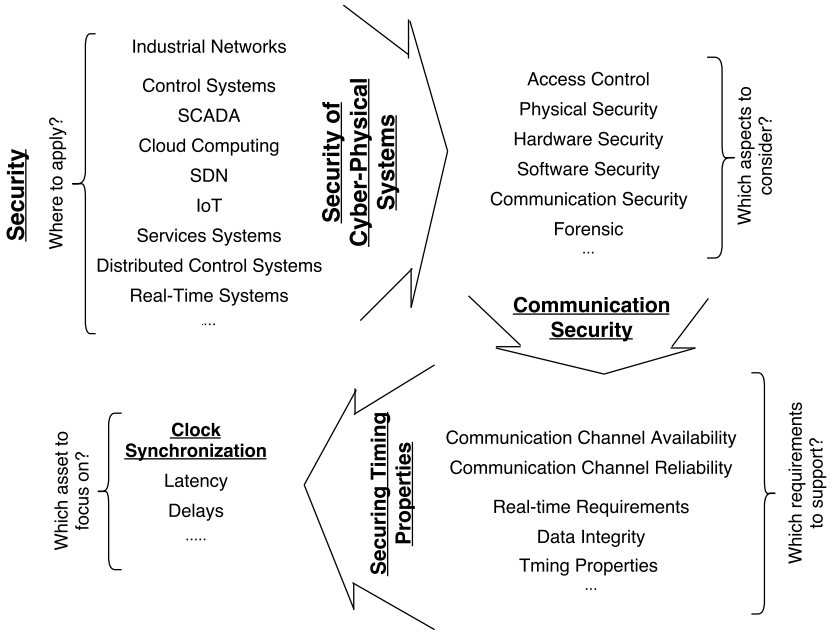

Figure 1.1: The thesis scope

are relatively new.

Coming to timing properties grouped in the lower left in Fig. 1.1, latency can be considered. There are applications requiring low latency, e.g., for a propagation of an emergency signal. Networks delays can be analyzed to make them more deterministic. Clock synchronization is required when the system components, which are addressed from a communications point of view as network nodes in this work, need to share the same notion of time within the system and/or with external connections of the system. Thus, securing clock synchronization of CPSs is the topic of this work. This is a new aspect of security which emerges as increasingly important in the era of connected things.

Once clock synchronization is broken, the whole network may be disrupted. Moreover, clock synchronization is a common component of many systems and clock synchronization is related not to a particular application, but to system communication infrastructure and its logic for decision making based on event ordering. There are not so many approaches for establishing

and maintaining clock synchronization, especially standardized ones. Thus, from an adversary point of view, it makes sense to invest resources in developing a way to influence clock synchronization, as the technique can be re-used in different applications. From a research perspective, it is interesting to see how clock a synchronization failure can be adequate handled in time in order to ensure the required system safety and security levels.

1.7 The Overview of Thesis Contributions

This thesis includes four main contributions answering the three research questions formulated above. These contributions are presented below.

1.7.1 Contribution A: Threat Model and Clock Synchronization as an Asset

Approaching security, an adversary, i.e., an agent with malicious intent, needs to be considered, and further a way of analyzing the system needs to be formulated. One of the contributions of this thesis is the development of a threat model, where a threat is understood as a potential situation when a system vulnerability, i.e., a flow in system design or implementation, is exploited by an attack in order to get to a system asset, i.e., an object that need to be protected. The model is accomplished by putting together the adversary model, i.e., capturing important characteristics of the adversary, and goals, i.e., the overall aim behind an attack, as well as the system specification, identifying the most important assets and system vulnerabilities. Even for non-safety-critical applications with real-time requirements, there are still deadlines for messages transactions. This implies that the network should have some kind of schedule that is able to ensure that all messages meet their deadlines. In order to follow the schedule, the network participants should share the same notion of time, i.e. they should be synchronized. Clock synchronization is thereby an essential asset for cyber-physical networks. Consequently, it is also an appealing and fruitful target for an adversary, since the same technique can be applied to several different networks with different use cases. Therefore, it is more likely that the adversary will invest additional resources into its breaking as the attack technique can be reused.

This contribution is presented in Chapter 3 and in the following published papers:

- Elena Lisova, Elisabeth Uhlemann, Johan Åkerberg, and Mats Björkman, "Towards secure wireless TTEthernet for industrial process automation applications," in Proc. *19th IEEE Conference Emerging Technologies and Factory Automation (ETFA)*, Barcelona, Spain, Sep. 2014.
  My contribution: Proposed the method for threat modeling and wrote most of the text in the paper.

- Elena Lisova, Elisabeth Uhlemann, Wilfried Steiner, Johan Åkerberg, and Mats Björkman, "A survey of security frameworks suitable for distributed control systems," in Proc. *IEEE International Conference on Computing and Network Communications (CoCoNet)*, Trivandrum, India, Dec. 2015.
  My contribution: Was the main driver of the paper, wrote the major part of the text and proposed the evaluation technique for the frameworks.

1.7.2 Contribution B: Breaking Clock Synchronization and Proposing to Use Monitoring

In order to protect clock synchronization in CPSs, an investigation on how a possible intruder can break it, is needed. To this end, IEEE 1588 [22] was chosen as the use case due to it prevalence in industrial networks. IEEE 1588 is a standard for establishing and maintaining clock synchronization in a centralized manner. The next contribution is thus a detailed investigation of the IEEE 1588 standard for clock synchronization from a security point of view. Two well-known approaches are proposed and combined: the possibility to break IEEE 1588 synchronization by conducting a delay attack, and the possibility to perform a man-in-the-middle attack such that delay attacks can be introduced, using the so-called address resolution protocol (ARP) poisoning technique. This technique exploits a vulnerability in ARP, a protocol used to translate between IP and MAC addresses. The proposed attack is verified using Automated Validation of Internet Security Protocols and Applications (AVISPA) [23], a tool used for security protocol evaluation that has several inbuilt types of security analyses. The attack consequences and applicability for cyber-physical networks could then be investigated.

Next, clock synchronization was considered in a bigger scope and an attack tree analysis (ATA) was conducted. ATA is a threat modeling technique, in which the root of the three is an adversary goal and the leaves indicate possible threats, i.e., ways to achieve the target. The intension is to investigate possible ways of breaking clock synchronization, i.e., considering not only a

delay attack.

In addition, an overview of possible mitigation techniques and countermeasures to protect against the identified threats is given. A monitor, collecting and analysing specific statistics about the data traffic, is introduced. The collected data can be used beneficially for many purposes, but its applicability from a security point of view is evaluated. Environmental conditions are used, as a possible way to detect the adversary influence in the network. In particular, an investigation of the possibility to use network monitoring and analysis for detecting an attack on clock synchronization provided by the IEEE 1588 standard is evaluated. A monitor can collect the needed statistics derived from the data traffic patterns and detect an intruder and its influence on the network. Also, an additional Relaxed mode is introduced as a way of coping with the consequences of synchronization being broken. Several ways of performing delay attacks and its corresponding influence on the network traffic statistics are demonstrated using the AVISPA tool and mathematical analysis. All solutions were also investigated for suitability of use in CPSs.

This contribution is addressed in Chapter 3 and is presented in the following published papers:

- Elena Lisova, Elisabeth Uhlemann, Wilfried Steiner, Johan Åkerberg, and Mats Björkman, "Risk evaluation of an ARP poisoning attack on clock synchronization for industrial applications," in Proc. *IEEE International Conference on Industrial Technology (ICIT)*, Taipei, Taiwan, Mar. 2016.
  My contribution: Was the main driver of the paper and wrote the majority of the text. In addition, the author proposed the possible ways of breaking clock synchronization, i.e. combining ARP poisoning and delay attack, and performed the evaluation using the AVISPA tool.

- Elena Lisova, Marina Gutiérrez, Wilfried Steiner, Elisabeth Uhlemann, Johan Åkerberg, Radu Dobrin, and Mats Björkman, "Protecting Clock Synchronization, Adversary Detection by Network Monitoring," *Journal of Electrical and Computer Engineering*, Hundawi,v. 2016, 2016.
  My contribution: Wrote the part of text concerning the system model, the attack description, the possible countermeasures and the attack evaluations as well as one of the mitigation techniques, namely using environmental conditions. In addition, the author proposed the scenario of the attack, and did the evaluation of the attacks in AVISPA.

1.7.3 Contribution C: Monitor Stategy and Indicators

Monitoring allows detecting and reacting in run-time to changes in the monitored system, which is necessary for maintaining an appropriate system security level. Monitoring only traffic patterns is not sufficient for detecting a delay attack, thus the concept of monitoring calculated offsets is introduced. Along with changing what to monitor, the monitor structure is developed further and networks states reflecting different confidence levels of system being under attack are proposed. The next contribution is a formal analysis of the interactions between an adversary and the network monitor applying a game theoretical approach. The game is formulated by defining its players, together with their strategies and payoffs depending on the outcome. In order to analyse the interactions between the adversary and the network monitor in this work, different probability functions for action strategies are introduced. Furthermore, adaptive rules for switching between network states are considered. The analysis allows a comparison of adversary strategies from a detection point of view. Finally, the detection coefficient is introduced as a metric for adversary exposure in the network. This is used to reason about the efficiency of different types of attacks and to predict the adversary behavior.

The approach was then developed further and several indicators were proposed as prerequisites for switching between network states. The behavior under a delay attack for these indicators was investigated to see how effective they are and where they can be applied. Furthermore, a sensitivity analysis is conducted in order to generalize indicator behavior. The challenge of the time chase between an adversary and the monitor was addressed especially to see which factors can influence the outcome of the chase, i.e., who is faster, the monitor in detecting an attack or an adversary in reaching its goal to breach security. A hybrid approach was also proposed to cope with some types of delays whivh are undetectable without a reference from a clock of the same time domain, i.e, clocks being synchronized to the same grand master.

This contribution is described in Chapters 5 and 6 and is presented in the following published papers:

- Elena Lisova, Elisabeth Uhlemann, Wilfried Steiner, Johan Åkerberg, and Mats Björkman, "Game theory applied to secure clock synchronization with IEEE 1588," in Proc. *IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communications (ISPCS)*, Stockholm, Sweden, Sep. 2016.
  My contribution: Was the main driver of the paper, wrote most of the text, proposed the idea of applying game theory to model interactions be-

tween an adversary and the monitoring system and formulated the game settings.

- Elena Lisova, Elisabeth Uhlemann, Johan Åkerberg, and Mats Björkman, "Delay attack versus clock synchronization — a time chase," in.Proc. *IEEE International Conference on Industrial Technology (ICIT)*, Toronto, Canada, Mar. 2017.
  My contribution: Was the main driver of the paper, proposed to use Neyman-Pearson criterion for detection, evaluated the attack detection by analyzing a part of the offset introduced by the adversary and proposed the discussion on how to shape the monitor strategy.

- Elena Lisova, Elisabeth Uhlemann, Johan Åkerberg, and Mats Björkman, "Monitoring of clock synchronization in cyber-physical systems: a sensitivity analysis," in Proc. *The International Conference on Internet of Things, Embedded Systems and Communications*, Gafsa, Tunis, Oct. 2017.
  My contribution: Was the main driver of the paper, conducted the analysis and applied it to the use case to evaluate the results.

1.7.4 Contribution D: Monitoring Applications

The next contribution targets to extending the security solution application area. First, implications of the security solution on system safety were considered. Interdependencies between safety and security were investigated, focusing on threats towards breaking clock synchronization and its possible effects on system safety in complex systems of systems, as an example of modern CPSs. The contribution is a demonstration of safety and security overlapping and a possible way to address it while targeting a common high-level goal, e.g., protecting clock synchronization in an autonomous quarry with several cooperating vehicles. Joint safety and security clock synchronization argumentation using goal structuring notation was presented to illustrate the implications of clock synchronization protection on system safety. Monitoring allows to detect an anomaly that can be originated in both safety and security domains, thus can be an example of a tehcnique incorporating consideration of both. The methodology of approaching monitor design was tested in a bigger scope for channel reliability estimation. A design of channel state manager that detects attacks and failures originated in communication channel was investigated.

This contribution is presented in Chapters 4 and 7 and in the following published and submitted papers:

- Elena Lisova, Aida Čaušević, Elisabeth Uhlemann, and Mats Björkman, "Clock synchronization considerations in security informed safety assurance of autonomous systems of systems," in Proc. *the 43rd IEEE Annual Conference of the IEEE Industrial Electronics Society (IECON)*, Beijing, China, Oct. 2017.
  My contribution: Wrote the part of text, which proposed to use clock sycnhronization to illustrate safety and security dependencies, refined the threat model for clock sycnhronization and connected it to the considered hazard.

- Elena Lisova, Svetlana Girs, Irfan Šljivo, "Designing a reusable black channel state manager for cooperative systems", submitted to *the 23rd International Conference on Reliable Software Technologies Ada-Europe 2018*, Lisbon, Portugal, Jun. 2018.
  My contribution: Wrote the part of text which brought security into consideration and partly developed the methodology.

1.8 Research Process and Methodology

The two main methodological approaches in science are the deductive and inductive ones. The deductive method implies the following chain of steps:

$$\text{theory} \rightarrow \text{hypothesis} \rightarrow \text{observations} \rightarrow \text{confirmation/rejection.}$$

The confirmation of the observation does not imply the correctness of the hypothesis, but supports it. The inductive approach has the opposite logic:

$$\text{observations} \rightarrow \text{pattern} \rightarrow \text{hypothesis} \rightarrow \text{theory.}$$

In this case an observation is the idea base for theory building. Another way to classify methods is to divide them into reductionism and wholism [24]. The first group implies a simplification of the considered problem and examination through different components of the problem. This is suitable for approaching many complex tasks through the set of smaller ones. However some problems which are related to a complex system cannot be decomposed and, thus, require a wholism approach. The mentioned approaches could be combined to address the research goal at their best.

The research goal is formulated based on the background knowledge in the deductive way, however a mix of deductive and inductive techniques is used while addressing it. Conclusions correspond with the last step of the method

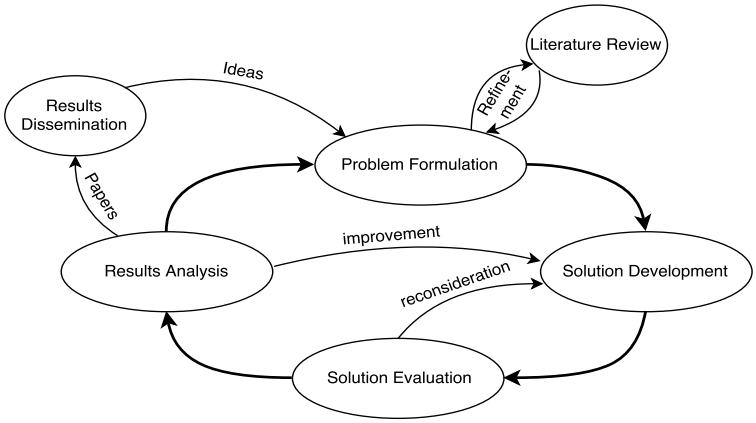

Figure 1.2: Research process

chains presented above. Thus, conclusions drawn from simulations represent a deductive way, while conclusions drawn from a theoretical analysis in their turn represent an inductive way. A decomposition of the research goal into *RQ*1-3 in Section 1.5 corresponds with the reductionism approach. However, *RQ*3 addresses implications of the solution on system level, thus it belongs to wholism.

When conducting research in a specific area, e. g., computer science in this case, it is important to understand the related research methods and be able to apply them. This research work aims to construct new methods and approaches based on already existing ones by adjusting them and filling in missing parts. Thus, it falls into a constructive research group [25]. This type of research implies solving problems by providing new constructions or solutions that have theoretical and practical contributions. The process of developing such a construction as adopted within this thesis is presented in Fig. 1.2. This is an adaptation of the general framework for research methods in the computing area presented by Holz et al. [26] that consists of a repetitive circle with four main steps: specifying what to be achieved; specifying where the data can be collected and how it can be used; choosing the way to process the data; and finally evaluating the results and analyzing whether the goals have been achieved or not. For the research work conducted in this thesis, the process circle was adjusted as follows:

- The research work starts with *Problem Formulation*: defining the goal of what is to be achieved, specifying the questions that are needed to be answered on the way and relating the research problem to state-of-the-art in the area. The latter brings refinement of the research goal or even its reformulation, thus several inner iterations could be needed in this step. This thesis was done in collaboration with industrial partners, thus their input was also used to formulate the goal and relate it to the state of practice.
- The next step is *Solution Development*, where, based on existing techniques and appropriate analysis of these, a proposal is developed regarding how to solve the problem or a part of it. Usually, a solution could be presented as a combination of small parts that, assembled together, build the required solution.
- A proposed solution needs to be analyzed using related performance metrics in the step *Solution Evaluation*. Based on the results, some reconsideration and further development could be needed. During this step the following tools were used: AVISPA [23] and OMNeT++ [27] to conduct attack and solution evaluations and Matlab for calculations and simulations.
- Finally, the last step in the circle is *Results Evaluation*. This step is needed to analyze if what has been derived is what was actually wanted and formulated in the goal of step one. As an outcome, possible refinements can be identified and applied to the goal, after which the circle can be repeated until the results are satisfying and the goal has been achieved. The outcome of this step is knowledge dissemination via publications. Discussions at conferences and paper reviews give an input for the next problem formulation and possible improvements for the developed solution.

Shaw [28] formulates the three following research components: research setting, research approaches/products and validation techniques. The author also presents the common options for the computer science field. Research settings represent how the research goal can be decomposed into easier steps leading to the goal in a composition. Thus, research setting corresponds with the Problem Formulation step. Hence, research questions identified in Section 1.5 could then be classified as: *RQ*1 — *Characterization*, as the considered system needs to be characterized and analyzed to answer the question; *RQ*2 — *Methods/Means*, as this it a search for the solution, *RQ*3 — *Generalization*, as

the answer to this question provides solution implications. Research approaches/products show how the research problem is addressed, thus it corresponds to the Solution Development step. Based on the products, the approaches adopted in this thesis fall into four categories: *Qualitative or Descriptive Model*, as the research includes threat model formulation and channel state manager design; *Techniques* as it presents development of a monitoring approach for clock synchronization anomaly detection; *System*, as possible dependencies of the developed solution with other system components and properties are considered; and finally, *Analytic Model*, as the solution development is done via formulating a model of the monitor. Validation techniques correspond with the Results and Solution Evaluation steps and include *Persuation*, as the analysis is based on logical reasoning and observations, and *Analysis*, as the solution is analyzed in order to be evaluated.

1.9 Ethics Aspects

Today, technologies are developing with tremendous speed, but they are still only tools in the hands of humans. Thus, how these tools are used depends on formal regulations, laws, and informal regulations, ethics. Ethics could be defined as a set of moral principles that result in decision making. Many ethical challenges have emerged with the increase of autonomy of smart and interconnected systems and our reliance on the decisions made by the software that runs inside them. These ethical challenges are related to the different system properties such as safety, security, privacy etc. For example, many safety related ethical issues have been raised regarding autonomous vehicles, such as "the trolley problem" [29]. Similarly, the privacy of personal data and the possibility to harness information about individuals from different sources has lead to some ethical challenges [30].

This thesis considers security and thus touches upon ethical aspects. However, no direct interactions with humans are considered in this research work, apart from representing an adversary as an actual person. In addition, the thesis considers industrial information security rather than private personal data. Thus, the author believes that it is enough to address ethics within the thesis by narrowing it to decision awareness presented within an ethics assurance case [31]. While the system properties such as safety and security are often regulated by domain-specific and property-specific standards, the ethics of the system is conventionally left implicit. The standards usually require, either implicitly or explicitly, an assurance case to be provided, where an assurance case

aims to reason about system trustworthiness. It is common to build a safety or security assurance case. The lack of coverage of the ethics in such regulations is due to the fact that different individuals and societies are governed by different ethical principles. Hence, the companies provide a statement with the product regarding certain ethical issues (e.g., privacy) to communicate and assure the user how some of the ethical challenges have been handled within the system. The assurance to the standardization body presents how the requirements stated in the corresponding standard are fulfilled. However, the assurance to the user does not have clearly stated requirements or a standard to follow as they are challenging to standardize. The requirements imposed by the standards are based on the ethical conduct of the society, hence they focus on the general well-being. Those ethical attributes not covered by the standards remain for the user to judge based on their personal moral code. The idea behind the ethics assurance cases is to cover the ethical attributes that remain to the user to judge, which aligns with the agent-centered deontological approach [32] where stress is on the actor's moral rules. The framework includes a technique to identify which parts of the system are ethically challenging from the agent-centered ethics approach. The focus is on the safety and security critical systems and aligning the technique with the existing safety and security analyses. The last chapter of this thesis presents a methodology to design a channel state manager to estimate channel reliability, within the proposed approach, but system response is left out of scope and assigned to be handled by a system state manager. The decision of the system state manager could be ethically challenging if the user of the system is not provided with an acceptable level of awareness. However, the channel state manager as is, is currently not involved in ethical challenging decisions and, thus, ethical aspects are not considered further.

1.10 Thesis Outline

The rest of the thesis is organized as follows.

- **Chapter 2 - Background and Prior Work**: This chapter presents background and related work relevant for the thesis. Background includes security and safety overview, i.e., terminology, analyses, brief overview of game theory and monitoring approaches along with clock synchronization main approaches.
- **Chapter 3 - How Can Clock Synchronization Be Broken?**: This chapter presents a threat model and proposes a way to break clock synchro-

nization, namely a combination of ARP-poisoning and a delay attack. It also presents a more general overview of possible attacks on clock sycnhronization and their classification.

- **Chapter 4 - How Can a Broken Clock Synchronization Influence System Safety?**: This chapter considers implications of clock synchronization being broken on system functional safety on an example of an autonomous quarry and argues towards a joint safety-security assurance required for systems of systems.
- **Chapter 5 - How to Predict the Adversary Behaviour?**: This chapter proposes a monitor concept for detecting clock synchronization anomalies and presents a formulation of interactions between the monitor and an adversary within game theory terms. The game is analyzed further to make an assumption about adversary actions.
- **Chapter 6 - How to Define the Monitor Strategy?**: This chapter presents grounds for the monitor strategy choice. The indicators suitable for detection of clock synchronization being broken are proposed and analyzed to derive a way to threshold them and evaluate their effectiveness.
- **Chapter 7 - Can the Approach Be Applied in Different Settings?**: This chapter proposes to abstract the monitor idea from considering only clock synchronization and apply it for estimating a reliability of a communication channel for cooperative systems. The design methodology of a black channel state manager is presented.
- **Chapter 8 - Conclusions and Future Work**: This chapter concludes the work by summarizing its contributions and discussing the future work.

Chapter 2

Background and Prior Work

In this chapter, the necessary background for positioning the thesis is presented, i.e., security and safety as system properties and clock synchronization as a technology are introduced.

2.1 Approaching Security

According to Glossary of Key Information Security Terms by NIST [33], security can be defined as ”a condition that results from establishment and maintenance of protective measures that enables an enterprise to perform its mission or critical functions despite risks posed by threats”. This thesis considers system security, which implies that policies and regulations on higher levels are left outside the scope. Avizienis et al. [34] present security as a composition of Confidentiality, Integrity and Availability, (CIA) attributes. In this thesis, the term ”security objective” is used in the same sense as attribute. In the scope of communication security, CIA could be enriched by authentication, authorization, auditability, non-repudiability and third-party protection [6]. Kunze et al. [35] connects system security to being ”free of danger”, however the term danger is not well defined. Security deals with adversaries and malicious intent. Security solutions should be dynamic with respect to continuous updates and further refinements. More importantly, a system cannot certified as being secure once and forall, i.e., security assurance and certification are continious processes. Security as a system property can be provided and guaranteed to a specified extent only for the current state-of-the-art in terms of threats and

attacks, as new system vulnerabilities are constantly exposed and new attack techniques are continuously being developed [36]. Thus, run-time monitoring is a promising solution as a part of security system.

2.1.1 Terminology

To approach security, its terminology needs to be defined first. Security starts with the system definition. Once we know what the system is and its functionality, relevant system assets could be identified.

Each system has a set of **assets**, i.e., values that need to be protected against an adversary. A **vulnerability** of a system is a flaw in the system that allows an adversary to impose a threat targeting one of the system assets. In order to perform adequate security analysis, the characteristics of the possible adversary should be collected in an **adversary model.** First, the assumptions about adversary goals should be made. Next, the value of system assets related to the adversary goals along with the system structure and design are used to estimate possible adversary properties to complete the model [14]. A **threat** in its turn can be defined as "the potential source of an adverse event" [33]. A **threat profile** is a collection of possible threats from a potential adversary that reflects possible adversary goals and is based on an adversary model. A **threat model** is a result of the threat profile and the system collation. The overall process of threat model derivation is called threat modeling. Threat modeling takes as input system assets, system entry points, i.e., how it can be accessed, and through the system characterization gives as output identified threats that should be analyzed [37]. System characterization includes defining scenarios of how the system can be used, identification of dependencies and assumptions. Once a high-level threat has been identified based on an adversary model, knowledge of the adversary goal, and existing vulnerabilities in the system, the threat can be decomposed into possible attacks leading to this threat. Fig. 2.1 demonstrates how an attack realizes a potential threat and exploits an existing system vulnerability in order to break a system asset.

2.1.2 Security analysis

Compared to safety, security is a less structured domain. The notion of certification is not directly applicable to security, as a system cannot be deemed secure forever due to the dynamic nature of security. Thus, security standards are presented as a collection of guidelines, e.g., Cybersecurity Guidebook for Cyber-Physical Vehicle Systems [38], or focused on security evaluation, e.g.,

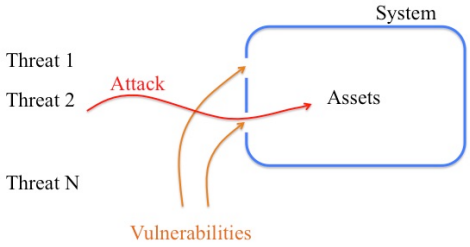

Figure 2.1: System security terminology

Common Criteria [39]. The first step in making a system acceptably secure for the current state-of-the-art in attacks and vulnerabilities is threat modeling. Similar to As Low As Reasonable Practicable (ALARP) [50], an appropriate level of system security is defined when the resources required to be invested to breach it are compared to a value of the system assets.

One of the threat modeling techniques is based on the considered threat types, namely Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elicitation of Privilege (STRIDE) [40].The threats considered within STRIDE are opposites of the following security objectives [41]: authenticity, integrity, non-repudiation, confidentiality, availability, and authorization. Threats related to the groups mentioned above are systematically considered within this approach.

ATA is another formal technique used in security for threat modeling [42]. It provides a systematic way to describe system security based on varying attacks. The root of such a tree is an adversary goal corresponding to an identified high-level threat. The root has leafs that presents possible ways of achieving the root, i.e., possible threats. Leafs can be connected via AND or OR gates. Each leaf can be decomposed further until the desired level of granularity is achieved. ATA is an appealing technique for system analysis as it has logic similar to Fault Tree Analysis (FTA) and, thus, a combined version of these two methods can be used for a joint safety and security analysis. Ruijters et al. present an uniform meta-model allowing to merge ATA and FTA in attack-fault trees (AFT) [43]. The developed tool provides bidirectional transformation between a joint AFT model and independent models. A feedback between different models exists, and thus it belongs to the true safety-security co-analysis approach at the level of hazards/threats. The AFT model can be

transferred to UPPAAL [44] for quantitative analysis purposes, e.g., reliability or expected cost. Kumar and Stoelinga propose an approach handling AFTs with dynamic gates allowing to consider more complex multiple step scenarios [45]. The authors present a possible transformation of dynamic gates into stochastic timed automata that allows to use the UPPAAL model checker for statical model checking. The approach includes quantitative analysis of AFTs and consideration of several safety-security scenarios, e.g., *as-is scenario* and *what-if scenario*, leading to identification of the most risky scenarios and selection of the most effective countermeasures.

System-Theoretic Process Analysis for Security (STPA-Sec) is an approach that deals with securing software intensive systems against intentional disruptions [46], where instead of focusing on threats coming from adversary actions, one focuses on controlling system vulnerabilities. In doing so, STPA-Sec identifies and enforces required constraints on insecure control actions that put the system in vulnerable states when exposed to intentional or unintentional disruptions. Just like its safety counterpart, System-Theoretic Process Analysis (STPA) [47], STPA-Sec follows four basic process steps: *i)* establish the systems engineering foundation for the security analysis; *ii)* identify the control actions that constitute a threat to system security; *iii)* use control actions to create security requirements and constraints; *iv)* identify scenarios in which the security constraints are violated. In general, this approach does not give any concrete information about what specific countermeasures that should be taken, but provides a useful way to identify scenarios that should be the focus of security experts when securing complex systems and especially system of systems (SoS).

2.2 Approaching Safety

Safety can be defined as ”freedom from unacceptable risk” [48], where risk is defined by the probability of harm to occur and its severity. Safety can be considered as a dependability attribute [34]. Accordingly, security and safety are not overlapping, however for complex systems that communicate with each other and have humans in the loop, these two properties are interdependent. A safety-critical system is a system for which the consequences of its failure could lead to a significant loss in terms of money, damage to environment or human life. It is impossible or unreasonable in terms of invested resources to achieve absolute safety, thus an acceptable level of system safety is required. The safety level is connected to the risk level, thus, risk assessment is used in

conjunction with safety analyses. Given the system definition, a hazard analysis is conducted. A hazard can be defined as a set of conditions that in a combination with other environment conditions lead to an accident [47]. A hazard is not directly related to a failure, as a failure can be defined as an interruption in providing the required functionality, i.e., a situation when the delivered services deviates from specified one [34]. Thus, a failure does not necessary lead to a hazard, and a hazard can occur without a failure contributing to it. A failure is a consequence of an error within the system, where an error can be defined as a part of the system state that may propagate itself into a corresponding failure by reaching the system boundary [34]. An error is in its turn is a consequence of a fault.

Within the safety domain, to achieve a required safety level, the system development process should follow a safety standard. There are different standards for different domains, and ther ones used in this thesis are briefly detailed below. However, the underlying steps for these standards are similar. Once the system is defined, i.e., the boundaries separating the entities performing the specified task and its environment are defined, hazard analysis and risk assessment is performed. An example of hazard analysis is Hazard and Operability Study (HAZOP), a systematic search for hazards based on considering effects of normal parameter deviations. The complementary technique for safety assessment analyzing events causing hazards is FTA. The root of the tree is a high-level hazard that is iteratively decomposed further into single failures via AND or OR gates. Further developed intermediate, undeveloped intermediate and basic events are depicted using rectangular, rhombic and circular shapes respectively [49]. An acceptable level of risk can be established by applying the ALARP approach [50] that suggests to stop reducing a risk once the cost of its further reduction is greatly disproportional to the possible obtained gain.

An assurance case is required to assure that a desired system property is provided. Usually, safety-critical systems require a safety assurance case for certification. The assurance case consists of structured arguments and evidences supporting them [51]. To avoid being ambiguous, a special unified notation is used for reasoning, namely Goal Structuring Notation (GSN) [52]. It consists of the following components: *a goal*, i.e., what should be achieved, *a strategy*, i.e., how it should be achieved, *a solution*, i.e., means for the goal realization, and *a context*, i.e., assumptions and conditions. GSN is used to formalize arguments and also to demonstrate logic in their decomposition into corresponding sub-goals and correspondence with the way arguments are addressed, i.e., conditions in which arguments are valid and strategies required to fulfill them. The main symbols used in the language are demonstrated in

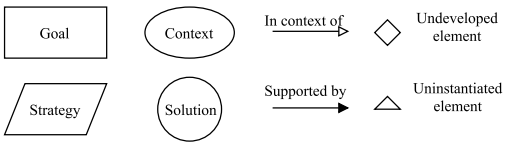

Figure 2.2: GSN notation

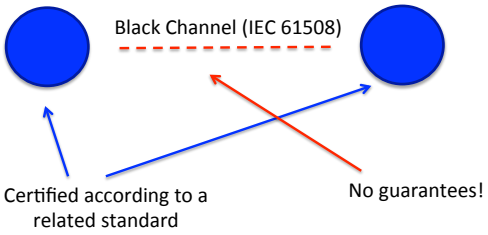

Figure 2.3: Black channel model

Fig. 2.2.

Acknowledging that not all communication channels can be developed to high safety integrity levels, the IEC 61508 standard [10] describes the model of a black channel. Within this model as it is shown in Fig. 2.3 all the safety integrity is achieved by the smart end-point nodes of such channels. A communication channel itself is not designed and validated according to IEC 61508, meaning no guarantees can be provided for such channel characteristics or properties. The standard suggests, that in case of using a black channel, measures should be deployed to define its failure performance. As such components need to be developed and assured to high levels of safety integrity, it is often recommended by the safety standards to use semi-formal approaches for their verification. A safety contract is a pair [53]:

$$\mathrm{C} = (\mathrm{A}, \mathrm{G}) \text{ of } \{\text{Assumptions, Guarantees}\},$$

that formally characterizes the assumed context of the system and its obligations. The guarantee defines the component's behaviour given that its environment meets the conditions stated in the assumptions. Assumption/guarantee contracts can be used to verify and assure the behaviour of such smart nodes [54]. Safety-relevant components such as the black channel smart nodes

are described via safety contracts, a type of contracts that capture safety-related behaviours of the component.

ISO 26262 [55] is an automotive industry functional safety standard that introduces the notion of safety element out-of-context (SEooC), a component developed explicitly for reuse, i.e., out of the context of a particular system, and according to the standard. Since the context of the system in which SEooC should operate is not fully known, the standard proposes that assumptions on the system context should be explicitly captured, and the component should be developed according to those assumptions. When SEooC is being reused in a particular system, the assumptions should be validated. These standard guidelines have been enriched with the assumption/guarantee contracts [56] so that the system assumptions are captured in the preliminary contracts, that are further refined during the development of the SEooC and supported by evidence assuring the confidence in such contracts.

2.3 Clock Synchronization

Many systems are required to have consistent time within its nodes, i.e., to have the same notion of time within the system border and be aligned with entities it has connections to. A way to keep time in the network consistent is to provide a clock in every node and align these with each other. If each clock is ideal the problem is solved, however in reality every clock has each natural drift. The drift cannot be eliminated completely, though clocks have lower drift. Thus, the clock times cannot be exactly the same, and instead the aim is keeping their difference bounded. Therefore, clocks need a periodical correction in order to keep the difference between their times, called offset, bounded. The periodical correction of a clock time is presented in Fig. 2.4. The dash-dotted line represents the ideal case, when the clock time is identical to the real time. The green solid line shows the dependency of the clock drift compared to the real time according to a linear approximation. The clock time is corrected periodically, once every resynchronization interval (RI). The clock is called synchronized if the offset is bounded, i.e., the green line stays within the dotted lines.

To avoid failures, caused by inability of applications to meet their deadlines, clock synchronization protocols are used. Depending on the system architecture and its application, the notion of ”real time” to which nodes are synchronized can vary. If the system is closed and does not have outer dependencies, its nodes need to be synchronized to each other without a particular

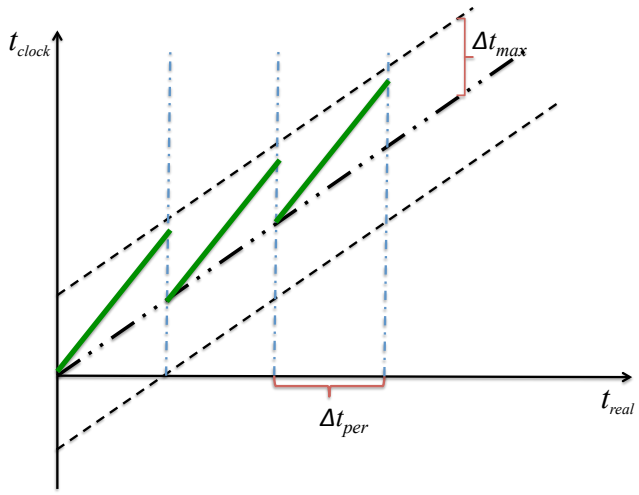

Figure 2.4: Periodical correction of clock drift

reference to the environment. In this case two approaches are possible a distributed approach, i.e, when the clocks are synchronized to a commonly derived average time, and a centralized approach, i.e., when clocks are synchronized to a chosen grand master. However, given dependencies, the system nodes may need to be synchronized with external nodes from which data or control information is coming to or from.

Network delay and clock synchronization precision are tightly connected, as to establish synchronization regardless of a particular approach, clocks need to exchange time information. This exchange is done using communication channels, meaning that an algorithm needs to handle network delay in some way. It can be done by a delay approximation or by assuming that delay is symmetrical in both directions. The latter implies measuring a round-trip delay for the offset estimation. Also the delay influence could be minimized by a corresponding correction of the exchanging messages in the intermediate nodes between the sender and the receiver.

2.3.1 Distributed Approach

In a distributed approach for clock synchronization establishment and maintenance, there is no master or reference clock, and clocks synchronize to an average time on which they agree on. Hence, this is a suitable technique when a system is isolated or when the required clock precision is not very high. Welch and Lynch [57] investigate limits of possible precision for clock synchronization in distributed systems. Further boundaries for clock synchronization precision has been investigated by Kopetz and Ochsenreiter [58] along with advantages and drawbacks of continuous and instantaneous synchronization. An average among several clocks can be obtained by clocks announcing their time and calculating the mean, however this solution is vulnerable to an extreme value or to a Byzantine clock, i.e., a clock that announces different times to different nodes. Lamport and Melliar-Smith [59] demonstrate that m Byzantine clocks could be tolerated by $3m + 1$ clocks within the interactive convergence algorithm and $2m + 1$ clocks within the interactive consistency algorithm. Within the first algorithm, while calculating an average time, clocks ignore values that differ from theirs for more than a defined threshold. The second algorithm implies that every clock builds a vector of how it sees other clocks and sends it out. Once all clocks perform this operation, every clock can build a matrix out of such vectors and immediately identify a Byzantine clock. However, this algorithm implies bigger communication overhead and unforgeable digital signatures. Fault-tolerant clock synchronization is a mature research domain, however centralized approaches are in the scope of this thesis as these are more suitable for safety-critical applications that require high precision.

2.3.2 Centralized Approach

The centralized approach for clock synchronization is a common choice for networks, in which high time precision is important, e.g., safety critical applications with real-time requirements. There are a few protocols broadly used in industry, the main ones being the Network Time Protocol (NTP), the Precision Time Protocol (PTP) and the IEEE 802.1AS standard. These are presented below with a more detailed representation of PTP as this protocol is used for the analysis in the thesis.

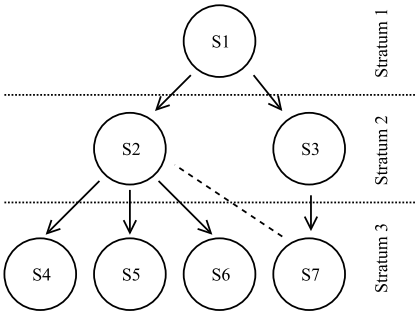

Figure 2.5: NTP subnetwork

NTP

NTP was designed by Mills [60], and its current version 4 is presented in IETF RFC 5905 [61]. This protocol uses User Datagram Protocol (UDP) and operates between an NTP server, which is supposed to have a reliable clock, and its clients. Timestamping is performed at the application level, through software timestamping. According to NTP, primery servers synchronize directly to an external time reference. Then, secondary servers are synchronized to primery ones and so on. The number of hops from the direct time reference for a server is the number of the stratum group for this server. An example of a subnet in NTP, i.e., a hierarchical group of servers connected to the same time reference, is presented in Fig. 2.5. NTP implies having a backup path for synchronization (the dotted line in Fig. 2.5), e.g., if node $S3$ fails then node $S7$ will be synchronized with node $S2$ instead. Backup paths can optionally be used for information exchange without synchronization paths reconfiguration due to a failure.

NTP uses timestamps exchange between a server and its clients to define corresponding round-trip delays and clock offsets. The protocol implies a two-way handshake initiated by the client as it is presented in Fig. 2.6. The measured clock offset is calculated according to the protocol based on two measured delays $a = t_2 - t_1$ and $b = t_3 - t_4$ as following:

$$\sigma_{meas} = \frac{a + b}{2}. \tag{2.1}$$

RFC 5905 also provides public a key authentication mechanism descrip-

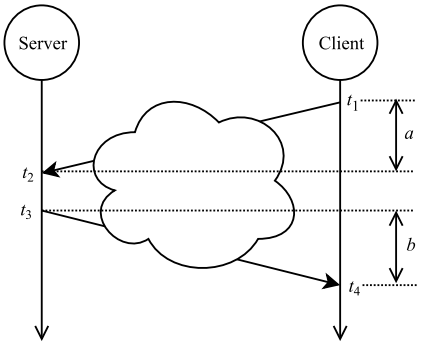

Figure 2.6: NTP message exchange

tion and discusses a set possible attacks on the protocol. The assumption used in the calculations of the offset is that $a \approx b$ and thus the protocol is vulnerable to a delay attack [62]. NTP is a scalable and effective algorithm with low communication overhead and a good candidate for clock synchronization algorithm when the required precision of synchronization is in the millisecond range [63].

PTP

In the PTP, one of the nodes is chosen as a grandmaster (GM), and the rest of the nodes are referred to as slaves S_i. Every slave clock executes the best master clock algorithm to choose its grandmaster. If the current grandmaster fails, the slave does not support the clock synchronization protocol in the interval before a new grandmaster is chosen. Below, the synchronization protocol is detailed to see how the offset is calculated.

Given two clocks, A and B in a network such as they have been synchronized at a moment in the past, i.e., $t_A = t_B$, at a later moment, the time values provided by these clocks will have drifted apart according to:

$$|t_A - t_B| = \sigma. \tag{2.2}$$

This drift is caused by the non-ideality of the clocks and environmental conditions, e.g., heat affecting the frequency of the oscillators differently at the two clocks.

Slaves are synchronized to the grandmaster, such that the differences in time values provided by the local clocks in the nodes, as expressed in (2.2) is bounded. This is expressed in the synchronization condition:

$$|t_{S_i} - t_{GM}| < \sigma_{max} \tag{2.3}$$

that should be constantly preserved. Here, σ_{max} is a parameter that should be chosen so that the application requirements are satisfied. The minimum value of σ_{max} can be calculated as:

$$\min(\sigma_{max}) = 2 \max(\rho_i) R_{int} \tag{2.4}$$

where ρ_i are the clock drifts of the clocks in the network and R_{int} is the re-synchronization interval. The value of the offset used to correct the slave clock is calculated applying the protocol depicted in Fig. 2.7. The clock synchronization protocol consists of a series of messages being exchanged and time-stamped between the grandmaster and the slave in order to gain enough information to calculate the offset. This process is repeated periodically every re-synchronization interval R_{int}. The message exchange is as follows:

1. At $t = t_1$ according to the grandmaster time, the grandmaster sends a synchronization message, (`sync` in Fig. 2.7) containing t_1 to the slave.

2. At $t = t_2$ according to the slave time, the slave receives the `sync` message. Now both t_1 and t_2 are recorded in the slave.

3. At $t = t_3$ according to the slave time, the slave sends out the delay request message (`delay_req` in Fig. 2.7). t_3 is also recorded in the slave.

4. At $t = t_4$ according to the grandmaster time, the grandmaster receives the `delay_req` message.

5. At $t > t_4$ according to the grandmaster time, the grandmaster sends the delay response message (`delay_resp` in Fig. 2.7) containing t_4 to the slave. When the slave receives the `delay_resp` message, lastly, t_4 is recorded.

Finally when all the time-stamps have been collected by the slave, the offset, σ_{meas}, can be calculated according to

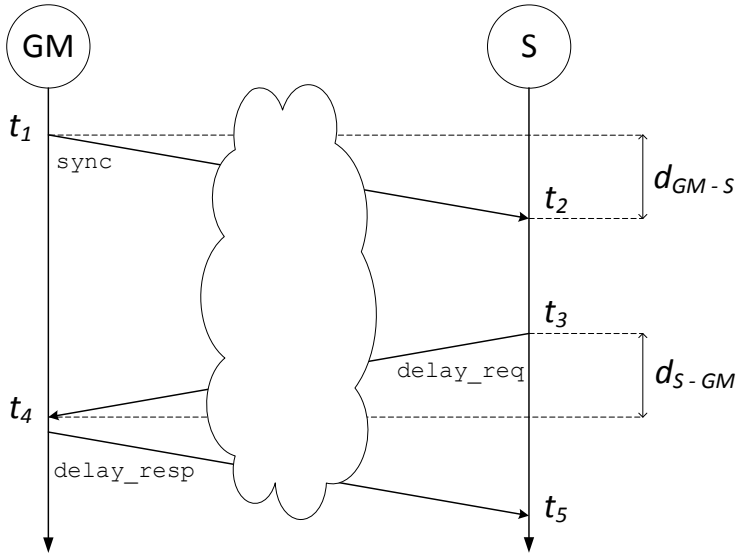

Figure 2.7: Message exchange in PTP

$$\begin{aligned} d_{GM\to S} + \sigma_{meas} &= t_2 - t_1, \\ d_{S\to GM} - \sigma_{meas} &= t_4 - t_3, \end{aligned} \tag{2.5}$$

where $d_{GM\to S}$ and $d_{S\to GM}$ are the transmission delays of a message going from the grandmaster to the slave and from the slave to the grandmaster respectively. Now if we assume that the transmission delay is symmetric, i.e., it is the same in both directions, then $d_{GM\to S} = d_{S\to GM} = d_0$ and the measured offset is:

$$\sigma_{meas} = \frac{1}{2}((t_2 - t_1) - (t_4 - t_3)). \tag{2.6}$$

Ideally, this value of the measured offset reflects the difference between the grandmaster clock and the slave at the moment when the offset is measured

$$t_S - t_{GM} = \sigma_{meas}. \tag{2.7}$$

IEEE 1588

PTP is part of the IEEE 1588 Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems [64]. The protocol operates between slaves and a grand master, to whom slaves are synchronizing, thus it has a similar hierarchical structure as NTP. The prime application area is industrial automation, as the protocol enables precise clock synchronization over heterogeneous (in the sense of equipment) networks. Within PTP several clock types are defined:

- *ordinary clock*: a clock with one port that serves as a grandmaster or as a slave for another ordinary or boundary clock.
- *boundary clock*: a clock with several ports, can be both a grandmaster for slave clocks and a slave for another boundary or ordinary clock.
- *transparent clock*: a clock that measures the time taken for a PTP message to transit through it and provides this information for the message recipients.
- *peer-to-peer transparent clock*: a transparent clock that also provides corrections for the propagation delays of the link via which PTP messages that are received.

In the IEEE 1588 standard, the equation for offset calculation is more complicated, as there are parameters compensating the propagation delays. For the sake of simplicity they are omitted here, as they do not change the logic of the protocol and do not make any significant difference for conducting a delay attack.

In addition, the standard defines two possible operation modes and two modes for synchronization messages exchange. The following operation modes are possible: end-to-end and peer-to-peer. In the first mode, clocks get information about the delays in links from the exchange of synchronization messages each time they want to make a correction. In the second mode, this exchange of messages is performed for all links regularly and without relation to the clock correction events. Each time a clock wants to correct its time, it has information about all delays with all neighbours. The end-to-end operational mode is suitable for networks where it cannot be guaranteed that all devices in the network support IEEE 1588. The peer-to-peer mode implies that the IEEE 1588 standard is supported by all devices in the network. All above mentioned synchronization message exchanges can be performed in two or in four steps.

In the second variant, follow-up messages are used to provide more precise time-stamps.

A profile within PTP is a set of specified options and attributes that allows a particular organisation to configure the protocol according to its requirements, e.g., in order to support a specific application. A PTP profile that provides synchronization with sub-ns accuracy is called White Rabbit (WR) [65]. Apart of configuring PTP, WR also defines implementation-specific functionalities that allows provision such high precision.

The initial version of the IEEE 1588 standard does not have any security services. Based on this the authors of [62] show the effects of a delay attack on the IEEE 1588 Precision Time Protocol (PTP). This thesis work can be considered as an extension of the paper mentioned above, as a way of conducting a delay attack is proposed together with a broader rangge of mitigation techniques. In release 2008, Annex K was added to the standard to provide a set of security solutions [66]. However, the amendment provides only a very limited set of services: guidelines for message integrity protection and group source authentication. These security solutions do not help against a delay attack, as in this case the adversary does not need to change the messages or create new ones - it simply delays them. Mizrahi applied a game theoretical approach to analyze the delay attack influence on clock synchronization [67]. As a result, the author proposed to use a multiple-path approach to prevent and mitigate delay attacks. However, this approach is not compatible with IEEE 1588.

802.1as

IEEE 802.1as [68] is part of the IEEE 802.1 TSN set of standards that contains a time synchronization description for a specific IEEE 1588 profile. IEEE 802.1as contains standards for Audio Video Bridging (AVB) to transport high quality multimedia traffic over time sensitive networks. According to 802.1as, clock synchronization is established in grandmaster-slave manner, where the grandmaster is elected and is a root of a spanning tree of nodes synchronizing to it. The main objective of this protocol is precision and it ensures that two synchronized or time-aware systems that are six or less hops away have a synchronization precision $\leq 1\mu s$. To achieve this, the protocol provides media-dependent delay measurements. Even though the original claim about precision refers to quite small networks, M. Gutiérrez [69] et al. show that even for a network with 100 hops, the precision is in the range of $2\mu s$.

2.4 Security Solutions for Clock Synchronization

Security issues related to clock synchronization have been pointed out in earlier works. Threats to clock synchronization include packet manipulation, packet delay manipulation, replay attack and others [70]. For instance, the security amendments in the current version of 1588 have vulnerabilities allowing to reset a security association and perform a replay attack [71]. It was also shown that clock synchronization can be broken by a combination of an ARP poisoning attack and a selective delay attack [72]. Implications of a delay attack on different synchronization protocols were investigated [62]. However, a delay attack is only one example of how clock synchronization can be broken compared to a more general analysis of clock synchronization, namely ATA, presented in this thesis.

Communication can be considered as a component of a system and clock synchronization is a property of communication. The need to consider communication and its implications on system dependability is broadly recognized, for instance the IEC 61784-3 standard [73] is specifically addressing industrial communication networks. In this standard, communication error types, e.g., unacceptable delays that can be caused by clock synchronization being broken, are mapped with safety techniques, e.g., for delay handling time-stamping and time expectation are used, i.e., packets are time-stamped and nodes know when to expect them. Safety aspects of communications are highly connected to a particular implementation and are considered for safety-critical applications, e.g., nuclear plants [74]. However, malicious actions, i.e, security related, are left out of the standard. Therefore, considering security implications on safety and, consequently, on communication dependability, advances the state-of-the-art.

Dependability of communication can be improved by providing redundancy, an approach coming from the safety domain [75]. However, it can only mitigate a malicious disturbance once an attack is detected or its consequences are visible. Therefore, monitoring can be used to detect malicious and unintended disturbances in a network, preferably before their consequences are obvious, or guide the mitigation strategy, i.e., switching routing paths to redundant ones, if the default ones are under attack. There are works coming from the security domain targeting network monitoring and attack detection for a vehicle infrastructure [76] or general computer networks [77]. However, they do not consider security solution implications on safety and dependability.

This work provides a taxonomy of attacks targeting clock synchronization that can be used for building its proper security solution. The solution con-

sidered in this thesis is distributed monitoring, which obviously cannot detect all possible attacks, but even in its current simple configuration, it can enhance clock synchronization dependability. Moreover, the stress is on security implications of clock synchronization protection on dependability and also on claiming the necessity of appropriate security solutions to assure system safety.

2.5 Game Theory

Game theory is a mathematical tool that allows formal analysis of possible interactions between game players and connections with payoff functions and game outcomes. This is helpful when investigating a decision-making process of a problem formulated as a game. Therefore, correctly applied with valid assumptions allowing to formulate the game, it helps to predict the outcome and most probable behavior of players [78]. This approach is widely applied in analyzing security interactions as it is possible within this theory to consider parties with contradicting interests. Depending on application requirements, the game can be zero-sum if the payoffs of players are balanced and sum up to zero, dynamic or static depending on the number of rounds, and complete or incomplete depending on the knowledge of the players about each others' strategies and actions [79].

The case of applying game theory to wireless ad hoc network analysis is investigated by Srivastava et al. [80]. The authors explain the basics of the theory and particularly reason about choosing this mathematical tool for analysis. Scalability, behavior prediction and finding acceptable Nash Equilibrium are within the reasons. The authors also list possible benefits from using the approach, such as a possibility to analyze a distributed system and to get a cross layer optimized solution as a result. Layers of the Open Systems Interconnection (OSI) model are considered separately along with related technical issues. Even though this area has its difficulties in modeling (e.g., traffic model complexity, dynamic topology etc.) the approach still looks promising when applied to different layers and may bring together results for cross-optimized solution. Yu and Liu also investigate game theory in the scope of mobile ad hoc networks [81]. The authors consider nodes that can be normal, selfish or malicious, and includes an investigation of possible way for a node to cheat, i.e., not sharing some information and keeping it only for itself. As criteria for decision making, not only Nash Equilibrium was considered, but other options like Pareto optimality. The authors investigate to which degree a node needs to help an opponent, e.g., in a packet forwarding game.

Agah et al. [82] apply game theory to analyze interactions between an adversary and a sensor network. The authors proposed a framework for an intrusion detection system (IDS) based on the derived Nash Equilibrium in the game. The general applicability of the approach for IDS is investigated by Alpcan and Basar [83]. In this thesis game theory is applied to analyze a monitor that can be considered as an IDS example. Buchegger and Alpcan address security issues in vehicular communication by formulating the corresponding game [84], the authors target developing an optimized strategy against an adversary in different scenarios. A comprehensive analysis of network security problems and related formulated games are presented by Manshaei et al. [85]. The authors consider six categories of security techniques and areas, one of them being IDSs and examine a possibility to optimize IDSs in the sense of configuration and attack response.

2.6 Monitoring

Applying network monitoring techniques to security issues leads to the development of an IDS. There are two main types of IDS depending on the logic of detection [86]. The first one is a signature-based network IDS. In this approach, there is a set of known attacks together with their corresponding patterns. The IDS is monitoring the system and raises an alarm, when there is a system pattern matching the one from the set. This approach has an obvious limitation, if there is an attack that was not considered at the development phase, it will not be detected. The second group is called heuristic or anomaly-based IDS. With this approach, the system instead knows some standard ways of behavior of the network and search for any anomaly, anything that does not match the standard pattern. The advantage of this method is the possibility to detect a previously unknown attack. On the other hand, if the intruder knows the specific network patterns, the adversary actions can be masked and indistinguishable from the normal network behavior. The patterns of communication in conjunction with clock synchronization algorithms, which are in the main scope of this thesis, are well known. Therefore, in this thesis a hybrid technique is targeted, i.e., combining a heuristic and a signature-based method. In this case, the monitor can be more flexible and have a higher probability of attack detection.

Monitoring is a common techniques to support real-time system properties as it allows to react upon detected changes in the system environment or its dependencies in order to provide required system functionalities. Already in 1991 Chodrow et al. [87] have considered monitoring of run-time constrains

for real-time systems. The authors have presented a monitoring toolkit for two types of monitoring when monitoring constraints are embedded, i.e., they are examined within the execution, and when monitoring is performed by a separate task. The authors have also discussed the possibility to use results of monitoring for system adaptation.

In Chapter 1, system safety was pointed out as the main motivation to consider security aspects for CPSs. Run-time monitoring as a technique for maintaining an acceptable safety level has been considered previously as well [88]. However, compared to other approaches, the methodology for monitor design proposed in Chapter 7 is aligned with the SEooC concept and focus on methodology of channel state manager design that can provide input for the system state manager to react upon the current situation. There are a number of ways to react upon changes in the environment, e.g., to relax system requirements [89]. Also the outcome of monitored data analysis can be used at run-time as well as during design time [90].

General guidelines of a particular monitor architecture choice are presented by Kane [91]. The author proposed the following list of fundamental points to consider: where it executes, how input is obtained, system properties that are monitored and the effect of the monitoring. Looking at monitoring in the security domain, Arora et al. [92] propose run-time monitoring of correct code execution, given security threats. The monitor considered in this thesis operates on a higher system level and targets to include both safety and security aspects. An example of a monitoring concept for resilient computing at the system level is presented by Stoicescu et al. [93]. The authors proposed to support dynamic system adaptation by a combination of a monitor and an adaptation engine.

Gutiérrez et al. [94] consider network monitoring in the context of synchronized networks. The concept of a configuration agent is introduced: an autonomous entity that learns the characteristics of the network through continuous monitoring, with the goal of facilitating the configuration and re-configuration of time-triggered networks. The configuration agent is composed of four elements: a monitor, an extractor, a scheduler and a reconfigurator. The duple formed by the monitor and the extractor gathers data from the network and distills relevant information from it. This information can be sent to the scheduler so it can produce a new schedule for the network, with which the network is re-configured [95]. The monitoring approach investigated in this thesis fits in the configuration agent concept, as detection of broken clock synchronization can be considered as a cause for a network reconfiguration allowing isolation of the unsynchronized node until the synchronization precision is brought back into acceptable boundaries.

An example of a network security monitor running on Ethernet and applied for Local Area Networks (LANs) is considered by Heberlein et al. [19]. The authors propose an hierarchical model of data analysis that allows separation of network activities into host-to-host, services, and connections groups. Eslahi et al. [96] propose to use traffic patterns for detection of periodic communication of Botnets, subnetworks consisting of infected devices. These two examples are from different areas and have twenty five years in between, demonstrating that security and monitoring can complement each other in an efficient way.

When evaluating a new solution or mitigation technique, it is essential to have some benchmarks and evaluation criteria. Fink et al. [97] propose a metric-based approach for IDS evaluation. Logical, architectural and performance metrics were presented as the main groups of criteria. Logical metrics imply such characteristics as cost, maintainability and manageability. Adjustable sensitivity, data pool scalability, data storage, and similar, can be considered as architectural metrics. Finally, error reporting and recovery, induced traffic latency, operational performance impact, and observed false positive and negative ratios, represent performance metrics.

Chapter 3

How Can Clock Synchronization Be Broken?

In this chapter, an approach for threat modelling is first introduced. Next, vulnerabilities of clock synchronization in the 1588 standard are considered and a corresponding classification and an analysis of attacks towards clock synchronization are presented. Finally, a particular way to break clock synchronization using an attack consisting of ARP poisoning followed by a selective delay attack is detailed with an investigation of consequences and mitigation techniques.

3.1 Threat Model

To address security issues, we need to define a terminology. In particular, consider the following components, organized as proposed in Fig. 3.1. *System assets* are the features that we want to protect in the system. They are the main assets of the use case, as they affect its workflow most and have the highest value. For example, in a system with sensitive data, the data confidentiality is an asset. *Adversary goals* represent the possible targets of a potential malicious intruder. If we consider intrusion detection systems in plants, a possible adversary goal is system hijacking. An *adversary model* represents a set of possible adversary features, such as assumed geographic location, time and budget. The adversary model is extremely important for correct risk estimation and appropriate security design. *System vulnerabilities* are flaws and weak spots in the

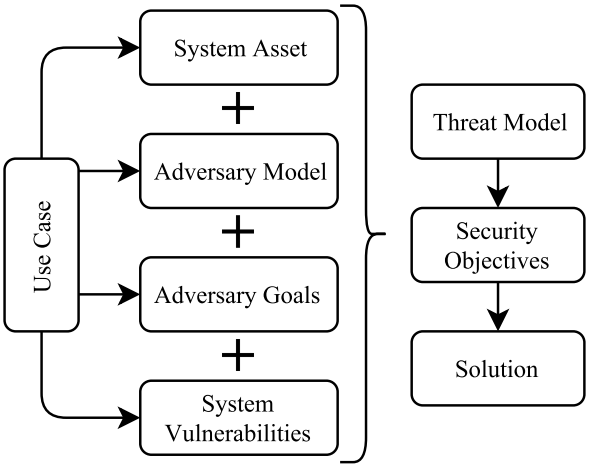

Figure 3.1: An approach for threat modelling

system design that could be used to conduct an attack.

The system assets, system vulenrabilities, the adversary model and goals can all be derived based on the considered use case, as shown in Fig. 3.1, since they reflect the specific properties of the use case. Together, these four features (system assets, adversary model, adversary goal, system vulenravilities) form a *threat model*. The threat model is an aggregation of security aspects to address in the system. The threat model is needed to formulate a set of *security objectives*, describing the security features and services we need to have in the *solution*. Below components of the threat model are considered together with possible security objectives using an example of automation control applications.

The threat model should reflect real adversary possibilities which are usually connected to the specific application, since this directly specifies the adversary goals, i.e., what an intruder targets, together with the assets that needs to be protected. Therefore the threat model should also include an adversary model. By adversary model we consider the specific features of an intruder, which can help understanding the possibilities of the intruder and structure ways to characterize the influence of an intrusion.

Application Specification. Industrial applications, due to their requirements on dependable transmission of real-time data, which often require clock syn-

chronization, are considered here as a use case. Today, sensor networks are widely used in industry for controlling, measuring and aggregation of data. For instance, a typical application example from process automation is a paper machine. To produce paper of sufficient quality the humidity of the process should be measured continuously, as deviations from the specified values can lead to quality degradation. As a result, real-time requirements are imposed by the system, and the humidity characteristic should be transmitted timely, reliable and continuously. Due to the speed of the paper production process, it is safer to measure humidity with wireless sensors rather than using a wired solution as wireless sensors can be fitted directly into a fast rotating part of the machine, rather than manually measuring using wired sensors.

Adversary Goals. The next step is setting the adversary goals and based on these, try to analyze what consequences it can have if the goals are reached and what countermeasures that can be imposed. According to [98], the three main adversary goals for sensor networks are disruption, eavesdropping and hijacking. As follows from the application example mentioned above, eavesdropping is not terminal for the considered sensor network. If an intruder can get information from the sensors (e.g. a water percentage in a specific type of paper) it can be objectionable as it potentially is a production secret but not fatal. On the other hand, if the intruder can change the data from the sensors or damage the system it may have terminal consequences. Therefore, eavesdropping can be eliminated from the list of adversary goals mentioned above. Consequently, the most significant adversary goals are disruption and hijacking as their impact on the targeted application is considerable.

Assets. In the considered use case, the paper itself is an asset. Further, this asset can be broken into a set of things that need to be protected in order to support paper as the asset. For example, if an adversary manages to deprive the system from water that is used to provide a required level of humidity, paper production will be disrupted. Thus, water supply is also an asset, however it is very application dependent. Talking about assets applicable generally for applications with real-time constraints, one of the main assets is clock synchronization. Mizrahi provides a good classification of possible treats for clock synchronization in [99]: interception and modification; spoofing; replay; rogue master; interception and removal; delay manipulation; denial of service (DoS) attacks for OSI layers 2 and 3; cryptographic performance; time source spoofing. This classification is very general and can be used regardless of if we consider a wired, wireless or mixed system. The specific set of tools needed to protect the asset clock synchronization depends on the system structure.

Adversary Model. In general, all adversary aspects mentioned by Clark et

al. [100] such as being passive or active, insuder or outcider and others, are of importance for wireless safety-critical applications. In particular, the most important adversary features in this context are: whether or not it is passive or active, static or adaptive, an insider or an outsider, the adversary mobility, communication capabilities and computational power.

Vulnerabilities. To complete the threat model, system vulnerabilities need to be identified and connected with system assets. The system is defined by the use case. The vulnerabilities can be related to design or system implementation. For the considered system asset, i.e., clock synchronization, vulnerabilities on the design level could come from assumptions in the clock synchronization protocol if they do not hold in a particular scenario.

A passive adversary can be a prerequisite for an active one, since at first, the adversary passively collects data and then, based on its analysis, starts to actively influence. If the adversary is an outsider, the goal is more connected to system disruption, as it is easier to suppress the channel or cause interference rather than to hijack the whole system. If the adversary is an adaptive one and can change its behavior depending on network response, it is more dangerous for the system functionality. Communication capabilities reflect whether the adversary acts through the network protocols or through the wireless channel or both. Possible adversary computational power depends on the specific application and the value of its assets. The higher the potential gain of interfering or hijacking a system, the more likely it is that the adversary invests in more computational power.

3.1.1 Security Objectives

After defining the threat model we can propose an approach that includes a list of security objectives. By security objectives, we consider system features that should be imposed and according to which we can choose a protocol that can cover all, or almost all, requirements needed. Therefore we need to analyze the assets, i.e., the things we want to protect. We should identify in which way an adversary can gain possession of or control over the assets such that we can identify the objective of the security mechanisms that need to be introduced. The set of security objectives that should be introduced depends on the application (e.g. military or personal data). Initially, we consider all objectives mentioned in [100] and [6] and thereafter, we remove the ones that are not directly applicable for process automation applications.

Confidentiality. Generally this security objective refers to that the adversary must not know information from the sensors. However, according the

identified adversary goals, confidentiality is not strictly required for the considered application, as the level of paper humidity data is no secret, but rather a well-known fact used as a feedback to control the process.

Integrity. This notion is connected with the following questions: has the data been corrupted; can we trust this source? Integrity is a prime issue for the asset clock synchronization in applications with real-time constrains.

Authentication. By this we consider that we must know from whom (which network node) we get this information. Consequently, authentication is also a key point for the clock synchronization asset.

Availability. Mainly this objective refers to the fact that the service provided must be available, i.e., in our case, the paper making process must function all hours of the day since paper machines are too expensive to stand still. When considering our safety-critical application, availability is possibly the most important objective.

Anonymity. This notion usually is understood as the possibility to use a network without being identified or having private data shared without consent. As we consider sensor networks in process automation, this is not the most important security objective.

Auditability. System behavior reconstruction which is assured by auditability can help to enhance reliability if it is made adaptive, but if we are talking about safety-critical applications, it is not a prime issue as such systems should be as reliable as possible already from the beginning. Note that a sensor network used for controlling a paper machine can be considered safety-critical from a financial loss point of view.

Nonrepudiability. This objectivity is about liability and has more legal than safety consequences, and therefore it is not a prime issue for the targeted application.

Third-party protection. This is about preventing damage done to third parties and it also more connected with reputation and legal consequences and therefore out of scope here.

Conformance. The network should work in accordance to the protocol. Just as with availability, this is one of the most important objectives for safety-critical applications.

After analyzing the security objectives mentioned above we can conclude that the ones that are most important for our considered applications are conformance, availability, integrity and authentication. Based on the list of identified objectives, we determine suitable techniques and approaches that can be used as a basis for the security framework. It is reasonable to first analyze existing protocols and techniques, to establish whether or not they can be of use for the

identified set of targets above.

3.2 Attacks towards Clock Synchronization

In this section, attack towards clock synchronization on a general level are considered and systematized by considering an object of an attack.

Due to the nature of clock synchronization correction protocols, once the attack is stopped, the clock comes back to synchronized state after the next correction. Therefore, the attack should be persistent for a period that is needed to achieve adversary goals. We assume that the adversary goal is to break clock synchronization and keep it broken for a time that is needed to cause significant consequences. This time is tightly connected to a targeted application and a situation when the attack is conducted. For example, if the adversary targets a steering system in a vehicle while it is being driven, information exchange between a switch and a wheel sensor is quite extensive, and therefore, communication will be disrupted some time-slots after the unsynchronized state. Therefore, the considered adversary target is refined for the rest of the chapter as follows: *to keep a node in an unsynchronized state for at least* t_{tar}.

To conduct the threat analyses, possible attacks should be considered and matched with existing vulnerabilities of a system. Thus, before conducting the ATA analysis, the possible ways of breaking clock synchronization should be systematically considered, e.g., an attack classification presented, and feasibility of presented attacks should be assessed, i.e., related IEEE 1588 vulnerabilities should be identified.

3.2.1 Attack Classification

The attack classification for a single- and multi-hop network with a time-triggered architecture is presented in Fig. 3.2. The first considered criteria for the classification is an object of attack. Two options are investigated, it can be a communication link or a node. A node compromisation can be achieved by compromising all communications links that have a connection to the node, however such case is considered as related to the first group, as the second group implies actual compromization of the node from ”inside”.

The first case implies that an adversary performs operations with a message, such as manipulation, deletion or insertion. Message manipulation can be performed by its replaying, i.e., sending it again, delaying, i.e., increasing or decreasing its transmission time, or forging, i.e., changing one or several

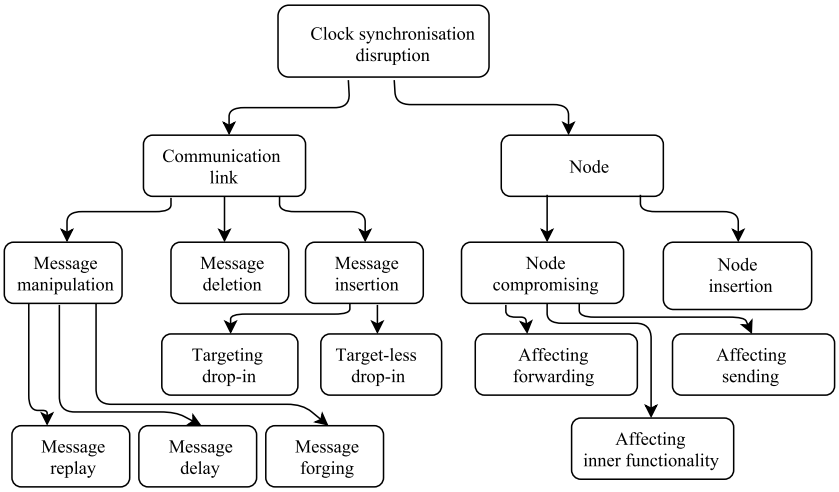

Figure 3.2: Classification of attacks targeting clock synchronization

data fields in the message. Message deletion can be conducted by messages interception and termination of its further propagation. Finally, message insertion implies forging not parts of the existing messages, the case we considered above, but forging the complete message and dropping it in the network. In this case two ways of ”drop-in” are possible, targeting approach means that a particular message is forged and inserted, targetless drop-in means that the adversary breaks clock synchronization through network availability, e. g., by flooding the network.

When a node is an object of an attack the following two cases are considered. First, the adversary can conduct node insertion, i.e., authenticate a new device to the network. Secondly, the adversary can compromise a legitimate network node. The latter can further be classified based on the behavior of the compromised node. Once a node is compromised, adversary capabilities are plenty, however we specify three cases depending on which functionality of the node is affected. The adversary can affect its forwarding functionality, e.g., change a route of a message or do not forward it at all. The adversary can affect its sending functionality, e.g., do not react to an event by sending out a message. Finally, it can affect inner functionality of the node, e.g., perform inaccurate message time-stamping. An illegitimate node, i.e., the case of node

insertion, is not decomposed further, as in this case the node does not have an apriory assumed ”correct” way of behavior, therefore, we do not consider possible ways of its behavior deviation. Also note that the considered attacks can be combined into a more complex attack strategy.

3.2.2 IEEE 1588 Vulnerabilities

According to the clock synchronization algorithm, in each RI an offset is calculated and the local clock is corrected according to the calculated offset value. This process has two points where adversary can act and affect the outcome. First is the offset that is calculated, if the value is incorrect, the outcome is a failure. Second, the process of calculation, if it is impossible to perform the calculation due to lacking information, the outcome is a failure too, as without correction the clock will eventually come to unsynchronized state. Below we consider these two cases and discuss which vulnerabilities in IEEE 1588 can lead to an attack targeting calculated offset or possibility to calculate it. This separation is also a basis for the ATA presented in the next subsection.

According to IEEE 1588, the offset is calculated based on four timestamps obtained during synchronization messages exchange between a GM and a local node. Therefore, the offset can be calculated incorrectly if timestamps are incorrect, i.e., if they are affected by an adversary. IEEE 1588 does not require any security solution implemented, in this case open communication is an obvious vulnerability allowing an adversary to easily penetrate the network affecting transmitted timestamps. However, optional suggestions provided in the Annex K includes security associations (SA) concept. Each incoming message is mapped to a SA that has a replay counter, lifetime identification and key identification. Also an integrity check via an integrity check value (ICV) is proposed. However, ICV is calculated not over the whole message and network protocol addresses are not included in the check. Therefore, an adversary can change them without a node realizing it, hence this vulnerability leads to a possibility to create a twin SA and to cause a reset to an existing one along with resetting a replay counter [71]. The provided ICV excludes forging of parts of the messages. i.e. a time-stamp field. However, synchronization messages still can be replay by tampering with SA and they can be delayed. The last case is the most challenging one, as it cannot be eliminated even by an integrity check covering the whole message. The possibility to create and reset SA also implies that an adversary can insert messages in the network that look valid. The other possibility to affect timestamp is to compromise a node that produces timestamps. We do not consider hardware compomisation in this work, however, a

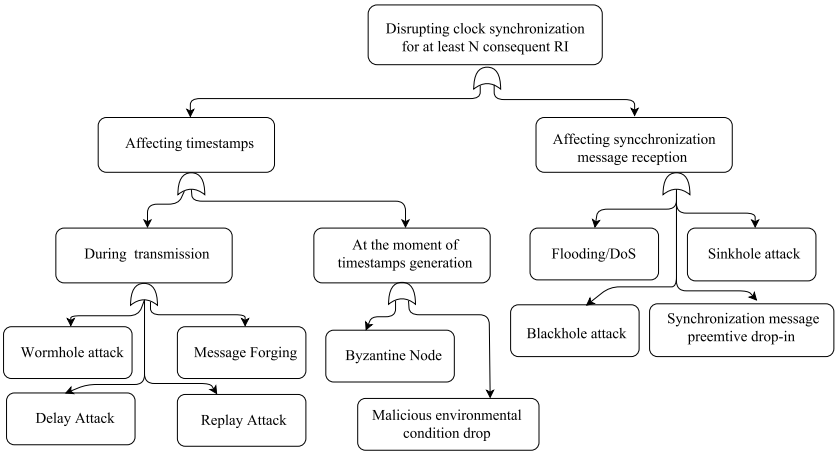

Figure 3.3: Attack tree for clock synchronization

situation when a malicious node can be authenticated into the network and can be chosen to be a GM or a communicational channel to a GM are taken over by an adversary, are all equal to a compromised node situation. IEEE 1588 even with security recommendations from Annex K still leaves these situations possible.

The second scenario that leads to clock synchronization being broken is when a node is isolated from synchronization information. This can be achieved in two ways. The first option is connected to network availability, e.g., a targetless messages drop-in or a DoS attack. As SA can be created and reset, a flooding attack is a possible option. The other way to keep a node without synchronization data is by targeting isolation. Information can be unavailable if there is a node blocking it, e.g., a blackhole attack. 1588 allows this as it is possible to take over a communicational channel, e.g., by an ARP poisoning attack.

3.2.3 Attack Tree Analysis

The assumption made in the ATA is that the object of the attack is a communication link, i.e., the case of a compromised or inserted node is not considered.

In ATA, the root of the three represents the adversary goal, in the considered case it is breaking of clock synchronization for at least t_{tar}. Next, leaves represent ways to achieve the target, i.e., threats. Each leaf can be a root for

the next refined level of threats, i.e., the lower leaf in the tree, the more specific and particular threat it relates to. Also leafs can be connected to a root through OR and AND logical gates. For clock synchronization there are two points for a possible failure: the correction can be done in a wrong way, i.e., the offset is calculated incorrectly, or the correction is not performed, i.e., the node failed to receive synchronization information. Therefore, as it is shown in Fig. 3.3, possible ways of affecting clock synchronization can be divided into two main groups, those related to affecting timestamps and affecting the reception of synchronization messages.

Further, the first group can be decomposed into attacks during message transmission time and at the moment of message generation. The group with timestamp transmission can be decomposed to the following options. It can be a wormhole attack, when a route of the message is changed, therefore its transmission time differs from the expected. A message can be forged, if there is not integrity check. The message can be replayed, which means a targeting node will be unaware of possible changes in the network, e.g., a change in the environment conditions causing additional difference in the clock drifts. Finally, the message can be delayed. This case can be decomposed further, as it has been shown that a delay attack can be performed if an adversary has some knowledge about the network structure and performs a man-in-the-middle (MIM) attack, e.g., via an ARP poisoning attack.

Attacks at the moment of a timestamp generation are represented by two possible options. The first considered situation is when there is a Byzantine node in the network, i.e., a node that behaves differently to different nodes. A prerequisite for this attack is this node compromization. As a result the node can send different timestamps to different nodes. The second identified option is a situation when the node is not compromised, however an adversary manages to affect the timestamp process. For instance, if the adversary has a physical access to the node and can suddenly change environmental conditions, e.g. temperature, for a short time. However, the possible effect depends on the clock quality.

Attacks affecting the reception of synchronization information might target network availability or a particular node. The first group includes flooding attacks causing DoS. A sinkhole attack implies changing routes of the messages in the way that they come to a ”sink”, i.e., an adversary, and are not propagated further. A blackhole attack, when message propagation is blocked by an adversary, can affect the whole network or a particular node, in case it is performed selectively. One more way of attacking a particular node is a preemptive drop-in of a message. If a node receives a message that looks valid

and starts its processing, it can discard other messages during a certain time interval. Therefore, an adversary can drop-in fake messages just before the valid one should arrive and keep the node unaware of information contained in genuine messages. The presented attacks can be decomposed further, however their prerequisites, e.g., knowledge of the network structure or physical access to it, are not considered.

3.3 Use Case: ARP-poisoning and Subsequent Selective Delay Attack

After considering on a high level atacks towards clock synchronization, the focus is narrowed down to a delay attack. However, the delay attack needs a communication chanel being under adversary control, thus ARP-poisoning is considered as a technique possible enabling subsequent selective delay attack.

Considering possible ways of attacking the system and analysing system reaction to the intrusion, it is important to set the limits and assumptions of the investigated scenarios. There are many possible ways of interactions between the adversary and the system depending on the assumptions for both. Thus, a *use case* or a system model in this section is the following: starting with wired networks, synchronization is established and maintained according to IEEE 1588. Using the peer-to-peer mode in 1588 implies that network participants periodically exchange messages to be aware of delays in the channels between them and their neighbours. The networks consist of routers capable of message time-stamping and nodes, particularly grandmasters and slaves. Routers and nodes have the transparent type of clocks, meaning that they perform hardware time-stamping of synchronization messages upon arrival and transmission, via update of the correction field in the follow-up messages.

Fig. 3.4 shows the sample topology. The network consists of a grandmaster, *GM*, a slave, *S*, and a set of routers, R_i. There are two communication channels between the grandmaster and the slave, namely downlink and uplink. The only considered *asset* of the specified network is clock synchronization.

To perform an attack analysis, the *adversary model* should be specified first. We assume that the adversary has access to and initial knowledge about the network. The adversary targets clock synchronization breaking, therefore, a link which is involved in the synchronization protocol is attacked. Apart from the assumptions mentioned above, adversary choices can be random or based on an analysis of the network conducted in advance. Consider a case when the adversary attacks only communication channels, links. In this case, the adver-

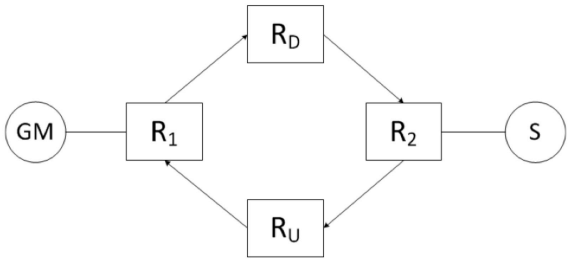

Figure 3.4: The network topology used for the simulations

sary is capable of receiving, transmitting and delaying messages. At this point, the capability of learning (i.e., the possibility to analyse the reaction of the opponent and connect consequences with causes) and behavioural adaptation is not considered. The main *adversary goal* is system disruption, i.e., the adversary intends to cause system error and propagate it as much as possible, so that it leads to a system failure. Also, the adversary targets to maintain influence and stay undetected. We assume that the behaviour is rational in pursuing the above-mentioned goals.

We investigate the case when the adversary uses an ARP poisoning attack to penetrate the network and take control over communications in the targeted channel, i.e., the adversary conducts a man-in-the-middle attack using a vulnerability in ARP. Once successfull, the next phase for the adversary is to perform a selective delay attack targeting synchronization. As it was specified in the adversary description, the objective of the attack is a link. The considered scenario is a case when one link is under attack, i.e, when the adversary controls the communication between router R_2 and slave S.

The correction of clocks according to IEEE 1588 is done via exchange of synchronization messages. In this work the exact character of the clock drift does not change the analysis, thus even a linear dependency can be considered. During this exchange, the slave clock calculates its drift and as a result performs a correction. This approach is based on two assumptions:

- A message needs the same time for being transferred from node A to node B as for being transferring in the opposite direction, from node B to node A, i.e., that the delays are the same in both directions.

- The message exchange can be done in short enough time so that the

information acquired about the clock drift is still valid and can be used for correction, i.e., that the calculated offset can be considered correct.

The first assumption is used as a vulnertability in the rest of this chapter, as a selective delay attack breaks the first assumption and exploits this vulnerability.

3.3.1 ARP-poisoning

ARP is a well-known and widely used protocol, and therefore, ARP poisoning is also a commonly used attack method [101]. The ARP protocol is part of the TCP/IP stack, and hence, it is used in many networks. To resolve the correspondence between MAC and IP addresses, ARP uses two types of messages: ARP request, a broadcast message, and ARP reply, a unicast message. The algorithm is simple: when a node A wants to send a message to a node B, and A has the IP address of B, it needs to ask about the corresponding MAC address for it. First A checks its ARP cash table, and if the address is not there, it sends a broadcast ARP request message to all network participants asking about the MAC address of the IP address it has. The node with the mentioned IP address answers with an ARP reply message, after which communication can start. These messages do not have any authentication properties, so it is easy to intercept and forge them. An adversary can send fake ARP replies to A saying that he is B and to B saying that he is A. After this A and B will communicate with each other through the adversary.

The ARP poisoning attack can be performed independently of the physical layer implementation, and therefore it is suitable for both wired and wireless networks. For instance, a possible scenario of applying ARP poisoning against an industrial network building on PROFINET IO was considered in [101]. In the paper, the authors demonstrate that by using an ARP poisoning attack, an adversary could gain control over the outputs of a PROFINET IO device, which can lead to a number of possible attack continuations with a huge impact on the network. In [102] the authors consider performing a combination of a hole 196 attack (an attack that uses a vulnerability of WPA2 and exposes the network to insider attacks) along with an ARP poisoning attack in networks based on the IEEE 801.11i standard. Such an approach allows the adversary to decrypt all traffic going through the access point from an attacked user. In both works mentioned above, the authors consider only the possibility of performing an ARP poisoning attack in different networks, whereas this thesis considers the specific use of an ARP poisoning attack to break clock synchronization.

As it is possible to perform a man-in-the-middle attack in the network via ARP poisoning, it opens possibilities for the adversary to influence the clock synchronization algorithm by imposing a delay in the delay-request-response mechanism used by IEEE 1588. Such kind of an attack can lead to network synchronization failure. Depending on the link and node chosen for the attack, the consequences can range from one node failure to complete system disruption. The most obvious way to influence IEEE 1588 is to try to break one of its basic assumptions listed in the previous subsection. The second assumption regarding the message exchange being fast enough is difficult to violate for an intruder as it mostly depends on system specific configurations and to be effective, the propagation delays should be significant. In contrast, the first assumption that requires delays in both direction being the same, is appealing from an intruder point of view, as its violation implies an additional delay in only one direction and it does not need to be large. Moreover, the message does not need to be changed — it needs to be delayed. An ARP poisoning attack performed in a network using IEEE 1588 implies that the adversary has full control over the communication between two chosen nodes, even if the optional security extensions from IEEE 1588 Annex K are applied. This means that the adversary can easily impose the necessary delay in one of the directions of the delay-request-response mechanism. The idea is that the adversary needs to influence the system in such a way that the output of the synchronization algorithm (the correction shift) is bigger than the allowed threshold.

The described attack is possible even for networks using the optional Annex K of IEEE 1588. The security extensions provided in the annex include message integrity, group authentication and protection against replay attack. However, message integrity cannot help against this attack, as the message does not need to be changed. Group authentication cannot help as the adversary is pretending to be a device already existing in the network. ARP poisoning implies a forge in the initial unprotected part of the communication between nodes, and later the messages between them would be passed normally without any change, apart from an additional delay caused in one direction. Replay attack protection also cannot help as the message is not supposed to be replayed.

There are some limitations for this type of attack and cases when the attack is complicated or useless. To perform this kind of attack successfully, an adversary needs to know which link, node/switch or set of nodes/switches to attack. This means that the adversary first needs to perform a network analysis to find the desired or the weakest point for attacking as attacking a random node most probably will not lead to costly consequences. Also, ARP attacks work only for subnetworks, and therefore, if the whole network consists of several sub-

networks, the adversary cannot affect the entire network, as it cannot influence one subnetwork while performing the attack in another one. Another limitation is the network configuration. If we consider a network with a completely static configuration, then all network participants can get complete ARP tables with all addresses during the first initialization phase. This kind of attack is therefore possible only in bigger networks or networks where devices are allowed to join during its operational phase. On the other hand, the attack also has a number of advantages from an adversary point of view, where the two most appealing ones are that industrial networks of today do not consider this possibility and that the same technique can be used in several different types of industrial networks.

As we consider industrial networks with different levels of criticality and complexity, some factors of the attack can depend on the specific use case and adversary goals. The scenario described above will put the two communicating nodes in an unsynchronized state. This is a straight forward case, but if the industrial system is developed to consider possible faulty nodes for increased robustness, which is often the case in systems with high cost of failure, the adversary has to target the disruption of a set of nodes. The size of this set depends on the application and the network architecture. Thus it is rational to target a node that is critical to the system functionality.

Within the considered network, an adversary can target a grandmaster clock or a slave clock. If the grandmaster clock is put into an unsynchronized state, the system will choose a new grandmaster clock according to the Best Master Clock (BMC) algorithm. In this scenario, the adversary can influence the overall network performance, e.g., if the networks has only a limited amount of clocks with external GPS receivers, the adversary can aim to remove these from the network first, by putting them into unsynchronized mode. This way, the network will degrade considering clock synchronization precision. It is worth to mention that in order to keep a clock in unsynchronized mode, the adversary needs to keep influencing the propagation delay, or else only a transient clock synchronization error occurs [103].

If the adversary targets a slave clock, putting it in unsynchronized mode will not influence the other clocks. This can therefore be beneficial for the adversary only in case the corresponding node has a critical functionality and the system does not have any redundancy. Note that even after detection of an unsynchronized node, the reason for becoming unsynchronized will not be discovered unless the system has countermeasures against ARP poisoning. Therefore, even if maintenance functionalities will replace the node thinking that it is out of order, the adversary can simply continue to influence it in a similar

way. In case a node is critical, it is also interesting to investigate if and when its unsynchronized behavior would be detected by the system.

The adversary can be interested in transient system influences, i.e., keeping a clock unsynchronized for a short period of time, if he needs to masquerade or hide some other short time activity that otherwise can be detected. For example, if the adversary wants to sabotage an assembly line in a plant, he can target the pressure or distance sensor nodes. Their unavailability, even for a short period of time, can lead to an accident. This scenario is important for critical applications where availability is one of the main security objectives.

The cases described above demonstrate that the ARP poisoning attack is problematic for industrial networks that do not have any kind of protection against it. Further, it shows that an adversary can pursuit several different goals by conducting the same attack. This fact makes the considered combination of attack techniques even more appealing for an adversary.

3.3.2 Selective Delay Attack

Here the analysis of the second phase of the attack towards clock synchronizarion is presented, performed once control of communication channel is taken over. An analysis of the consequences of introducing delays on the time synchronization in the network is shown.

If we use the value of the offset, σ_{meas}, as obtained in (2.6) to correct the slave clock, i.e., to tranform t_S^{old} to t_S^{new}, we have:

$$t_S^{old} \rightarrow t_S^{new} + \sigma_{meas}. \tag{3.1}$$

Using (3.1) in (2.7) we obtain $t_S^{new} - t_{GM} = 0$, making it the best possible value for the offset. This is the value that we would obtain in a attack free situation, therefore henceforth it will be referred as σ_{af}

$$\sigma_{af} = \frac{1}{2}((t_2 - t_1) - (t_4 - t_3)). \tag{3.2}$$

The value of the offset obtained above, assumes that the transmission delay is symmetric. However, the adversary in the attack that we are considering (Fig. 3.5) introduces an asymmetric delay, d_{adv} that affects the synchronization messages in the following way:

$$\begin{aligned} d_{GM \rightarrow S} &= d_0 + d_{adv} \\ d_{S \rightarrow GM} &= d_0. \end{aligned} \tag{3.3}$$

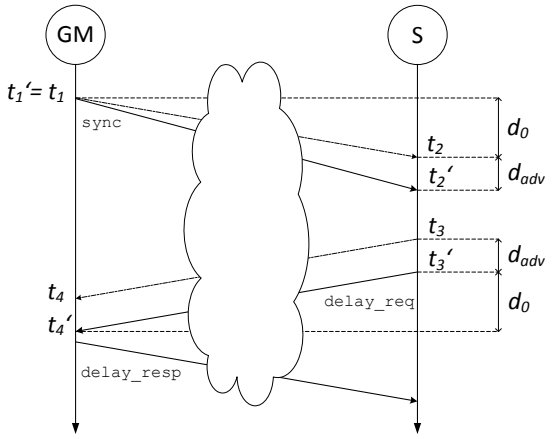

Figure 3.5: Clock synchronization protocol under attack

Now the time-stamps collected by the slave through the synchronization protocol described in Section 2.3.2 are:

$$\begin{aligned} t_1' &= t_1 \\ t_2' &= t_2 + d_{adv} \\ t_3' &= t_3 + d_{adv} \\ t_4' &= t_4 + d_{adv}. \end{aligned} \tag{3.4}$$

Substituting (3.4) to (2.6) and using (3.2), we obtain the value of the measured offset when there is a delay introduced by the adversary according to:

$$\sigma_{meas} = \sigma_{af} + \frac{1}{2} d_{adv}. \tag{3.5}$$

If the adversary delays the delay request message instead of the synchronization message, only the sign of d_{adv} in (3.5) will change. The choice of message to delay does not affect the following reasoning. If σ_{meas} is used to correct the slave clock, then $t_S^{old} \rightarrow t_S^{new} + \sigma_{meas}$ and because initially $t_S - t_{GM} = \sigma_{af}$, the difference between the slave clock and the grandmaster is now:

$$t_S - t_{GM} = \sigma_{af} - \sigma_{meas} = \frac{1}{2} d_{adv}. \tag{3.6}$$

Taking into account that this is the value just after the correction, from where the slave clock will start drifting again and thus for the next Rl it will be:

$$t_S - t_{GM} = \frac{1}{2} d_{adv} + \sigma_{meas}. \tag{3.7}$$

The worst case would be if $|\sigma_{meas}| = \sigma_{max}$, combining this result with the synchronization condition (2.3), we can conclude that an attack that introduces an asymmetric delay, no matter how small, can break the synchronization. However, this holds true only as long as the chosen σ_{max} for the network is actually the minimum value possible as expressed in (2.4).

According to the defined adversary model, the adversary strives to keep the network penetration unnoticeable. Furthermore, in some cases, the adversary can succeed in keeping the slave ignorant of the synchronization breaking whenever protection techniques are lacking. The best scenario for the adversary is to break the clock synchronization, while letting the slave think that it still is in a synchronized state. If as a result of the attack, the slave still thinks that (3.13) holds, but in reality the offset between the grand master and the slave is bigger than σ_{rel}, the adversary has succeeded. The advantage, from the adversary point of view, of such an outcome is that the system remains oblivious of its failure. This means that the system will not apply any countermeasures to mitigate the consequences and consequently will not be able to return to a safe state.

The synchronization messages are sent from the grandmaster with a period equal to the RI, R_{int} and the use of transparent clocks eliminates any possible interference of the rest of the traffic in the network. This means that any variation in R_{int} of the synchronization messages as perceived in the slave could be a hint of something happening in the network.

To detect these anomalies, a monitor should be placed in the slaves. There the monitor will be tracking the arrival times of synchronization messages to the slave. The interarrival time between two consecutive messages is:

$$\Delta t_i = t_{i+1} - t_i. \tag{3.8}$$

Although, the synchronization messages are sent periodically, some variations of the interarrival time should be expected. Nevertheless, an abrupt and sudden change in the interarrival time could be an indication of an attack happening.

Of course, other subtler, smarter attacks are likely not to be detected just by inspection of the interarrival times. For those, some previous knowledge of the network needs to be used. Here, the assumption is that a history of the arrival times of synchronization messages to the slave is available. With a set

of n values we calculate the average of the interarrival time, that would be the RI, R_{int}, as perceived by the slave:

$$R_{int} = \frac{\sum_{i=1}^{n-1}(t_{i+1} - t_i)}{n-1} = \frac{t_n - t_1}{n-1}. \tag{3.9}$$

The main difference between using the interarrival times or the RI as a parameter to detect an attack is that the first is an instantaneous measure that shows right away if something is happening but a smarter attack can go undetected. On the other hand, using the RI it is possible to spot more subtle attacks but some data need to be accumulated before a trend arises.

3.3.3 Potential Solutions and Mitigation Techniques

Once attacks are introduced, mitigations techniques for both are considered. However, the delay attack is in the focus and the ARP poisoning attack is considered as one of the ways to enable the former.

Mitigation techniques against ARP poisoning

In this subsection, an overview of existing solutions together with evaluation from an industrial point of view is given. There are plenty of different security solutions, covering different sets of security objectives, applied at different layers of the OSI stack, suitable for different environments etc. [14]. The possibility to break clock synchronization by performing an ARP poisoning attack and thereafter imposing delays on a hijacked communication channel is considered. Due to this, the security solutions can be divided into the two main categories: mitigation techniques against ARP poisoning and mitigation techniques against delay imposing. Solutions from both categories can be used for clock synchronization protection.

ARP poisoning is a well-known type of attack, and therefore, possible countermeasures were investigated in many research papers. The most relevant ones for this particular use case are considered here together with an investigation if and how they can be applied in industrial applications. A comparative analysis of possible mitigation techniques for ARP poisoning was presented in [104], and in [105, 106] several protection techniques were presented. Most existing techniques like [105, 104] and [106] imply implementation of an additional security mechanism. It can be e.g., an added encryption scheme, i.e., the node can encrypt all ARP request and reply messages. This solution helps against an ARP poisoning attack, but it also means additional computational

power in the nodes, and additional delay for message transmission. The additional overhead is disadvantageous for industrial applications with low latency requirements, as it requires complementing all network participants with an encryption/decryption module which lacks backwards compatibility.

The second approach is the introduction of a control element in the network that monitors and analyses the data traffic in order to detect a possible ARP poisoning attack. This can be realized via a centralized detection and validation server [105, 107] or a passive analyzing detection system. The server can confirm ARP requests and validate the ARP tables of all the nodes within the network. The applicability of this method in industrial environments depends on the network size, since the process of controlling all ARP tables in a large network can become problematic, and also implies the introduction of a single point of failure, since if the server is compromised or fails, the protection stops working. This is a questionable solution for critical applications and for applications with a distributed control architecture. The passive detection system looks promising, but it also has its limitations. Such a detection system can record all ARP requests and replies and construct the network according to the information gathered. It can constantly monitor the resulting network map for any inconsistency that will indicate an attack. This approach can work only if the attack starts after the data analysis has been initiated, and consequently this method is also limited by the size of the network. However, the approach can be a good candidate for mixed protection systems, where several methods are combined in order to achieve an appropriate overall security level.

Another approach implies using an IDS with probe messages [106]. Classical IDS detects an intruder by monitoring the system states, and thus this approach works under the assumption that the intruder causes a difference in the state sequences that can be detected by the IDS. Therefore, a classical IDS cannot be used against ARP poisoning, as the attack does not cause any difference in the event sequence. Hence, to be used against ARP poisoning, IDS should be complemented with a probe message mechanism. The idea is that the control-monitoring center from a classical IDS can now send probe messages to the network participants and these messages cause a difference in the event state depending on the presence or absence of a malicious adversary performing ARP poisoning. Requests from the monitoring center to verify genuineness of ARP requests and replies can be used as such probe messages. This approach introduces additional communication overhead, implying that for delay-sensitive systems and large networks, using IDS can be expensive. In addition, the method leads to the introduction of a single failure point.

The main idea of protection against delay imposing is to check and con-

trol propagation delay [108]. The method implies monitoring performed by the node itself or a switch. In this case, the network participant needs to collect and analyze statistics about message propagation delays in the networks. However, the approach requires additional recourses that can be complicated if the participant is a wireless sensor node. Further, the approach works only if the adversary joins the network after the statistic collection and analysis has begun. This mitigation technique has a probabilistic character. Considering the known techniques against ARP poisoning and delay imposing, we can see that they do not fully suit the industrial environment. Therefore, a possible solution in this case can be a combination of several approaches. A combined approach can be considered as a defense-in-depth technique, because initiation of the second category assumes failing of the first one. This can bring flexibility and allow satisfying the requirements needed for a possible security solution. For example, for most small networks, we can use encryption, while in less critical parts of the network, an intrusion detection system along with a delay analysis can be applied.

Disscusion of mitigation techniques against delay attack

A few mitigation techniques that can be used alone or in conjunction with the traffic monitoring to strengthen the IEEE 1588 against delay attacks are indetified in this section. These mitigation techniques are not enough by themselves to prevent, protect against or detect an attack, but they can be used to put some boundaries to the damage caused by the attack, thus increasing the resilience of the system.

Mitigation techniques

(A) Bounding the breach

Recall that the IEEE 1588 clock synchronization protocol is based on the exchange of messages between the grandmaster and the slaves. Concretely, in Fig. 2.7 it can be seen how the message `sync` is sent at t_1 from the grandmaster to the slave and, as a response, the message `delay_req` is sent from the slave at t_3, arriving to the grandmaster at t_4. Similarly, the slave is waiting the response from the grandmaster, the `delay_resp` message that arrives at t_5. These two request-response relations can be used to prevent that delay attacks take the clock in the slave irreversibly away from the grandmaster clock. To do so, we define t_{ret}

for the grandmaster and the slaves as the timespan between sending the message and receiving the corresponding response message:

$$\begin{aligned} t_{ret}^{GM} &= t_4 - t_1, \\ t_{ret}^{S} &= t_5 - t_3. \end{aligned} \tag{3.10}$$

The minimum value for these is the transmission time of the message in the best case, i.e., when it does not suffer any intervention in terms of delay attacks, queuing delays or MAC layer contention. Thus, let n be the number of hops that the message goes through, then given the message length, m, and the data rate,r, the value of t_{ret} can be calculated as

$$\min(t_{ret}) = 2n \cdot \frac{m}{r}. \tag{3.11}$$

To calculate the maximum t_{ret}, contention and execution times in the nodes must be taken into account. For the contention we assume that the message can be delayed by at most one message of maximum length in each hop. For the execution time we assume the worst case execution time (t_{WCET}):

$$\max(t_{ret}) = \min(t_{ret}) + 2n \cdot \frac{m_{max}}{r} + t_{WCET} \tag{3.12}$$

In a small network as the one depicted in Figure 3.4 with just four hops between the grandmaster and the slave and keeping aside the execution time, the range of t_{ret} is [8, 104] μs (assuming synchronization protocol messages of 126 bytes, a $dataRate$ of 100 Mbps and for the contention using the maximum length for an Ethernet message, 1522 bytes). This means that any value that exceeds that range can imply that the network is under attack. Note however, that with this method only the delay attacks that introduce a delay longer than $\max(t_{ret})$ are detected each time. An attack that introduces a delay of, e.g., 50 μs will not be detected in a situation of low contention even though it is clear from Section 3.3.4 that this is a delay large enough to break synchronization. Hence, this method can not be used alone as a detection mechanism, but only as a mitigation technique to prevent the attack from causing excessive clock drifts.

(B) Relaxed Mode

One of the possible network reactions to the detection of an attack is to switch to a relaxed synchronization condition mode. This means that σ_{max} in (2.3) is increased. In order to make the network more resilient to possible attacks, we can relax this assumption and a longer value can be chosen. Thus, let σ_{rel} be the maximum allowed offset, the synchronization condition would be:

$$|t_{S_i} - t_{GM}| < \sigma_{rel}. \tag{3.13}$$

Then we obtain a relation between d_{adv}, σ_{max} and σ_{rel} that states that in order to break the time synchronization, an attacker has to introduce a delay twice as long as the difference between σ_{rel} and σ_{max} or, stated differently, a network can tolerate attacks that introduce a delay twice as long as the difference between σ_{rel} and σ_{max} before the time synchronization is broken and thus

$$|\frac{1}{2}d_{adv} + \sigma_{max}| < \sigma_{rel}. \tag{3.14}$$

This relaxed mode leads to degradation of the network service quality, but may enable faster network recovery. Obviously, the applicability of such an approach depends on the criticality level of the application and the estimated time needed for recovery. It should be mentioned that the ability of the system to switch to the relaxed synchronization condition mode should be considered already during the system development phase.

(C) Using environmental conditions

IEEE 1588 targets industrial applications that implies coping with related environmental conditions (e.g., temperature, humidity). These conditions can influence hardware and particularly the clock crystals. To investigate possible consequences for clock synchronization, the message exchange between a *GM* and a slave *S* through a set of routers R_1, ... , R_n is considered, Fig. 3.6. At each message exchange chain, an error δ, that is caused by hardware time-stamping inaccuracy, is also considered.

For simplicity first we consider a synchronization message exchange without intermediate nodes, routes, as it depicted in Fig. 3.5. In this

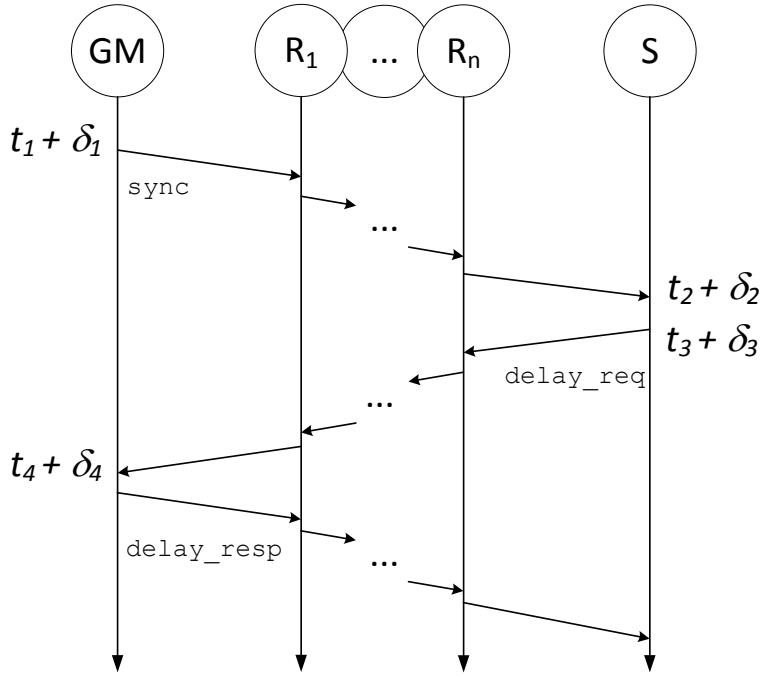

Figure 3.6: Synchronization protocol considering additional clock drifts introduced by environmental conditions.

case, in order to take into account additional deviations caused by environmental fluctuations, the following substitutions are required:

$$\begin{aligned} t_1\text{"} &= t_1 + \delta_1 \\ t_2\text{"} &= t_2 + \delta_2 \\ t_3\text{"} &= t_3 + \delta_3 \\ t_4\text{"} &= t_4 + \delta_4. \end{aligned} \tag{3.15}$$

Then by using (2.6) we can see that the resulting value of the new offset $\sigma_{meas-envir}$ can be obtained as:

$$\sigma_{meas-envir} = \sigma_{af} + (\delta_2 - \delta_1 + \delta_3 - \delta_4)/2. \tag{3.16}$$

The values constituting the error δ_i can be grouped according to which node they are produced by. Actually, errors made by the same node are similar if we consider events occurring close in time, implying that the events of sending the `sync` and the `delay_req` messages are likely to be subject to the same environmental delay, and conversely, the event of the receiving the sync and the delay requests messages are likely subject to the same delay, according to

$$\begin{aligned} \delta_{GM} &= (\delta_1 + \delta_4)/2, \\ \delta_S &= (\delta_2 + \delta_3)/2. \end{aligned} \tag{3.17}$$

This yields:

$$\sigma_{meas-envir} = \sigma_{af} + \delta_S - \delta_{GM}. \tag{3.18}$$

If we add intermediate nodes, routers to the chain of synchronization messages exchange, their corresponding errors will be included in equation (3.18) twice (once per link they are connected to) but with different signs. Each intermediate node is the end of the first related link and the beginning of the second related link, i.e., as we consider the difference of these errors for each link, they will have different signs in the final expression. Strictly speaking, when the two errors associated with the same node are subtracted, the result is not exactly zero, but it is negligibly small. Therefore, when considering the errors caused by environmental conditions only the grandmaster and the slave errors are significant even if intermediate routes are included in the path. This leads to the conclusion that the most significant influence from the environment occurs when the grand master and the slave are separated far enough, such that they can have different environmental conditions.

This knowledge can also be used for detecting abnormal delays in the communication that cannot be explained by the environmental conditions alone. If nodes have sensors collecting data about main factors influencing clocks crystals, it is viable to calculate possible clock offsets between nodes caused by different environmental conditions. If the observed shift is bigger than what can be expected by environmental conditions, this can indicate the presence of an adversary. Having a clock offset bigger than what was estimated can trigger additional checking of, e.g., the links for asymmetry delay detection. Under the assumption of having 5 μs offset with 50 ppm drift, environmental conditions can add 10 ppm to the drift and result in 6 μs offset [109]. This number shows

that if the clocks are without temperature compensation, they can affect clock synchronization quite significantly.

When an attack is detected, while the system is in the Relaxed Mode, it should first try to mitigate existing consequences, i.e., the synchronization being broken and, second, try to prevent the propagation of further consequences. To complete the first goal, related network participants should be informed that there are compromised links. Once the attack is detected, the monitor could simply indicate between which grandmaster and which slave that there is a security breach. The monitor typically knows the path on which this breach occurred, but it is not known where exactly the adversary is. In the worst case, the whole path from the considered grandmaster to the slave should be eliminated from the clock drift calculations. The adversary localization can be made by checking, one by one, all the links in the path under the assumption that there is a technique for checking the suspicious link without letting the adversary know about the check. It can be forged synchronization messages, where delaying would directly reveal the presence of the adversary. It is a challenge, as there are many parameters to consider and assumptions to validate.

3.3.4 Results

Evaluation of Impact with ARP Poisoning

In this section the results of an evaluation of the impact of an ARP poisoning attack targeting the clock synchronization functionality is presented. The evaluation process can be separated into two steps. The first step concerns the ARP poisoning itself, by formally specifying the ARP protocol and possible adversary actions. The second step is to evaluate the breaking of clock synchronization in the system assuming that the ARP attack was performed.

The tool used for the evaluation is AVISPA [23]. AVISPA is typically used for sensitive security protocols and application evaluation and analysis. It uses the High-Level Protocol Specification Language (HLPSL) for interactions with users. The Security Protocol Animator (SPAN) [110] tool was developed to simplify the interaction process with a user. SPAN allows a user to use the CAS+ language to describe the protocol and then it translates it to HLPSL. This makes the work with AVISPA easier, since all a user needs in order to analyze a protocol is to specify the modules: identifiers, messages, knowledge and goals. To formalize the proposed attack process, the following situation was considered:

Table 3.1: Identifier declaration of users and numbers

Type	Identifier
User	A, B, C
Number	IPa, IPb, MACa, MACb, MACc

Table 3.2: Identifier declaration of user knoweledge

User	Knowledge
A	C, B, IPa, MACa
B	A, C, IPb, MACb
C	A, B, MACc, IPb

Identifiers. We use the simplest case, when there are three participants in the network A, B, and C. A and B are benign network participants and C is an adversary. As it is shown in Tab. 3.1 they are specified as users, whereas IP and MAC addresses of all three are specified as numbers.

Messages. The specified message set implies that the adversary sends ARP responses to both A and B with the wrong IP addresses, i.e., C sends an ARP response with the IP address of B and MAC address of C to A and correspondingly the IP address of A and MAC address of C to B.

Knowledge. Each network participant knows all IP addresses in the network plus its own MAC address (Tab. 3.2).

Goal. The tool is limited in the definition of possible goals, so we apply the reverse technique to its formulation, i.e., in case the tool proves that the goal is achieved, it means that clock synchronization is broken. To prove that ARP poisoning can be performed, we specify the goal as: to keep the secrecy of the MAC address of B from A. If the tool shows that the protocol is secure, this means the adversary wins, as A cannot understand that he is communicating with C instead of B.

The assumption we use in the modeling is that the intruder (node C) knows the IP addresses of the targeted network participants. If the network allows a new device to join, then node C can be considered as a new device in the network, and otherwise it is assumed that node C was in the network already from the initialization phase.

AVISPA has several types of analysis techniques, namely the On-the-fly Model Checker (OFMC), the Constraint-Logic-based Attack Searcher (CL-AtSe), the SAT-based Model Checker (SATMC) and the Tree Automata based

on Automatic Approximations for the Analysis of Security Protocols (TA4SP). We used OFMC in the evaluation, as this technique can prove that the specified protocol is correct, and this is exactly what we need, given the way we defined the goal and that we included an intruder in the user list (node C). OFMC [111] explores the transition system by using a demand-driven approach. The checker uses symbolic techniques, a lazy Dolev-Yao intruder model and lazy data types. The last one means that data constructors do not evaluate data arguments while building it, which allows infinite data computing. This attacker model is suitable for the approach, as ther modeling assumes that an intruder is a legitimate network participant.

The analysis with OFMC shows that an ARP poisoning attack is possible under the given assumption, namely that the adversary knows the IP addresses of network participants, implying that some prior network analysis has been conducted.

The second step of the evaluation can be done using logical reasoning without any verification tool. The clock synchronization algorithm can be broken if an adversary succeeds in breaking one of its basic assumptions. In our case, the assumption is that the propagation delay is equivalent in both directions within the same logical channel. Obviously, if the adversary successfully performed a man-in-the-middle attack and controls the communication process in both directions, the necessary delay can be imposed in one direction. More precisely, the adversary needs to impose a delay greater than the maximum allowed clock drift. This can be done if the adversary knows the synchronization period and the maximum allowed clock drifts in the system. This knowledge, as well, can be gained through prior network and specific application analysis.

Simulations with OMNeT++

To evaluate the proposed approach, a network simulation using OMNeT++ [27] together with the INET framework [112] was created. For the concrete modules needed for the simulation of the clock synchronization protocol, the implementation made by Lévesque et al has been used [70]. The goal of these simulations is first to demonstrate that the delay attack can indeed break the clock synchronization [113]. The simulations will alsopartly verify that the data obtained by the monitor is the same data that a real monitor would gather in a real network.

Fig. 3.4 shows the topology of the simulated network. The communication starting in GM and going to S through the downstream router R_D. Communication starting in the slave goes to the grandmaster through the upstream

router R_U. The RI R_{int} is set to 100 ms. We assume that the drift rate of the slave clock is 50 ppm and, therefore, applying (2.4), $\sigma_{max} = 10$ μs. We chose $\sigma_{rel} = 20$ μs, thus implying that the system has been designed to work properly with this synchronization accuracy. Without loss of generality, just in order to simplify the explanations of the simulations, it is assumed that the master has a perfect clock, i.e., it does not drift. However, the slave is, of course, not aware of this fact.

We simulate the effects of an attack that breaks the time synchronization as shown in Section 3.3.2. For that different models for the delay are used: a constant delay and a linearly increasing delay. Once an adversary penetrated the network, it can use different techniques for messages delaying. The goal is to investigate different cases going from the simplest one to more complex and try to analyze the differences from a detection point of view.

A. Constant delay, $d_{adv} = 50$ μs

In Fig. 3.7 the variations of the difference between the slave clock and the grandmaster clock with time is shown. Before the attack, the difference was oscillating between 0 and -5 μs, as the result of the clock synchronization protocol performance. After the attack, the difference has increased and oscillates between 25 μs and 30 μs. Because these values are bigger than σ_{rel}, we conclude that this attack breaks the clock synchronization. This is the expected result as the value chosen for the delay satisfies (3.14).

Although Fig. 3.7 is useful to show how the time synchronization is broken, it can be obtained just in the context of this simulation and not in a real-life situation. To detect the attack we must restrain the information used to the one available to the slave.

In this section as it was mentioned above the monitor in the slave that collects the arrival times of synchronization messages is considered. Fig. 3.8 shows the interarrival times of synchronization messages to the slave as obtained with (3.8). Before the attack, the interarrival times were constantly equal to RI. The peak in that figure is the first message affected by the delay. However, all the following messages are also affected by the delay, but we can not see it in the figure because the delay is constant, therefore it only shifts the arrival time of the messages, but not the distance between them.

The value of the peak in Fig. 3.8 ($d_{adv} = 50$ μs) is longer than the maximum possible value ($2\sigma_{max} = 20$ μs), therefore, this attack will be easily detected just by analyzing the interarrival times.

After the attack has been detected, some mitigation techniques can be ap-

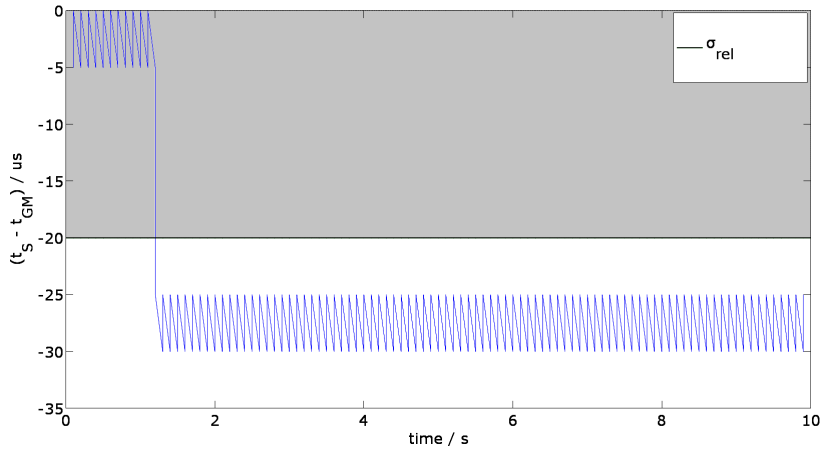

Figure 3.7: Time deviation between the grandmaster clock and the slave clock — constant delay, $d_{adv} = 50\ \mu$s

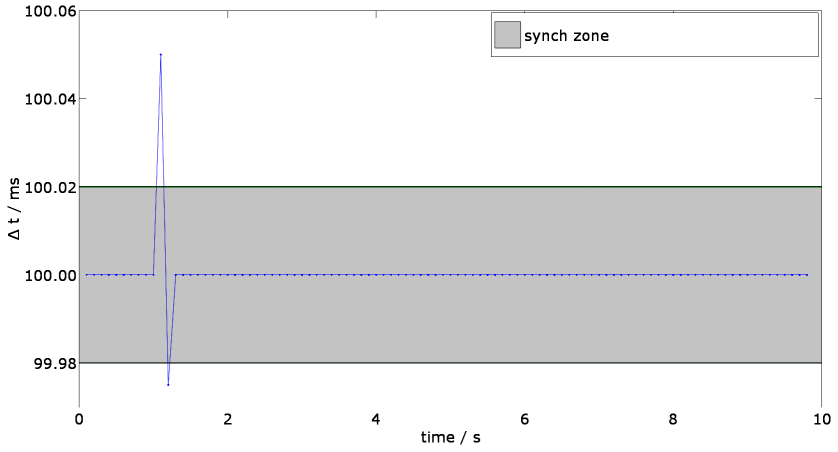

Figure 3.8: Interarrival times of sync messages to the slave — constant delay, $d_{adv} = 50\ \mu$s

plied. For example, if the system has been designed to function in a relaxed mode such as $\sigma_{rel} > 30$ μs then a configuration agent [94] can carry out this change of mode. Thus, even though the synchronization is deteriorated, the system still behaves in a predictable manner.

B. Linearly increasing delay

Now, a delay that increases linearly with every synchronization message that arrives to the router is simulated:

$$d(n) = d_{adv}n, \tag{3.19}$$

where $d_{adv} = 1$ μs and $n = 1, 2, ...$ for all messages after the attack starts.

In Fig. 3.9 it can be seen how the initial delay $d(1) = d_{adv}$ is not big enough to break the synchronization, but after enough RIs it is. However, in this case, as compared to the previous one, the attack cannot be detected just by inspecting the interarrival times of synchronization messages to the slave. This can be seen in Fig 3.10: the effect of the attack on the interarrival times is so small that the slave might as well confuse it with a drifting in the grandmaster clock. This attack puts the slave in a state in which it is not aware of the fact that it is going out of synchronization when, indeed, it is.

For this case we conclude that other parameters, different than the interarrival times, should be used to be able to detect the attack. If we want to keep the detection local to the slave, we can assume that the Monitor has been gathering data for some time before the attack happens and use those values to obtain statistical parameters.

For example, we can examine the variations of the calculated value of the RI obtained using (3.9). In Fig. 3.11 it is shown how this value is consistently increasing. Therefore, in order to detect the attack we need to be able to identify this kind of pattern. Because the increase on the value is so small, it could probably not be used as a sole criterion to detect an attack, but it can be one of a multi-criteria detection method. A further study could be done comparing the amount of RIs that the attack will needin order to break the time synchronization with the number of RIs that the monitor needs to detect the attack.

Independently of the detection capabilities, here it is demonstrated again how choosing a σ_{rel} longer than σ_{max} gives the network some time to react to the attack even before the synchronization has been broken. The inner difficulty of attack detection only by means of local monitoring was shown, especially in the case of a smarter attack that introduces a linearly increasing

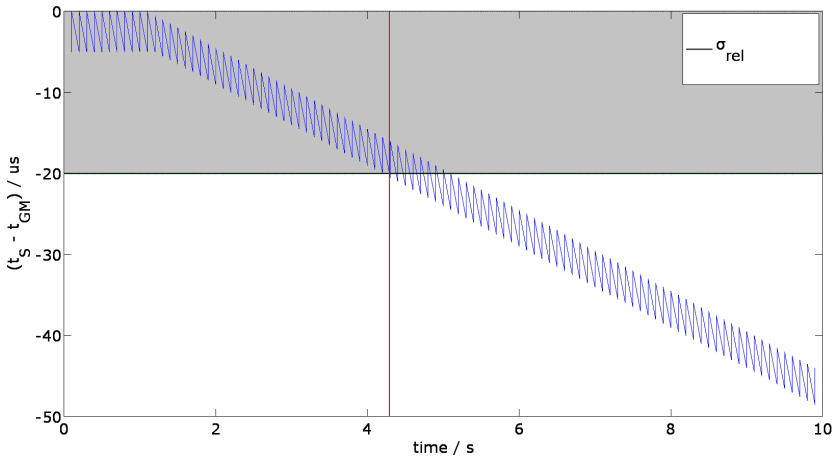

Figure 3.9: Time deviation between the grand master clock and the slave clock for the case of linearly increasing delay

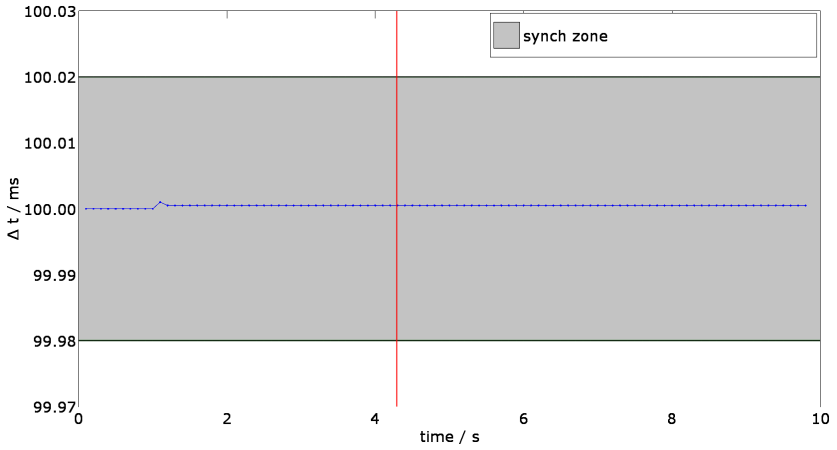

Figure 3.10: Interarrival times of sync messages to the slave for the case of linearly increasing delay

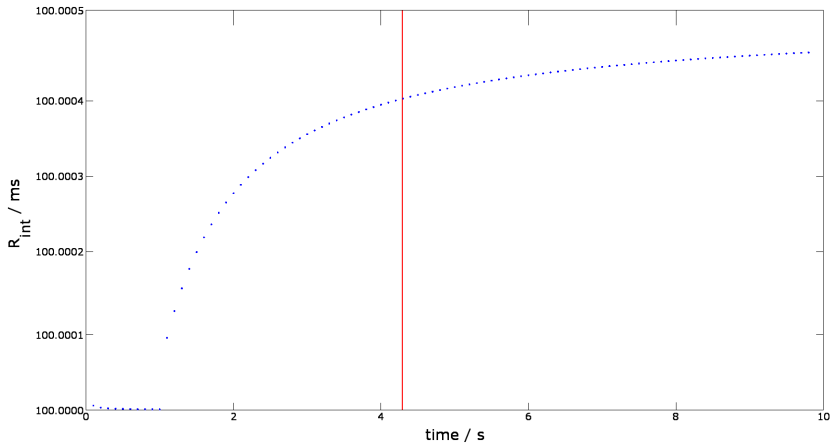

Figure 3.11: RI as obtained in the slave for the case of linearly increasing delay

delay [113]. The proposal is to use mitigation techniques, e.g., like the ones presented in Section 3.3.3, to cope with attacks that cannot be detected by distributed monitoring in nodes. If we apply the mitigation technique described in Section 3.3.3 to the topology used in the simulations, it is demonstrated that in the worst case scenario (i.e., low contention in the network) this method is able to detect attacks introducing delays longer than 100 μs. This value is way above the minimum delay required to break the synchronization and therefore, the technique cannot be used to prevent time synchronization breaking. Nevertheless, it has two important benefits in the case of linearly increasing delay. First, it actually allows the slave to detect an attack that it is not possible to detect with the monitoring approach and secondly, by including the knowledge of this upper limit of the delay in the design, the system can be prepared for this scenario, i.e., has an approach for returning into safe mode.

3.4 Answering the Question

First, a way of approaching security issues was proposed and a the threat model combining main aspects to condsider was formalized, i.e., a use case and system assets, adversary model, adversary goal and system vulnerabilites. The model was discussed using the example of an industrial network. Next, the

presented ATA analysis demostrates possible ways to break clock synchronization. In this thesis, the focus is on the delay attack, which is just a small part of the whole tree. The delay attack is one of the most chanllenging, as it does not require message modification or creation, thus countermeasures are not trivial. Finally, an answer to the chapter question was found, i.e., it is possible to break clock syncrhonization and it was demostrated how. Namely, the possibility to break the clock synchronization mechanism established using IEEE 1588 standard was identified, by performing an ARP poisoning attack and a subsequent selective delay attack. Possible mitigation techniques that can protect the network from the attack or can protect the clock synchronization even in case of a successfully performed combination of the attacks were also considered. The mitigation techniques take into account the derived requirements from the targeted industrial applications. The evaluation using AVISPA show that this scenario is indeed possible. This conclusion demonstrates the necessity to provide industrial networks with appropriate protection measures. Next, a traffic monitoring approach was proposed as a mean of detecting a delay attack. Simulation of the monitor showed the possibility to detect the delay attack in the case of imposing a constant delay. Simulation results also showed that if the system is designed with a relaxed synchronization condition mode, it can help with mitigating the consequences of a delay attack once it has been detected. Furthermore, the results also demonstrated that in the case of linearly increasing delay, adversary influence can remain undetected. Therefore, more sophisticated detecting techniques are needed to detect such attacks. Algorithms for growing trend detection can then be a possible solution for coping with non-constant delays.

Clock synchronization is an essential part of all networks with real-time requirements. Basically all industrial networks have real-time requirements, and thus if there is a way to break clock synchronization, the method will disrupt the network functionality, and is applicable to a range of use cases. IEEE 1588 is typically used to provide clock synchronization, but lacks security mechanism. Not even the Annex K, which has been introduced to enhance security, is capable of handling delay attacks such as the ones evaluated in this chapter. However, there is nothing in IEEE 1588 which prevents us from using a monitor and thus the proposed solution can be used to enhance the standard.

Chapter 4

How Can Broken Clock Synchronization Influence System Safety?

Systems around us are no longer separate units, but part of larger cooperating systems, connected to public or semi-public networks, where information errors can propagate throughout the system in many, sometimes unpredictable ways. The system control no longer depends on human operators only, but also on other systems they are connected to. Most systems involve multiple stakeholders, have dynamic system reconfigurations, and unpredictable operating environments. Such software-intensive systems are also referred to as Systems of Systems (SoS). SoS are a running example in this chapter.

To enable dependability guarantees in systems like this, both safety and security have to be satisfied through dedicated efforts in detecting and recovering from failures. Historically, safety and security have been addressed by two distinct communities, each focusing on their own methodologies, techniques and tools for system development. However, already in the 1990s, some researchers noticed commonalities between these dependability properties [114, 115, 116, 117, 118] and tried to provide a way to reason about them in a unified way. There have been several efforts to (re)use already existing techniques, trying to identify similarities, as well as differences, introducing perspectives on how the properties of safety and security can be harmonized [119, 120, 121, 122, 46, 123]. Given the existing literature, one can

notice that the interdependencies between safety and security are increasingly understood and accepted, however there is still a lack of dedicated methods and approaches that consider safety and security jointly while designing and developing software-intensive systems.

Let us consider a fully autonomous quarry as depicted in Fig. 4.1, where battery-powered electric load carriers operate in cooperation with other machines. These carriers are expected to follow a path, load/unload, transport, avoid waiting, avoid carrying load over longer distances than needed and avoid any unnecessary movements including rework. It is expected that a fleet of these unmanned carriers will jointly be able to move the same amount of load as one large haul truck and if one of these carriers would go down, the loss to the overall quarry production should be much smaller, compared to the loss of a large haul truck. However, since these machines are assumed to be fully autonomous, all possible processes and scenarios need to be documented and analyzed, taking into consideration all new critical situations. Considering different types of communication in such systems (e.g., GPS, machine to server communication, machine to machine communication, etc.), one has to account for all possible threats coming from the security domain affecting the safety of the system, as well. For instance, just by delaying message delivery, one could provoke unexpected events, such as carrier crashes, carriers being unavailable at the expected time, most likely causing loss of production and decreasing the overall efficiency.

In complex SoS as autonomous quarries are, one has a set of assets to protect which also includes communication and protection of clock synchronization as these are paramount requirements for proper scheduling of message exchange [124]. In these systems, it is important to be able to keep track of fresh data [125] and assess data validity based on the time it has been obtained. Furthermore, if the clock synchronization is broken the whole network becomes disturbed. Therefore, securing clock synchronization is an emerging and important challenge to consider when enabling reliable and predictable communication in complex SoS. Moreover, it is evident that the security level of clock synchronization affects the SoS safety, as broken clock synchronization can lead to potential hazards related to system correctness, availability and reliability.

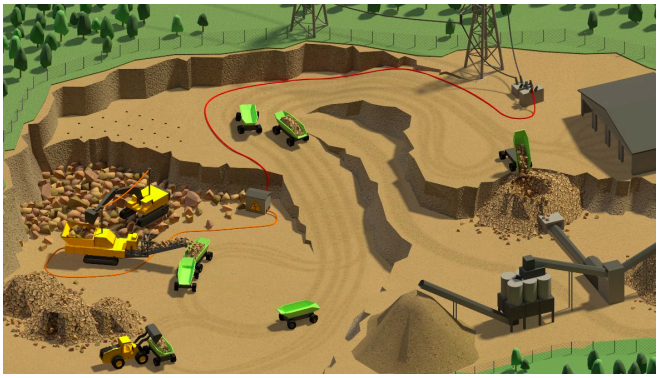

Figure 4.1: An example of an autonomous quarry [1]

4.1 Autonomous Quarry

A fully autonomous quarry is considered to illustrate and motivate the findings. It is an example of a SoS as it includes several subsystems: wheel loaders, autonomous electric carriers, rock crushers, charging stations, trucks, a remote control room, etc. (Figure 4.1), where each subsystem is a stand-alone system with its own function and purpose, but capable of sharing its resources and capabilities to create a new, more complex SoS.

A quarry operation is organized in six main phases: *(i)* site establishment, *(ii)* exploitation, *(iii)* processing, *(iv)* distribution, *(v)* maintenance and *(vi)* reclamation [126]. Quarries are often situated in remote areas without cellular communication coverage. Additionally, the environment is harsh due to dust and solid materials that implies obstructed line-of-sight for wireless communication. The environment and topology change over time imposing challenges on providing reliable maps, location, path and route data for the involved subsystems. Reliable communication is an important asset to enable accurate and correct decisions to be made. Otherwise any communication disruptions might lead to inaccurate actions, negatively affecting production, cost and most likely safety and security at the site. Therefore a reliable and predictable wireless link is of absolute importance. Different types of wireless communication technologies between autonomous machines are possible: *satellite* (provides good coverage, but has limited bandwidth, high latency and cost), *cellular* (comes

with high bandwidth, but the coverage might be hard to ensure in remote areas) and *dedicated short range communication* such as vehicular ad hoc networks (VANETs). We assume that each construction machine broadcasts its activity (machine type, machine task, operational status) and basic position data (position, speed, and direction). Autonomous electric load carriers run on a battery and it is important to enable charging to prevent any disruptions in normal production. They are also equipped with a vision system, which allows the machine to detect humans and obstacles in its vicinity. The whole quarry can be seen as a complex cooperative SoS, where the machines should work in coherence, synchronized with each other to enable efficient operation and avoid losses in production, equipment or in the worst case a human life.

4.1.1 Identified challenges

An autonomous quarry is an open safety-critical SoS in which expected production and operation of all subsystems rely on a correct and timely exchange of messages over the communication network. Complexity of SoS brings many challenges in its analysis and evaluation in terms of safety and security [127, 128]. Due to SoS being an open system, and existence of communication infrastructure when providing safety assurance one has to think about threats emerging from the security domain.

An autonomous quarry is an open safety-critical SoS in which expected production and operation of all subsystems relies on correct and timely exchange of messages over the communication network. Therefore, timeliness and reliability are one of the main SoS properties. Both attributes are tightly connected to the communicational part of the SoS. Vulnerabilities related to communication channels open up possibilities to breach SoS and cause new or contribute to existing safety hazards. This thesis focuses only on the security challenges emerging from the communication, as it goes beyond traditional safety analyses. Additionally, given the existence of communication network, one can also expect attempts to breach security of the system as well. We are interested in learning about malicious attacks on the clock synchronization mechanisms in the system, and its effect on system safety. It is paramount to enable exchange of fresh and valid messages within given time-slots, otherwise it may lead to several hazards including loss of production due to electric load carriers not being charged, collisions within the site given that an old message regarding the position of machines has been sent, etc.

4.2 Why Considering Clock Synchronization?

Complex software-intensive SoS include different communicational protocols, e.g., TT-Ethernet, WirelessHART, as well as other protocols supporting the communications, e.g., protocols for scheduling and network reconfiguration. Clock synchronization is usually assumed to be in place by many of these protocols, e.g., in scheduling. Therefore, an additional protocol is usually deployed to establish and maintain clock synchronization. There are several protocols widely used in industry, e.g., NTP [129], IEEE 802.1AS [68] or PTP [64]. However, since the same protocol is used for many applications, an attack exploiting its vulnerability and causing clock synchronization being broken, can be reused by an adversary, and thus is appealing.

Even if the cause of clock synchronization being broken comes from the security domain, it has direct implications on system safety. If a node is in an unsynchronized state, it cannot propagate timely alarm messages, and hence it causes a failure. Thus, clock synchronization is an excellent example of safety and security overlapping each other.

In the SoS presented in Section 4.1, there are several types of wireless communication. Often, nodes have time-slot allocated access to the communication channel. Also, a control unit needs to assess data freshness by its timestamps to make a decision. Therefore, if an electric load carrier is brought into an unsynchronized state, depending on the offset it has compared to other network participants, it can time stamp data with an incorrect time or even try to access the channel in someone else's time-slot, with resulting collisions. In the first case, a control unit outside the vehicle can make an incorrect decision based on the received data. In the second case a decision required to be made due to the change of the environment of the carries can not be provided as no information is received due to data collisions.

4.3 Joint Clock Synchronization Argument

To enable a joint safety and security assurance of software-intensive SoS such as the one described in Section 4.1, we use the well known STPA-Sec approach and combine it with GSN to get a structured way of deriving safety cases. We focus only on network communication vulnerabilities resulting in clock synchronization being broken. More specifically the work builds upon the first step of existing STPA-Sec, which is used to establish foundations for a security analysis when the threat model is formulated. This model is complemented

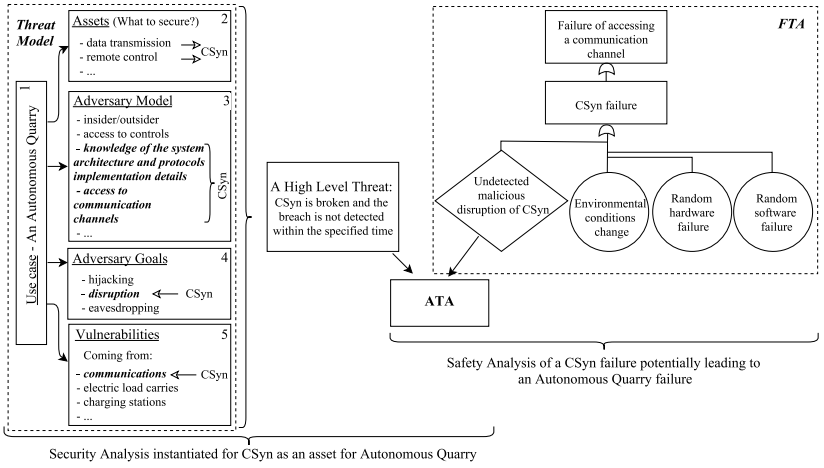

Figure 4.2: Joint safety and security assurance approach for an autonomous quarry and its asset — clock synchronization

with ATA and FTA to illustrate how threats derived from a threat model can lead to hazards for the considered use case.

In order to identify high-level threats for the autonomous quarry use case, a threat model consisting of five blocks as depicted to the left in Fig 4.2 was built [130]. The already existing structure for threat modeling of industrial systems [14] is complemented with a relevant set of vulnerabilities. The first block describes a use case and its specification. All components of the model are tightly dependent on the use case specifics and built upon it. The second block represents assets of the use case, i.e., what should be secured. There are several relevant assets required to assure data transmission, but this thesis focuses on clock synchronization as it needs to be guaranteed for all use cases with any kind of real-time requirements. Remote control, enabled by the transmitted data can also be guaranteed only if a control unit can judge about information freshness based on its timestamps. Note, that the description of threat model components include only information relevant to clock synchronization being broken, as we target to state the problem and illustrate a general approach.

The third block represents an adversary model, i.e., a collection of assumptions about a possible adversary. The knowledge of the SoS is relevant for clock synchronization, since the assmption is that the adversary might be familiar with the SoS architecture and know which protocols are used for clock

synchronization, meaning that the adversary can take over a communication channel. The communication part of SoS is especially vulnerable as system security is addressed under the assumption of external dependencies being secured and merging several systems into a complex SoS brings up challenges on protection of SoS interactions.

Generally, adversary goals can be related to hijacking, disruption or eavesdropping as shown in the fourth block. Disruption is identified as the main goal when intentionally breaking clock synchronization. The fifth block describes possible vulnerabilities. Since SoS, such as the autonomous quarry, rely on communication infrastructure (i.e., satellite, cellular and/or VANET), we isolate communication as the main vulnerability that needs to be addressed in order to protect the assets of the system.

Having the threat model completed, we can derive a high-level threat related to the considered SoS asset: *clock synchronization is broken for a time sufficiently long to allow the adversary to reach its goal and is not detected as broken within the specified time, i.e., before consequences are inevitable*. To analyze this threat, further ATA should be performed to identify possible single attacks realizing this threat.

The complexity of the SoS structure requires a revised taxonomy of hazards in order to identify and analyze them properly [131]. If we consider a high-level FTA, an SoS hazard can be decomposed into *single* and *emergent* hazards. Single hazards are usually related to a particular system within the SoS. Conversely, emergent hazards are the results of the integration of several systems into SoS, which is exactly the case for hazards coming from SoS communications. Emerging hazards, in turn, can be decomposed into *reconfiguration*, *interoperability* and *integration* hazards. Reconfiguration hazards might happen when switching from one state to another in the SoS. Interoperability hazards are emerging from possible misunderstanding between systems within the SoS, i.e., if a system command is interpreted in a way inconsistent with an intent of another system in the SoS. Interoperability hazards are thus related to a SoS failure caused by cooperation of correctly working systems. The last type, integration hazards, is caused by the integration of several systems into the SoS, e.g., by their interactions or through resources sharing. Integration hazards are related to a SoS failure caused by its system failure within the integrated structure. This group has a subgroup related to *interface* hazards, i.e., hazards caused by a failure at the system input/output.

The right side of Fig.4.2 depicts a part of FTA, i.e., a branch of an interface hazard. One of the faults leading to this hazard can be a failure of accessing a communicational channel between systems, as in this case the system cannot

receive and transmit data at the correct time instances under the assumption of time-slotted access to the channel. A traditional FTA would have three single undesired events leading to this failure, namely a random hardware failure, a random software failure and a failure caused by the environment. For example, a sudden drop in temperature can put a node in an unsynchronized state, as temperature can influence the clock. However, for the considered use case, the harsh environment is mostly caused by dust. To include threats causing clock synchronization failure, we consider also an undetected malicious disruption of clock synchronization. A further decomposition of this failure is done by ATA, identifying all intentional attacks causing a clock synchronization disruption. We state that there is a branch in FTA leading from the clock synchronization failure to an SoS failure. We consider only a subtree of the clock synchronization failure to demonstrate its connection to ATA, since an SoS failure can potentially lead to a hazardous event. For example, a collision of electric load carriers during their operation phase that might be caused by an SoS failure, namely incorrect positioning of other vehicles due to disrupted communication with a control unit in turn due to clock synchronization being broken, implies a hazardous event. Therefore, clock synchronization protection is relevant for safety assurance of the SoS. In general, Fig. 4.2 demonstrates how an outcome of the security analysis (the left hand side in Fig. 4.2), can be used as an input for the safety analysis (the right hand side in Fig. 4.2).

Given the approach presented above, communication is identified as a potential vulnerability that could lead the whole system into an unsafe state. In Fig. 4.3 an argument for a clock synchronization assurance case is depicted expressed with GSN.

The presented argument is an extension of the general argumentation for a safety assurance case presented by Troubitsyna [132]. The author follows the STAMP approach and considers three subgoals supporting the strategy of conducting STAMP-based analysis for safety assurance, that include: *(i)* assurance that the controlling software has a correct model of the controlled process; *(ii)* assurance that the logic of the controlling software is correct; and *(iii)* assurance that the controlling actions are implemented correctly. The first subgoal can be achieved via a strategy aiming to assure that malicious and accidental faults do not distort the model of the controlling process, considering safety and security together. This thesis work follows the STPA-SEC approach and therefore proposes to add one more subgoal to this strategy in order to adjust the method to consider security caused failures, namely $G1$ — *SoS communications are acceptably secured.*

To assure this goal, a strategy and a goal are proposed, namely $S1$ — *an*

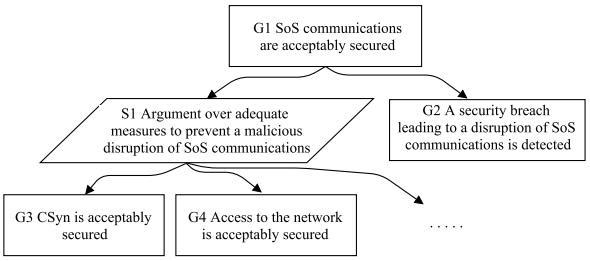

Figure 4.3: A clock synchronization argument

argument over adequate measures to prevent a malicious disruption of SoS communications, and *G*2 — *a security breach leading to a disruption of SoS communications is detected*. The latter is formulated as a goal, and it is not considered to develop it further in this work, and therefore we cannot address it as a strategy. However, it is needed in case the security attacks could not be prevented in order to make the SoS able to detect the existence of the breach and start switching into a safe state. In some cases, an attack needs time until its consequences become significant and if detected before, the harm of the attack can be minimized. For example, to have consequences, clock synchronization should be broken for some resynchronization intervals and the amount of intervals needed to disrupt the system depends on the system being under attack, e.g., how often it communicates. *S*1 can be realized by several subgoals, and it should be noted that only two of them are presented: *G*3 relating clock synchronization and *G*4 relating to access control. It is not expanded it further as the purpose of this work is to highlight the significance of communication safety and security assurance, as well as demonstrating how their overlap can be used to enrich the analysis and indicate crucial importance of considering clock synchronization with respect to communication security and SoS safety.

4.4 Answering the Question

Given the increasing integration of a large number of computing and sensing devices and systems with different types of network infrastructure, security has become a serious issue while ensuring system safety in autonomous systems of systems, including safety-critical construction equipment. In order to guarantee safety in such systems, one has to include reasoning about security as

well.

In this chapter, findings on exploring interdependencies between safety and security, focusing on threats towards breaking clock synchronization and its possible effect on system safety in complex systems of systems are presented. An approach where security and safety analyses are combined into one process while addressing a common high-level goal is proposed, in this case protecting clock synchronization in an autonomous quarry with several cooperating vehicles. Further, a joint safety and security clock synchronization argumentation using goal structuring notation is presented, where an extension by including communication aspects into the reasoning is proposed as it proves to be an important asset in complex autonomous systems of systems. Thus, the answer to the question posed in this chapter heading is that security influences safety by adding additional causes to hazards and thus, by changing the probabilities, risk assessment is required to be re-evaluated, and ways of systematic handling of security causes along with techniques that decrease those probabilities to acceptable ones need to be known.

Chapter 5

How to Predict the Adversary Behaviour?

Security implies malicious intent, thus there is always an adversary to consider. Security techniques are built on assumptions about adversary capabilities. However, adversary behavior is not static, thus even knowing the initial setup, interaction between an adversary and a system or more particularly its protection mechanism needs to be investigated. Based on such an analysis one can reason about the adversary behavior and consequently try to predict it. In this chapter the interaction between an adversary and the proposed security solution, i.e., the monitor, is considered.

First, the monitor model proposed in this thesis is specified. The model includes several states and rules for transitions between these. Also assumptions for the model are discussed. Next, given the model of the monitor interacting with an adversary, game theory is applied to investigate and formalize the adversary-network interaction considering clock synchronization protection issues. In this thesis, three possible strategies for a network monitor are considered, two of them are logically equivalents of pass and drop respectively, but the third one is called quarantine and allows a system to check the link and determine whether it is under attack and/or employ mitigation techniques. Also multiple interaction games are considered that allows consideration of different ways of imposing delays. Mizrahi proposes a multipath data collection procedure as a prevention technique against delay attacks. In this chapter the focus is on monitoring techniques as a way to secure clock synchronization, although multipath data spreading can be also used in the considered industrial

networks as a way to provide the desired level of reliability and availability. In addition, a game theory framework allowing comparison and evaluation of different types of attacks targeting clock synchronization, namely, constant, linearly increasing and random delays is proposed.

5.1 Monitor concept

The concept introduced in this thesis evolves having a monitor located in each node, collecting statistics about an offset measured according to IEEE 1588. As shown in Fig. 5.1, there are three possible states for the network, namely, normal (NS), quarantine (QS) and anomaly detected state (ADS). In NS, the monitor believes that there is no active adversary in the network and clock synchronization has not been broken. In ADS, the monitor is assured that there is an adversary in the network affecting clock synchronization. Finally, in QS there are indicators of anomalous activity in the network, but the monitor is not assured and needs additional arguments in favor of an attack being performed. There are rules for switching between the described states. These are based on evaluation of relevant indicators, i.e., characteristics calculated by the monitor based e.g. on observed samples of calculated offset values. There are two types of indicators: one group we call main indicators, which are monitored constantly; and a second group termed additional indicators used only in QS. An indicator has two thresholds, a low and a high. Thus, the following rules are used to make a decision, Fig. 5.1:

- *Rule 1.* If a certain number, K_{QS}, of the main indicators are above their corresponding low threshold, the system is switched into QS.
- *Rule 2.* If more than K_{QS} of the indicators are above their low thresholds, the system is switched into ADS.
- *Rule 3.* If a certain number K_{ADS}, where $K_{QS} > K_{ADS}$, of the indicators are above the corresponding high threshold, the system is switched into ADS.
- *Rule 4.* If any of the cases described below is true, the system is switched from QS mode to ADS:
    1. If there are positive indicators of network anomaly according to the additional indicators deployed in QS;
    2. If *Rule 2* or *Rule 3* can be applied.

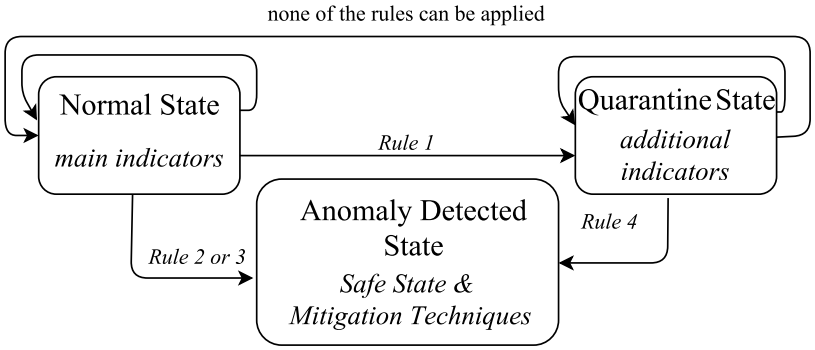

Figure 5.1: Monitor states and connections between them

In the analysis presenteed in this chapter, the mean and standard deviation of the delay are used as main indicators:

$$\overline{\sigma} = \frac{1}{N} \sum_{i=1}^{N} o_{meas,i}, \tag{5.1}$$

$$\sigma = \sqrt{\frac{\sum_{i=1}^{N} (o_{meas,i} - \overline{\sigma})^2}{N-1}}. \tag{5.2}$$

where N is the number of currently considered resynchronization intervals and o_{meas} is the measred offset.

The assumption is that the decision about switching the states in the monitor depends on several thresholds for mean and standard deviation of the calculated offset. Therefore, the probability of being in each state can be taken from the probability of the value of the delay belongoing in a corresponding interval. This probability we can calculate by knowing the distributions for "natural" delays occuring in the network and using appropriate variables for the adversary state probability. For each indicator, there are two thresholds: low and high, i.e., T_{σ}^{L} and T_{σ}^{H} are the low and high thresholds for the standard deviation whereas $T_{\overline{\sigma}}^{L}$ and$T_{\overline{\sigma}}^{H}$ are the low and high thresholds for the mean respectively. If one of the indicators is above the corresponding low threshold but lower than the high threshold, it is only suspicious. If the characteristic is

above the high threshold, we assume that an attack is detected. Considering the indicators and notations used in this chapter the switching rules could be simplified to the following ones:

- *Rule 1.* If one and only one of the monitoring characteristics is above the corresponding low threshold, the system is switched into QS: $\overline{\sigma} > T_{\overline{\sigma}}^{L}$ or $\sigma > T_{\sigma}^{L}$.
- *Rule 2.* If two of the monitoring characteristics are above the low thresholds, the system is switched into ADS: $\overline{\sigma} > T_{\overline{\sigma}}^{L}$ and $\sigma > T_{\sigma}^{L}$.
- *Rule 3.* If any of the monitoring characteristics is above the corresponding high threshold, the system is switched into the ADS : $\overline{\sigma} > T_{\overline{\sigma}}^{H}$ or $\sigma > T_{\sigma}^{H}$.
- *Rule 4.* If any of the cases described below is true, the system is switched from QS to ADS:
  - If there are positive indicators of network anomaly from the additional checking techniques that were deployed as a result of switching to QS;
  - If *Rule 2* or *Rule 3* can be applied.

5.2 Game Setup

Game theory is a mathematical theory describing possible interactions and/or cooperation between rational actors and studies the decision-making process along with the corresponding outcomes. T. Mizrahi applied game theory to analyze the influence of a delay attack on (PTP) [67]. The author considers two strategies for a node, it can pass or drop a synchronization packet. However, the presented game consists of one interaction between an adversary and a node making decisions. A game is created once players, i.e., an adversary and the monitor, and their strategies are specified. In order to formalize the interaction between an adversary and the monitor, the considered system model along with the participants of the game should be set.

Network Model. We assume that IEEE 1588 is used for establishing and maintaining clock synchronization. This means that clock synchronization is achieved through message exchange between a grandmaster and a slave. First, the grandmaster sends out a time stamped `sync` message, and when the slave receives the message, it also timestamps it to determine the arrival time and

sends out a `delay_req` message, which is timestamped as well. Finally, when the grandmaster receives the message, it timestamps and sends out a `delay_resp` message. In the end of such an exchange, both the grandmaster and the slave have time stamps of the `sync` and `delay_req` messages at the moments of transmitting and receiving. Knowing these four values and assuming absence of asymmetrical delay, the offset between the two clocks can be calculated.

We use a distribution analysis of the measured offset, o_{meas}. The offset is measured by a slave according to IEEE 1588 and the monitor in each slave is saving the measured offset in every RI and calculate statistics based on it [22]. In this chapter, we consider only two basic statistic parameters, namely mean and standard deviation. We assign thresholds for each of them, which we refer to as indicators. We say that an indicator is positive when it is above the allowed threshold and that it is negative otherwise. According to our assumption, the resulting offset measured by a slave consists of three components: offsets related to the clock drifts, related to a natural delay in the communication channel and, finally, related to the adversary. Offsets related to the clocks drift is always present, therefore, without loss of generality, it can be excluded from the consideration. The clock drifts do not affect the overall reasoning and calculations, but changes only the threshold set for making a decision about switching to a different system mode. The considered network is heterogeneous, implying that is contains a mixture of wireless and wired communication links, such that the route between a grandmaster and a slave can consist of both types of links. In wireless channels without fading, the variations in propagation delay was found to have an exponential distribution [133]. In other cases, a different probability distribution needs to be considered. Hence, we model offsets related to nature with an exponential distribution, since wired point-to-point links likely experience smaller delay variations compared to line-of-sight wireless links, and thus the worse-case scenario is considered.

Adversary Model. We assume that the adversary is using a combination of an ARP poisoning attack and a selective delay attack to break clock synchronization, as demostrated in Sec. 3.3. ARP poisoning exploits vulnerabilities of the ARP protocol that is used to translate between MAC and IP addresses, which enables a man-in-the-middle attack for protocols based on IP. To be able to break synchronization, the adversary does not need to forge or modify the synchronization messages, only delay it. This is a reason why encryption cannot help against this type of attack, as the timestamps already incorporated in the message do not need to be modified.

Quarantine State. The idea of introducing QS is to be able to react to an

attack before it breaks clock synchronization. If there is an indicator of any abnormal behavior, the system puts the suspicious link/route into quarantine or switches it to QS. In this state, the link is still used, but the system simultaneously tries to find out whether the abnormal behavior was an error or a consequence of a malicious interference with the network. This can be achieved by e.g., using additional techniques to provoke an adaptive adversary in such a way that it reveals itself or/and by further monitoring the network/link characteristics. In the chapter, we consider only the second option, leaving the first one for future work. However, provocation will only work in case of non-fixed strategy of the adversary. Therefore, in QS besides the two initials indicators, other additional indicators can also be considered, e.g. like monitoring the maximum return time for the messages or checking the current environmental conditions. The benefits of this mode includes the possibility to check what is going on with the link and try to mitigate any problems smoothly, while still continuing to do the best possible for clock synchronization.

In Sec. 3.3, the term Relaxed Mode was introduced. Relaxed Mode implies a degraded quality of synchronization, but also gives the system the opportunity to recover to a safe state. Quarantine can be considered as an extension of the previously introduced Relaxed Mode. QS not only gives the system an opportunity to recover if it is under attack, but also gives the opportunity to make a decision about whether there is an adversary in the network or not. Relaxed Mode was introduced by using relaxed boundaries for the offset between nodes, and a trust coefficient was introduced that can be considered as equivalents of relaxed boundaries for clock offsets. Also it should be mentioned that the system can switch from normal working state to the state where the attack has been detected without entering QS, if the symptoms or indicators of being under attack have significant values or significantly many are positive simultaneously.

5.3 Game Model

The main components of a game are actors and their strategies. This set is complemented with probabilistic functions for game strategies, as this allows binding all components together to analyze the adversary behavior and its consequences.

Three actors are considered in this game. The first one is the adversary, which targets to break clock synchronization and, in the best case, stays undetected. The second one is the network monitor in the node that wants to

stay synchronized with the grandmaster. Note that we consider local detection only in this chapter. The third player is the nature or the environment or the channel. It does not have strategies, but it has a probabilistic distribution of the delays in the channel. These delays can affect clock synchronization or/and mask adversary actions.

By strategy the set of possible delays that can be introduced by the players is understood. The game strategy space can be presented as:

$$S = S_{nat} \times S_{adv} \times S_{mon}, \tag{5.3}$$

where $\times$ is Cartesian product, and S_{nat}, S_{adv} and S_{mon} are nature, an adversary and the monitor strategies respectively.

For an adversary, it is reasonable to impose a delay only in one direction in order to make the delay asymmetrical, since IEEE 1588 can be broken only by an asymmetrical delay. Nature imposes delays in both directions, however, we consider also this delay as asymmetric since this is the most troublesome type of delay. Nature only has one strategy:

$$S_{nat} = \big\{ d_{nat} \mid d_{nat} > 0 \big\}, \tag{5.4}$$

where d_{nat} is a delay caused by nature.

The three different types of delay attacks when deployed at t_{dep}, are presented in Fig. 5.2. During the attack, a delay is imposed every RI, thus different types of delay attacks refer to how the imposed delay varies from one RI to another. A Constant Delay (CD) is the case when the delay is the same, d_{CD} in each RI. When the delay is increasing each RI in a linear fashion, d_{LID}, it is called a Linearly Increasing Delay (LID). Finally, a Random Delay (RD) is a situation when the delay value, d_{RD}, is chosen randomly from an interval of values allowing to bring a targeted node into a unsynchronized state.

Four different strategies for the adversary are considered: no attack, introducing a constant delay, a linearly increasing delay or a random delay. Therefore, the space of the adversary strategies can be presented as

$$S_{adv} = \big\{ S_{adv}^{NA}, S_{adv}^{CD}, S_{adv}^{LID}, S_{adv}^{RD} \big\}. \tag{5.5}$$

The corresponding strategies are defined as:

- No Attack (NA), the related strategy is a set of possible delays, which consist of only one element — 0.

$$S_{adv}^{NA} = 0. \tag{5.6}$$

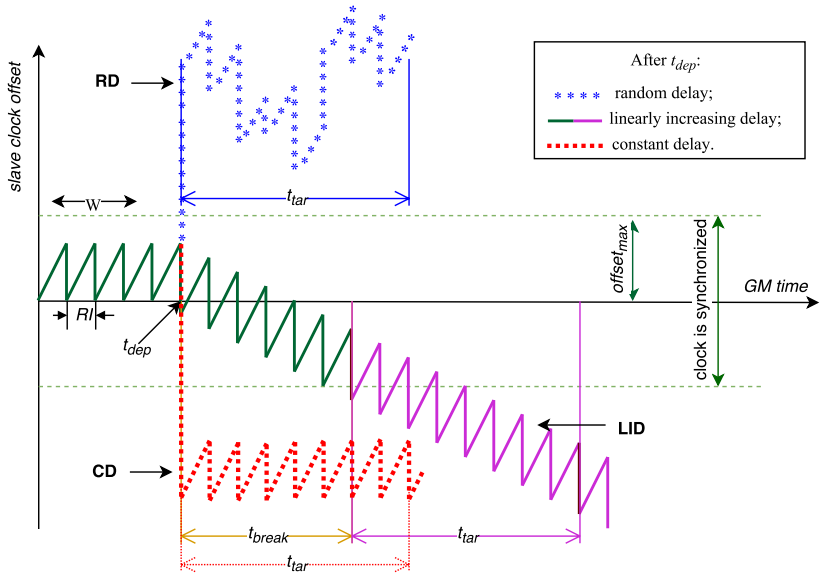

Figure 5.2: Clock synchronization under different types of a delay attack

- Constant Delay, CD

$$S_{adv}^{CD} = \{d_{adv} \mid d_{min} \leq d_{adv} \leq d_{max}\}, \tag{5.7}$$

where S_{adv}^{CD} is the strategy of imposing a constant delay, d_{min} and d_{max} are minimum respectively maximum values for the delay imposed by the adversary, d_{adv}.

- Linearly Increasing Delay, LID

$$S_{adv}^{LID} = \{i \cdot d_{adv} \mid 0 < d_{adv} < d_{max}\}, \tag{5.8}$$

where i is the number of the RI, i.e., an iteration for the adversary.

- Random Delay, RD

$$S_{adv}^{RD} = \{d_{adv,i} \mid 0 < d_{adv,i} < d_{max}\}, \tag{5.9}$$

where $d_{adv,i}$ is a random delay imposed by the adversary at the *i*-th iteration.

For the monitor, three strategies are proposed, link/route is in quarantine, anomaly not detected, anomaly was detected, and thus the space of the monitor strategies can be presented as

$$S_{mon} = \{S_{mon}^{NS}, S_{mon}^{QS}, S_{mon}^{ADS}\}. \tag{5.10}$$

- Link/Route is in QS, so the system switched to suspicious mode and can start applying additional techniques to confirm the attack.

$$S_{mon}^{QS} = \{o_{applied} = o_{meas} \cdot \beta \mid 0 < \beta < 1\}, \tag{5.11}$$

  where β is the trust coefficient. It can be seen that in this mode it is reasonable to question the validity of the value of the calculated offset, o_{meas} is the measured offset and $o_{applied}$ the offset that is used for actual correction. The value of the trust coefficient can be con-nected to the boundaries allowed for two nodes to stay synchronized.

- NS, normal state of functioning for the node. This situation can be described as the previous one but with $\beta = 1$:

$$S_{mon}^{NS} = \{o_{applied} = o_{meas}\}. \tag{5.12}$$

- ADS, in this mode the system is assured that it is under attack, so it can start applying mitigation techniques. The trust in the calculated offset in this mode is $\beta = 0$, as in this case the monitor does not trust the calculated value at all:

$$S_{mon}^{ADS} = \{o_{applied} = 0\}. \tag{5.13}$$

In order to investigate interactions within the game, we assign probabilities to all strategies or states. Probabilities for the adversary strategies are not known initially, even though they can be estimated during the process of interaction, and are set as follows:

$$p_{NA} + p_{CD} + p_{LID} + p_{RD} = 1, \tag{5.14}$$

where p_{NA}, p_{CD}, p_{LID}, and p_{RD} are probability of NA, CD, LID, and RD strategies respectively. One of the targets of the analysis can be these probability estimations, as this can allow to predict adversary behavior.

For the monitor the probabilities can be calculated assuming that attackers' actions are known. As in our model, the choice of strategy for the monitor

depends on the calculated values of offsets and its statistical characteristics, the probability of the choice can be calculated. Corresponding variables are:

$$p_{NS} + p_{QS} + p_{ADS} = 1, \tag{5.15}$$

where p_{NS}, p_{QS}, p_{ADS} are probability of the strategies NS, QS, and ADS respectively.

5.4 Analysis of Adversary Influence

For influences by the communication channel we consider two cases: wired and wireless channels, but as both of them have the same distribution of the delays, the Probability Density Function (PDF) of Channel Delays (ChD) can be modeled by an exponential distribution:

$$f_{ChD}(d_{ChD}) = \lambda \cdot e^{-\lambda d_{ChD}}, \quad d_{ChD} \geq 0, \tag{5.16}$$

where d_{ChD} is a delay caused by the channel, and λ is a distribution parameter.

CD strategy. A certain delay is imposed by the adversary, and it is assumed that there is a discrete set d_{CD} of values with size N_1 from which the adversary chooses the imposed delay:

$$D_{CD} = \left\{d_{CD,j}\right\}_{j=1}^{N_1}, \tag{5.17}$$

$$\sum_{j=1}^{N_1} p_{CD,j} = 1, \quad p_{CD,j} = f_{CD}(d_{CD,j}), \tag{5.18}$$

where $p_{CD,j}$ is the probability that the adversary chooses delay $d_{CD,j}$ from the set and f_{CD} is a discrete function relating a possible delay from the set with its possibility to be imposed. The resulting offset measured by a slave will consist of these two delays: $d_{CD,j}$ and d_{ChD}, The PDF of two independent variables is a convolution of the PDFs of these variables. Therefore, the PDF of the measured offset (without considering clock natural drifts) can be calculated as:

$$\begin{aligned} f_{offset}(o_{meas}) &= \sum_{j=1}^{N_1} f_{ChD}(o_{meas} - d_{CD,j}) f_{CD}(d_{CD,j}) = \\ &= \lambda e^{-\lambda \cdot o_{meas}} \sum_{j=1}^{N_1} \left(e^{\lambda d_{CD,j}} \cdot p_{CD,j} \right). \end{aligned} \tag{5.19}$$

The sum is a coefficient if the set of delays available for the adversary is fixed. Therefore, if we introduce the detection coefficient:

$$k_{CD} = \sum_{j=1}^{N_1} \left(e^{\lambda \cdot d_{CD,j}} \cdot p_{CD,j}\right), \tag{5.20}$$

we can see that, the PDF of the measured offset is proportional to the exponential distribution:

$$f_{offset}(o_{meas}) = k_{CD} \lambda e^{-\lambda \cdot o_{meas}}. \tag{5.21}$$

LID strategy. In the next case, a LID imposed by the adversary is considred. Here the adversary can choose the step of increment from a defined set with the size N_2. Thus, the imposed delay can be described as follows:

$$D_{LID} = \left\{i \cdot d_{LID,j}\right\}_{j=1}^{N_2}, \tag{5.22}$$

$$\sum_{j=1}^{N_2} p_{LID,j} = 1, \quad p_{LID,j} = f_{LID}(d_{LID,j}), \tag{5.23}$$

where i is the number of RI or the iteration of the delay attack. In this case, the PDF of the resulting measured offset can be calculated as:

$$f_{offset}(o_{meas}) = k_{LID} \lambda e^{-\lambda \cdot o_{meas}}, \tag{5.24}$$

$$k_{LID} = \sum_{j=1}^{N_2} \left(e^{\lambda \cdot d_{LID,j}} \cdot p_{LID,j}\right). \tag{5.25}$$

The PDF of the measured offset is again a scaled exponential distribution.

RD strategy. Finally, in the case of an RD attack, we assume that the probability distribution is uniform:

$$f_{RD}(d_{RD}) = \frac{1}{d_{max}}, \quad d_{RD} \in (0, d_{max}]. \tag{5.26}$$

In this case, the PDF of the measured offset can be calculated as:

$$\begin{aligned} f_{offset}(o_{meas}) &= \int_{-\infty}^{+\infty} f_{RD}(o_{meas} - \tau) f_{ChD}(\tau) d\tau = \\ &= \frac{\lambda \cdot e^{-\lambda \cdot o_{meas}}}{d_{max}} \int_0^{d_{max}} e^{\lambda \tau} d\tau = \frac{e^{-\lambda \cdot o_{meas}}}{d_{max}} (e^{\lambda d_{max}} - 1). \end{aligned} \tag{5.27}$$

After introducing the corresponding coefficient for random delay attack, the PDF of the measured offset can be presented as an exponential distribution:

$$f_{offset}(o_{meas}) = k_{RD}\lambda e^{-\lambda \cdot o_{meas}}, \tag{5.28}$$

$$k_{RD} = \frac{1}{\lambda d_{max}}(e^{\lambda d_{max}} - 1). \tag{5.29}$$

For exponential distributions, the mean and deviation are well known, so given these PDFs we can define thresholds for system mode switching. The thresholds defined with knowledge about adversary behavior and possible attacks are ideal ones that can allow the monitor to detect the attack. In reality we cannot obtain them, but nevertheless it is interesting to consider them to see possible ways of interactions between the adversary and the monitor as well as to vary their values to investigate the limits for adversary detection. For the monitor characteristic deviation, the lower threshold is the deviation of the corresponding exponential distribution, and higher thresholds can be chosen by taking the lower threshold and increasing it by γ per cent. For the monitor characteristic mean, the mean of the exponential distribution sets only the middle of the allowed interval, so here we assume that the allowed deviation is γ per cent for setting the lower threshold. For the higher threshold, we again set the allowed deviation to an additional γ per cent of the lower threshold. Therefore, in the case of linearly increasing delay, the thresholds are:

$$T_{\sigma}^{L} = \frac{k_{ID}}{\lambda^2}, \quad T_{\sigma}^{H} = \frac{k_{ID}}{\lambda^2}(1+\gamma), \tag{5.30}$$

$$T_{\overline{\sigma}}^{L} = \frac{k_{ID}}{\lambda}, \quad T_{\overline{\sigma}}^{H} = \frac{k_{ID}}{\lambda}(1+2\gamma). \tag{5.31}$$

One interesting point here is that the obtained thresholds actually depend on the number of iterations, which is logical as the delay also depends on this number. For the case of constant or random delay the thresholds can be obtained by changing k_{LID} to k_{CD} or k_{RD} respectively.

5.4.1 Security Game

Having set a formal model of the game, it is played by exploring possible interconnections and making numerical estimations of the detection coefficients for different adversary strategies.

If the thresholds defined in this section are used for switching strategies in the monitor, the overall system becomes a state machine with probabilities of

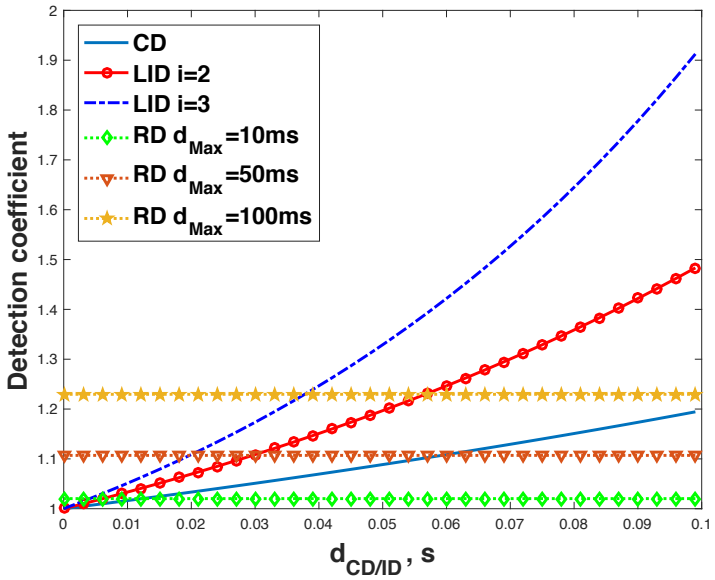

Figure 5.3: Detection coefficients for RD, LID, and RD types of attacks

transition between different states and remaining in the current one. Of course, this result is obtained under strong assumptions about the actions of the adversary, but it still shows that interaction between the adversary and the monitor can be described in a probabilistic manner. For deterministic networks with strict synchronization requirements, probabilistic characteristics are not sufficient, but they allow making intelligent predictions about adversary behavior and can be used to analyze the effects of prevention and mitigation techniques. Strategies for switching between modes can be compared based on the derived state machine configuration, where techniques with higher payoff functions are more efficient. Also this approach allows considering a combination of techniques.

The initial monitor concept in Fig. 5.1 shows the possible states of the monitor related to its corresponding strategies and their interconnections. We assume that after switching to ADS, the system deploys mitigation techniques, the details of which are not considered in this thesis. The trigger for any trans-

action is a new calculated offset value for the current resynchronization interval. Based on such representation, reasoning about system stability can be conducted. If we can calculate probabilities for all transitions for each type of attack or we can compare these probabilities for different types of attacks, it can be used as an attack efficiency metric.

In case there is no adversary in the network, it can easily be seen that with the assumption of an exponential distribution of channel delays in equation (5.30-5.31) and with $k_{ID} = 1$ a reasonable threshold is one allowing the network to eliminate clock synchronization being broken caused by channel characteristics. Therefore, by comparing the thresholds with and without an adversary present in the network, we can see that the adversary attack can be noticed when $k_{ID} > 1$. That is always the case, but the problem here is that this value can be very close to 1, so some qualitative metrics are needed to evaluate and compare different types of attacks. The difference between thresholds calculated with and without an adversary present, demonstrates the influence of the adversary. The bigger the difference, the more exposed the adversary is in the network. To compare the three types of considered delay attacks, we need to compare the values of the related detection coefficient from equations (5.20, 5.25 and 5.29) respectively. More precisely we need to compare their difference to 1, as the approach is more beneficial the more above the value 1 it is. Therefore, for the considered types of attacks, we need to see how sensitive this coefficient value is. To this end, we compare the detection coefficients from the different types of attacks. However we do not set probability correspondence between coefficient ratios and detection ratios. Fig. 5.3 demonstrates the detection coefficients and their dependencies of the value of imposed delays d_{CD} or d_{LID}, which are considered to be the same, to allow comparison of the CD and ID cases. These dependencies are obtained under the following assumptions:

- $N_1 = N_2 = 2$;
- $\lambda = 4$;
- $d_{CD,1} = d_{LID,1}, d_{CD,2} = d_{LID,2} = 20\mu s$;
- $p_{CD,1} = p_{LID,1=0.4}, p_{CD,2} = p_{LID,2} = 0.6$.

Furthermore, several iterations are considered for ID and several values of d_{max} for RD. Fig. 5.3 demonstrates that for RD the level of being exposed in the network depends on the channel characteristics λ and d_{max} — it can be both less and more than the other two considered cases. Therefore, from an adversary point of view, the expediency of applying RD depends on the

application specification: it is beneficial for the adversary if the network can be disrupted by a short-term period of clock synchronization being broken, especially in cases where bigger offsets between the nodes in the network is allowed, as this represents the right side of Fig. 5.3. An adversary employing CD or LID is more exposed in the network the higher the value of the imposed delay. Based on the chosen metric of delay being mean and standard deviation, iterations with LID are more difficult to detect in the network, therefore, the question here is more how fast the fact of clock sycnhronization being broken can be detected. The crossing points of the coefficient curves for different attack types are of special interest, as they show points where it is beneficial for the adversary to make the switch between different strategies. Generally speaking, the adversary wants to stay in the graph with as low detection coefficient as possible, as it indicates a lower exposure in the network. Thus, if from the very beginning, the adversary is applying the CD strategy (blue line without markers), it should switch to the RD strategy to minimize the exposure level at the moment when it crosses the RD line (yellow line with circular markers or orange line with triangular marker). Knowing this reasoning from an adversary point of view, however, is beneficial for the monitor as well. The monitor can predict and model possible adversary behavior and deploy proactive countermeasures given this knowledge. Therefore, if we can make valid assumptions about possible delays, this approach can be applied. The question is how to make those assumptions. One of the options if we have some history of interactions between the adversary and the monitor, is in case the monitor can learn the probabilities and possible delays and calculate the thresholds based on this. A learning period can be used for collecting network statistic and during the interaction with the adversary, the monitor can learn its properties and react accordingly, causing the adversary to invest more resources into adaptability and learning to be able to break synchronization.

5.5 Answering the Question

The proposed monitor model is generic enough to accommodate requirements from particular applications. The quarantine state and two types of thresholds are introduced in order to provide a flexible solution that can reflect several degrees of trust in the current precision of clock synchronization. In this chapter, a formal analysis of the interaction between an adversary targeting clock synchronization via a selective asymmetric delay attack and a network monitor collecting statistics of measured offsets according to IEEE 1588 has been

presented. A game theory framework is proposed as an evaluation tool for the described interaction. The detection coefficient is identified as an appropriate evaluation metric for comparing different adversary strategies. Based on this approach, it is possible to say which strategy is most beneficial for the adversary depending on the set of game configurations. However, knowing this data, the monitor has a possibility to deploy appropriate protection techniques. Also, as was mentioned above, actors can learn each others' behaviour, and therefore, the adversary can try to predict the monitor as well. Thus, the answer to the chapter question is that it is possible to make assumptions about adversary behaviour by applying game theory under the assumption that the adversary acts rationally.

Chapter 6

How to Define the Monitor Strategy?

Using distributed monitoring as a tool to detect that the network is under attack and prevent propagation of adversary influence by deploying relevant mitigation techniques is the core contribution proposed in this thesis. The main focus of this chapter is the delicate balance between the time needed for a monitor to detect an attack and the time needed for an adversary to reach its target; to keep the network unsynchronized sufficiently long to damage the application. Furthermore, the attack detection procedure is formulated as a detection theory problem. This allows introduction of an additional indicator that can be used by the monitor to confirm if an adversary is present in the network. An investigation of how a monitor can set thresholds for making a decision about suspicion of adversary presence in the network by continuous monitoring of indicators is also presented. Based on the ways those thresholds are set, the time chase between an adversary and the monitor can be evaluated.

However, although a general way to threshold the indicators is considered, it is not straightforward how to generalize the outcome of the analysis and thereby estimate the efficiency of each indicator. There are two main sources of uncertainties. Firstly, each particular application has its own parameters, functionality and precision, which dictates the load for each node, and thereby, defines, e.g., for how long an adversary needs to perform an attack to cause consequences or how many components that needs to be faulty in order to affect safety. Secondly, an adversary model was used as input for some indicators, and thus, the considered adversary strategies and attack space limit the

approach and affects the outcome of the analysis. Therefore, a sensitivity analysis is needed to determine how different inputs are affecting the thresholds for the indicators to be able to generalize the results. This chapter presents an analysis of monitor indicators and an investigation of their applicability. Within the analysis, the dependencies of the thresholds of the indicators with respect to the adversary model and the use case are investigated. To identify connections to a use case specification, an example of a CPS is considered.

6.1 Indicators Analysis

The mean (5.1) and standard deviation (5.2) of the calculated offset introduced in Chapter 5, are easy to obtain indicators that can be used to detect some types of attacks. Although these two indicators are not independent characteristics of the offset distribution, they are a natural initial choice in attempting to detect a new trend in the distribution, as they can be calculated easily and reflect the main changes.

As attack discovery is connected to the monitor checking the indicator values, we need to analyze how fast these are changing with response to a delay attack. To analyze how the indicators behave with time during the attack, we divide the measured offset, o_{meas}, into two components: one caused by natural factors (e.g., natural clock drift and channel delay caused by, e.g., environmental conditions), o_{real}, and one caused by the attack, o_{attack}. The offset measured in a node at the i-th RI is the sum:

$$o_{meas,i} = o_{real,i} + o_{attack,i}. \tag{6.1}$$

In the similar manner we split the indicators. Let j be a RI when the attack was deployed, i.e., offset values calculated starting from the $j+1$ RI and later have an error caused by the adversary. The mean then can be presented as:

$$\overline{\sigma}_k = \frac{1}{k}\sum_{i=1}^{k} o_{real,i} + \frac{1}{k}\sum_{i=j+1}^{k} o_{attack,i}, \tag{6.2}$$

where the first term represents the indicator value caused by nature and the second one by the attack. In other words:

$$\overline{\sigma}_{real,k} = \frac{1}{k}\sum_{i=1}^{k} o_{real,i}, \tag{6.3}$$

$$\overline{\sigma}_{attack,k} = \frac{1}{k} \sum_{i=j+1}^{k} o_{attack,i}. \tag{6.4}$$

Having equation (6.2), the dependency of the indicator value for the considered types of delay attack can be investigated. To do this, a statistic window, W, should be set, i.e., the amount of samples or offsets that are used for the calculations. With each RI the window moves by one element and allows partly discarding the previous history of the indicator, which can be useful, e.g., for industrial sensor networks with limited resources. The results presented in Fig. 6.1 are obtained for the following values:

$$\begin{array}{ll} j = 1200 & o_{max} = 40\nu s \\ W = 100 & d_{CD} = 2offset_{max} \cdot 150\% \\ \frac{1}{\lambda} = 80\mu s & d_{ID} = 2offset_{max} \cdot 25\% \\ mean_{real} = 80\mu s & d_{max} = 2o_{max} \cdot 300\% \end{array}$$

where λ is the parameter of the exponential distribution for natural delay. Note that the offset is multiplied by two in the calculations, as the offset imposed by the adversary is half of the imposed delay. Fig. 6.1 shows that the mean is affected by adversary influence, but, the values increase only slowly, i.e., it takes several RIs before the attack is visible. In the bigger perspective, the influence of the adversary is clearly visible, but it can be challenging to detect adversary presence before clock synchronization is broken for sufficiently many RIs.

The standard deviation can be analyzed in a similar manner. By using equations (6.1-6.4) we have:

$$\begin{aligned} (k-1) \cdot \sigma_k^2 = \Bigg| \sum_{i=1}^{k} (\overline{\sigma}_{real,i} - o_{real,i})^2 + \\ + \sum_{i=j+1}^{k} \Bigg[(\overline{\sigma}_{attack,i} - o_{attack,i})^2 + 2\overline{\sigma}_{real,i} o_{real,i} \cdot \\ \cdot \left(\frac{\overline{\sigma}_{attack,i}}{o_{real,i}} - \frac{o_{attack,i}}{o_{real,i}} - \frac{\overline{\sigma}_{attack,i}}{\overline{\sigma}_{real,i}} + \frac{o_{attack,i}}{\overline{\sigma}_{real,i}} \right) \Bigg] \Bigg|. \end{aligned} \tag{6.5}$$

The following notation for the component of standard deviation caused by nature is introduced:

$$\sigma_{real,k}^2 = \frac{1}{k-1} \sum_{i=1}^{k} (\overline{\sigma}_{real,i} - o_{real,i})^2. \tag{6.6}$$

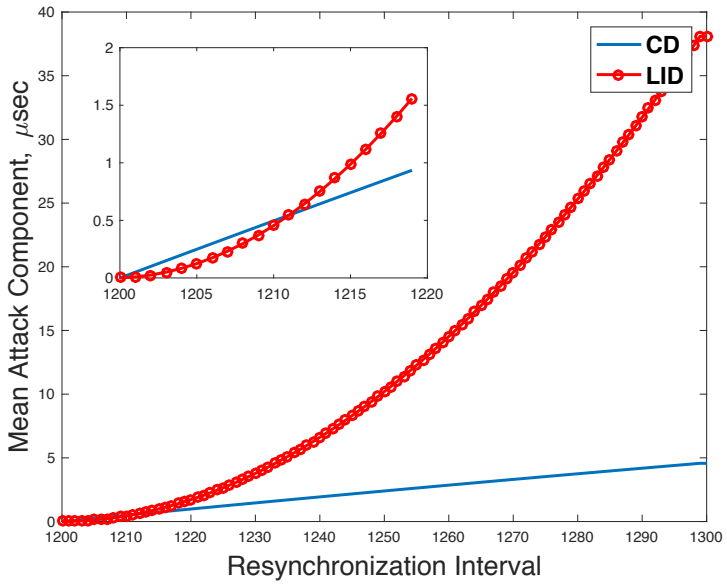

Figure 6.1: Mean values under different types of the delay attack

Then (6.5) can be presented as:

$$(k-1)\cdot\sigma_k^2 = \Bigg|(k-1)\cdot\sigma_{real,k}^2 + \\ + \sum_{i=j+1}^{k}\Bigg[(\overline{\sigma}_{attack,i} - o_{attack,i})^2 + 2\overline{\sigma}_{real,i}o_{real,i}\cdot \\ \cdot\left(\frac{\overline{\sigma}_{attack,i}}{o_{real,i}} - \frac{o_{attack,i}}{o_{real,i}} - \frac{\overline{\sigma}_{attack,i}}{\overline{\sigma}_{real,i}} + \frac{o_{attack,i}}{\overline{\sigma}_{real,i}}\right)\Bigg]\Bigg|. \tag{6.7}$$

In (6.7), the second component is caused by the attack, and therefore the fol-

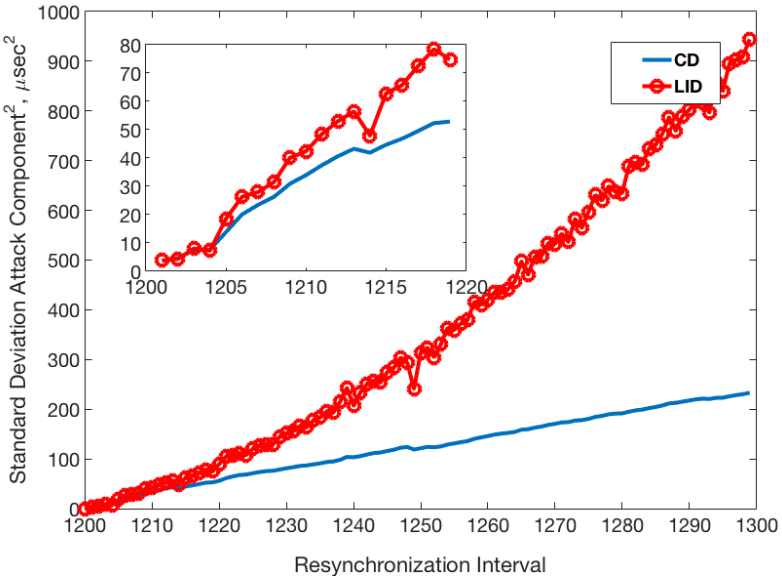

Figure 6.2: Standard deviation values under different types of the delay attack

lowing notation can be introduced:

$$\sigma^2_{attack,k} = \left| \frac{1}{k-1} \sum_{i=j+1}^{k} \left[(\overline{\sigma}_{attack,i} - o_{attack,i})^2 + 2\overline{\sigma}_{real,i} \right. \right. \\ \left. \left. o_{real,i} \left(\frac{\overline{\sigma}_{attack,i}}{o_{real,i}} - \frac{o_{attack,i}}{o_{real,i}} - \frac{\overline{\sigma}_{attack,i}}{\overline{\sigma}_{real,i}} + \frac{o_{attack,i}}{\overline{\sigma}_{real,i}} \right) \right] \right| . \tag{6.8}$$

By calculating σ_{attack} for different types of delays, the indicator value can be compared. The results are presented in Fig. 6.2. It can be noticed that compared to the attack component of the mean, the attack component of the standard deviation is more recognizable from the very beginning of the attack. This demonstrates that standard deviation can have a lower threshold set for making the decision of being under attack.

A relevant parameter that characterizes only the monitor, not the adversary, is the window size, W. It is a sliding window, i.e., each RI the window is moved by one position. The choice of having a sliding window was based on

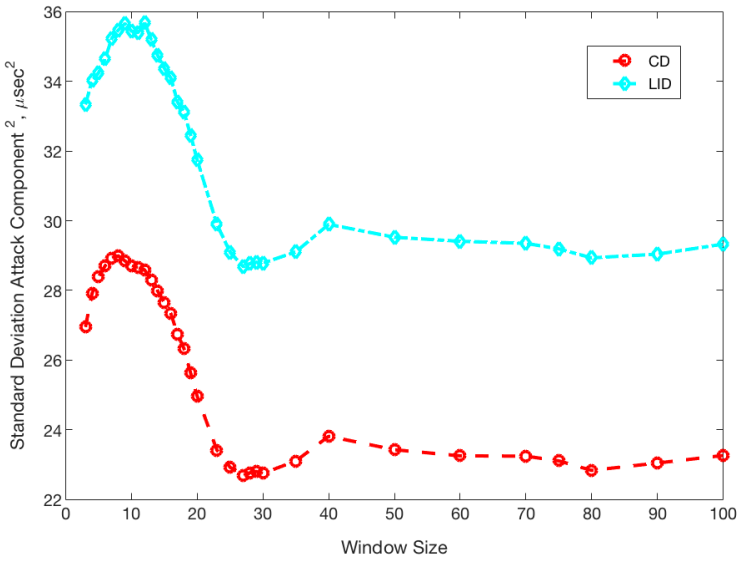

Figure 6.3: Attack component of the standard deviation for different window sizes calculated at the 10th RI

the possibility of restricted recourses for the node. It is interesting to see how the standard deviation depends on the size of the window. Note that this is not the case for the mean, as its attack component does not depend on the real components of the offset. Fig. 6.3 shows the dependency of the attack part of the standard deviation from a window size for a fixed number of RIs and different types of attack. We can see that after some time, the value does not change much, however there are rapid changes in the beginning. This ”shaking” is more visible when the amount of samples without and with an attack component are close to each other. This suggests that checking values of the standard deviation using different window sizes and comparing them can be used as an additional indicator [103]. Further, we can conclude that it makes sense to start checking it only after we have a suspicion that something is wrong, i.e., in QS.

Next, we investigate how detection theory and more precisely the Neyman-Pearson approach can be used as an indicator. Detection theory allows making an optimal choice in the sense of minimizing the probability of error of Type II (i.e., false negative probability), by selecting one out of two possible hy-

Table 6.1: Indicators applicability

Indicator	Applicability
Mean	requires accumulated data
Standard deviation	immediately after an attack was deployed
Neyman-Pearson	requires accumulated data
Window size	requires accumulated data

potheses based on test samples with fixed probability of wrongly rejected main hypothesis, i.e., Type I error (i.e., false positive probability) [134]. To test the hypotheses, the Neyman-Pearson detection lemma can be used [135], defining a threshold for making a decision about which hypothesis is correct based on a set of collected samples:

$$\Lambda(o_{meas}) = \frac{L(o_{meas}|H_0)}{L(o_{meas}|H_1)} \leq threshold, \tag{6.9}$$

where H_0 is the main hypothesis, H_1 is an alternative hypothesis, L is a likelihood function. Hypothesis H_0 claims that there is no malicious delay caused by the adversary and hypothesis H_1 claims that there is a malicious delay:

$$\begin{aligned} H_0 &: o_{attack} = 0 \\ H_1 &: o_{attack} \neq 0. \end{aligned} \tag{6.10}$$

In this case as it was demonstrated in (6.9), we can calculate the threshold for decision making as a ratio of the corresponding likelihood functions.

The overall conclusion from the conducted analysis is that once the attack is deployed, the statistical characteristics of the measured offset change. It means that the considered indicators can be used for making a decision and switching states of the network. Standard deviation is more sensitive to the attack already from the beginning as shown in Tab. 6.1 and therefore, it can be used as a main indicator. Mean in its turn requires some RIs to clearly indicate the attack, and thus it can be used as an additional indicator in QS along with a check based on window size and Neyman-Pearson detection. These three indicators require to obtain some amount of offsets calculated after an attack was deployed to be applied. Thus, in Tab. 6.1 they are charactirized as ones required some data being accumulated after an attack being deployed.

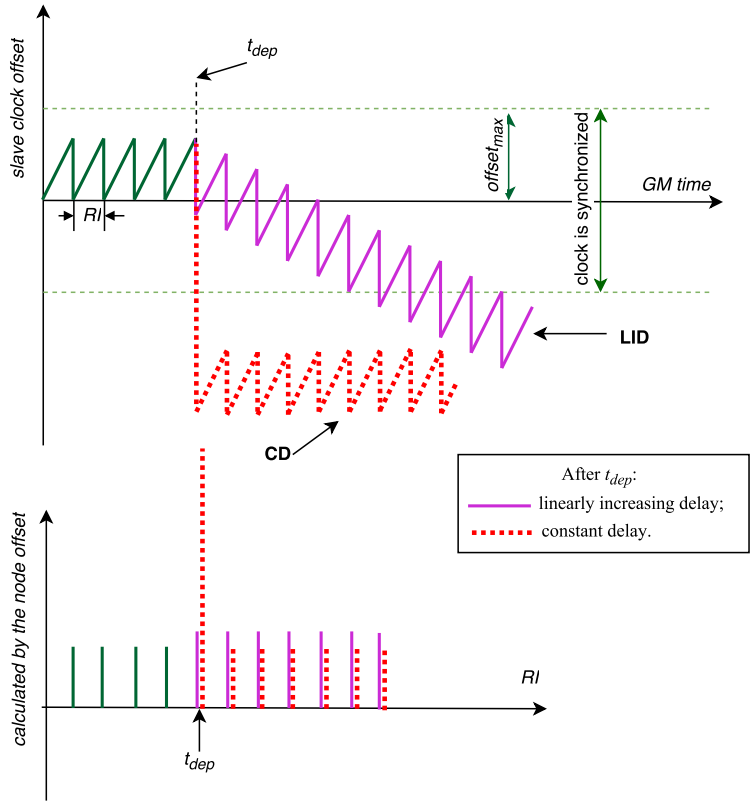

Figure 6.4: The offset correction and offset calculated in a node under LID and CD

6.2 Peer-to-peer Check

The following analysis of indicators is based on the analysis of the attack component of the offset. Thus, the assumption used is that a reference of the real part of the offset is available. This component is needed as the offset calculated according to the clock synchronization protocol is a combination, thus to reason about a new trend in the calculating offsets, a reference from the same time domain is required.

Fig. 6.4 demonstrates the correspondence between the usual presentation of offset in this work (the upper graph) and the offset seen from a node perspective every RI (the lower graph). As it is seen from Fig. 6.4, for the node a LID delay means only one jump in observed values and not a constantly growing trend. For the case of CD, it is again only one jump, though in this case quite significant. Thus, the techniques proposed in Chapter 3 could help with CD, but not with LID. Thus, a time reference is needed to use outcomes of this analysis. A specific way of obtaining the reference is left outside of the thesis scope, as the approach is valid with any such reference. However, two possible ways are presented below.

The first option is to use ”history”, i.e., to keep data from the operational phase and use it as a reference for the average offset observed between a GM and a node. However, this approach does not consider a change in the network topology or a change in environmental conditions causing a change in offset values. The second option is to have a periodical peer-to-peer check between two peers from the same time domain. The check is conducted by obtaining an offset with respect to the other clock that is synchronized to the same GM. It needs to be performed periodically, though not every RI, as the reference value is assumed to be changing slowly. Such a check helps in detecting an anomaly only if the other peer is not suffering from the same cause. To be more robust against a possible influence from adversary, the peer for this check could change now and then, as this way all nodes need to be affected to avoid inconsistency detection. Note that such a check does not require message exchange, as only one message need to be transmitted between peers. The peer-to-peer check introduces a communicational overhead, however depending on the application, information could be incorporated in regularly exchanged information between peers. A combination of both approaches, i.e., having a historical value as a base and updating it according to data obtained from the check, can help to reduce the overhead by increasing the period of such peer-to-peer checks and allow adaptation to changes in the network.

6.3 Tresholding Indicators

For each main indicator we set two thresholds: low and high. We call an indicator positive, when its value is bigger than the corresponding lower threshold, otherwise negative. The rules and their correlation to switching between states were described in Section 5.1. Using the results presented above, we can evaluate how the indicators react for the considered types of attacks. A threshold

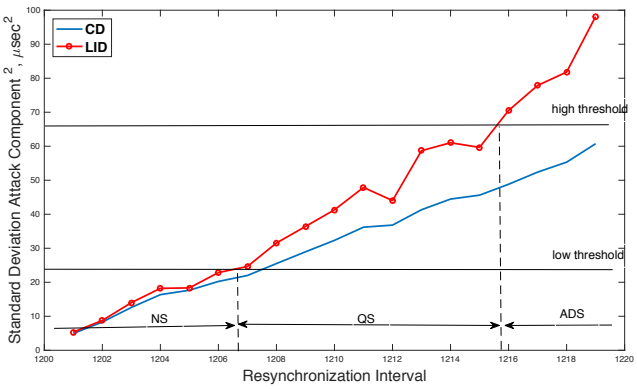

Figure 6.5: Thresholds allocation and switching before network states

example based on standard deviation and LID is presented in Fig. 6.5. The other types of delays can be considered in a similar manner. In the case of a LID attack, the chosen thresholds allow switching into QS after 7 RIs from the moment of an attack being deployed and into ADS after 16 RIs, for this example. For both mean and standard deviation it is possible to set two thresholds depending on the amount of RIs that the monitor can spend for switching to the corresponding states (Tab. 6.3). The threshold values for these indicators as well as the rest depend on the adversary model, i.e., attacks models, and the use case, and thus in Tab. 6.3 the main parameter to reason about the threshold value is indicated.

Window size as an indicator can have one or several thresholds. The value of the threshold depends on obtained historial data of other indicators, thus its threshold can be set based on collected historical data.

A threshold can be derived for the LID attack by considering the Neyman-Pearson lemma under the assumption that two scenarios are possible, namely: H_1 there is a LID attack or H_0 there is no attack, NA. Previously, in Chapter 5 it was showed how the PDF of the offset measured in a node can be calculated for the considered types of delay attacks. The detection coefficient was introduced to indicate the difference visible in the PDF when the network is under attack and to show the level of adversary exposure. For the LID the detection coefficient can be calculated as shown in (5.25). According to (5.16) the

Table 6.2: Delays sets and their probabilities for ID

d_{ID}	p_{ID}
$0.1x$	0.05
$0.5x$	0.30
$1x$	0.50
$3x$	0.15

PDF of the asynchronous delay during the attack is an exponential distribution scaled with the detection coefficient value. The likelihood function for an exponential distribution can be presented as a multiplication of the related PDF for each sample from the window. Therefore based on (5.16) the following likelihood for the described hypothesis can be derived:

$$L(H_{1,ID}) = k_{ID}^{W} \cdot \lambda^{W} \cdot e^{-\lambda sum_{m=1}^{W} o_m}. \tag{6.11}$$

For the hypothesis H_0, the likelihood function is the same as for different variants of hypothesis H_1, but without the detection coefficient. Now, we can calculate the thresholds for the considered scenarios:

$$CD/ID/RD : \Lambda_{CD/ID/RD}(offset) = k_{CD/ID/RD}^{W}. \tag{6.12}$$

To calculate values of the detection coefficient for different types of attacks, we need to define a set of possible steps for LID. Instead of using absolute values, we express the delays for different types of attacks through the variable x, which exact value can be chosen depending on the application. This way the results are comparable and more independent of the actual values of the delays. For example, if $o_{max} = 50\mu s$ and an adversary wants to break clock synchronization after maximum 10 RIs from the point in time when the attack was first deployed, x equals $10\mu s$. The delay set should be complemented with a corresponding set of probabilities of occurrence, which are assigned randomly in Tab. 6.2.

Then, for the window size W the following values of the detection coefficient for different strategies can be calculated for different numbers of RIs as:

$$k_{ID,i}^{\frac{1}{W}} = 5X, 65X, 1227X, 24443X, 490430X, ...$$

The results above show that the considered attacks can be detected using the Neyman-Pearson approach. However, the conclusion is only valid for the

Table 6.3: Indicator thresholding strategies

Indicator	Thresholding strategy
Mean	several thresholds, based the allowed amount of RIs for detection
Standard deviation	several thresholds, based the allowed amount of RIs for detection
Neyman-Pearson	one threshold, based on attack models
Window size	several threholds, based on historical data

values considered. A sensitivity analysis of the threshold and its parameters should be conducted to generalize the outcome. The results were obtained for the window size 10, i.e., under the assumption that ten observed samples are satisfying hypothesis H_1. However, this is not the case when the attack was just deployed. Therefore, the likelihood-ratio test can be used as an indicator for switching between QS and ADS, i.e. as an additional indicator. In this way, at the moment when the additional indicator is used the attack is already ongoing during some RIs and the technique can thus be used more efficiently. For this indicator as summarized in Tab. 6.3, one threshold is sufficient, as only two hypotheses are considered, and the value of the thresholds is obtained based on the detection coefficient, i.e., an attack model that includes probabilities for a chosen set of delays.

The conducted analysis allows to set thresholds depending on application requirements, i.e., how many RIs a node can be in an unsynchronized state. This number depends on how active the considered node is. For example, if it sends out status data once per 20 RIs, it can tolerate (i.e., does not propagate the adversary influence further itself) being in unsynchronized state for up 10 intervals on average. Setting these thresholds is a tradeoff between estimation of possible harm from the node being in unsynchronized state and cost of bringing the network into a safe state (as a result of switching into ADS) in case of a false alarm.

6.3.1 Time Chase

In this subsection the interaction between an adversary and the monitor is considered from a time chase point of view, i.e., what happens first: the monitor detects the adversary influence on clock synchronization or the adversary accomplishes its target of putting a node into an unsynchronized state for at least

a specified amount of RIs? The assumption is that the adversary goal is breaking clock synchronization and keeping it for at least t_{tar}.

The relation between the speed of a delay attack and the speed of an attack discovery influence the attack propagation in the network and thus defines the attack efficiency. The attack influence on the network depends on how the error propagates through the topology and the particular use case. In this analysis the time the adversary needs to put the node into unsynchronized mode is considered and the further propagation of its influence is left outside of the scope.

Each RI, the clock is corrected according to a calculated offset, so that the clock time stays within the allowed boundaries. The monitor needs $t_{NS \to ADS}$ after the attack was deployed to switch the network into QS and $t_{QS \to ADS}$ after it was switched to QS to switch the network into ADS. The alternative scenario is when the monitor switches the network from NS directly into ADS, $t_{NS \to ADS}$. The question is who is faster – the adversary achieving its goal or the monitor switching into ADS:

$$min\{t_{NS \to ADS}, t_{NS \to QS} + t_{QS \to ADS}\} \underset{\text{Monitor wins}}{\overset{\text{Adversary wins}}{\gtrless}} (t_{break} + t_{tar}). \quad (6.13)$$

Even though the precise formulation of the problem can vary for different types of delay attacks, the main question is the same, whether the adversary can succeed or the monitor can prevent it. In the LID case, the adversary needs some RIs to bring the clock into unsynchronized state. However, the LID mode may be appealing for an adversary as it implies that a trend in the indicators behavior appears before clock synchronization is actually broken and, therefore it can be challenging to detect. However, for the considered example, it is possible to see that this type of attack does not bring benefits from an indicators behavior point of view. For CD mode, the synchronization is broken from the moment the attack is deployed, i.e., $t_{break} = 0$. This is an advantage for the adversary, as the monitor needs to react faster compared to the case when the attack requires a preparation phase.

One way to cope with a delay attack is to introduce Relaxed Mode (Sec. 3.3.3), i.e., a mode when the clock is temporary allowed to exceed the boundaries. Such a mode brings degraded quality of synchronization, however it gives an advantage in the time chase when $t_{break} \neq 0$. It can also be beneficial even if $t_{break} = 0$, in case the adversary does not have knowledge about the boundaries in the Relaxed Mode. There are different proposals for IEEE 1588 extensions [136]. However, as the considered way to break synchronization is based on introducing a selected asymmetrical delay, many proposed fault-tolerance

techniques, such as on-the-fly time-stamping or grouping masters to a master group to detect a failure of one of them, cannot cope with this type of attack. The attack is challenging to counter react, as it does not require message modification. One of the feasible ways to detect such an attack can be a multipath strategy, however in such case an adversary just needs to take over several communication channels instead of one to succeed.

The more critical the application is the more desirable it is to make t_{QS} smaller (besides the overall goal to decrease t_{QS+ADS}) as the cost of a false positive can be tolerated. The earlier the network is switched into QS, the earlier additional monitoring techniques can be deployed, and the earlier we can start to decrease the level of trust for the current GM. However, it also implies a bigger false positive probability. To find the balance, risk assessment should be conducted to evaluate possible harm and cost of preventing techniques [137].

One more factor to consider when shaping the monitor strategy comes from risk assessment. In the security area risk assessment can be built upon ATA, a technique that helps to identify ways of an adversary to achieve its target and calculate the probability along with the cost of an attack [138]. The monitor strategy can be characterized by considering the risks that exist in the system. Risks imposed by a decision can be calculated as a multiplication of severity of consequences, s, i.e., potential loss, with a likelihood of event occurrence, p:

$$risk = s \times p \tag{6.14}$$

For example, let us consider switching modes $NS \rightarrow QS$ from the monitor perspective. There is a risk to make a switch when there is no actual attack. The consequences are costs of taking additional precautions, e.g., checking additional indicators, tolerating reduced quality of clock synchronization introduced by the Relaxed Mode, and costs of eventual switching of the system into ADS. The related probability is called false positive probability. There is also a risk to fail to detect a trend in the offset values. The consequences are the cost of the adversary achieving its target, i.e., disrupting the network, and the probability is termed false negative. The combination of these factors shapes the strategy of the monitor, i.e., how reactive it should be.

To reason about acceptable risks, the ALARP principle can be used [50]. The approach defines different risk zones that are connected to the probability of occurrence of related hazardous events. One such zone is an ALARP zone, in which the risk should be reduced to as low as reasonable practicable. The challenge is to define this level for a concrete system. The basis of such reasoning is a particular application, i.e., a use case. If it is a safety-critical system,

it is much more reasonable to invest more in its protection to reduce possible risks.

The time chase between an adversary and the monitor is defined by the adversary strategy to achieve its target and by the monitor strategy to counter-react. The only factor we can influence directly is the monitor strategy, and we can only reason and make assumptions about the adversary strategy based on possible gains and cost investments. Here (6.13-6.14) serve as useful grounds for deciding how proactive the monitor should be.

6.4 Sensitivity Analysis of Indicators

This section presents a sensitivity analysis of the indcators considered above. The outcome of the analysis allows to generalize the applicability of different indicators and identify limitations of the approach, i.e., providing grounds for further monitor enhancements.

6.4.1 Use Case Parameters and Indicators

To identify dependencies between thresholds and use case, we need parameters that are relevant for both. The following parameters are considered:

- *Generic Boundaries: o_{max}, and RI duration*. These two parameters are dictated by an assumed upper limit of the clock drift and node criticality. The first one determines how often the correction should be done so that the clock has the required precision, which in turn is determined by the criticality of the use case, the particular node load and functionality.

- *Window Size: W*. Calculations of offset values are done based on a set of historic values contained within a window. The window is sliding, i.e, in each RI, a new value is added and the oldest one is discarded.

- *Offset Mean: $\overline{\sigma}_{real}$*. This parameter is defined by the network topology and can be estimated if the offset history is available. Thresholds derived from it, needs to be reevaluated after each new network reconfiguration.

- *Nature Distribution Parameter: λ*. We assume that delays occurring due to natural disturbances in the network follows an exponential distribution [133], with the parameter λ.

Several indicators are suitable for clock synchronization monitoring. A node calculates an offset each RI, therefore, its mean and standard deviation can be monitored in order to identify a new trend in these values caused by an attack. Next, the window used to calculate statistics is an indicator too, as it can help to get additional assurance regarding the presence of a new trend, by scaling down the range of samples used for calculation. Finally, if the trend detection problem is formulated using detection theory, the Neyman-Pearson criterion provides one more indicator to consider.

Note that the offset considered in an analysis is the real offset, i.e., the value represented by the distance between a line and the x axis in Fig. 5.2, while an offset that is calculated as a result of the protocol is different (this offset is demonstrated by the vertical parts of time dependency in Fig. 5.2, i.e, the jump performed each RI). Therefore, an analysis based only on calculated values is limited to a prediction and reasoning of an adversary behavior. To use an outcome of the analysis for attack detection, a reference about a possible real offset is needed, e.g., via a peer-to-peer check of clocks within the same time domain. However, under the assumption of this check taking place, the analysis of indicators remains the same and the sensitivity analysis is still valid.

6.4.2 A Sensitivity Analysis

This section provides a brief description of the indicators followed by an analysis demonstrating their dependencies with respect to the adversary model.

Mean and Standard Deviation

Statistical characteristics of the true offset, that can be obtained from a measured one if there is a drift reference, are the mean and standard deviation. These can be used as indicators and calculated using a window *W*, as presented below:

$$\overline{\sigma} = \frac{1}{W} \sum_{i=1}^{W} o_{meas,i}, \tag{6.15}$$

$$\sigma = \sqrt{\frac{\sum_{i=1}^{W} (o_{meas,i} - \overline{\sigma})^2}{W-1}}. \tag{6.16}$$

According to the approach described in 5.1, there are two thresholds for every indicator, one lower and one higher. For the mean and standard deviation, the method to set these thresholds is to investigate their attack component, i.e., the component of an indicator value caused by the delay imposed by an adversary,

and based on use case characteristics define at which RI the attack needs to be detected after being deployed. To set thresholds, the part of the indicators that are caused by an attack component of the offset can be separated. For simplicity, it is assumed that the attack component of the offset before the attack is deployed, is equal to zero, thus (6.4, 6.8) could be presented as:

$$\overline{\sigma}_{attack} = \frac{1}{W}\sum_{i=1}^{W} o_{attack,i}, \tag{6.17}$$

$$\sigma^2_{attack} = \Big|\frac{1}{W-1}\sum_{i=1}^{W}\Big[(\overline{\sigma}_{attack,i} - o_{attack,i})^2 + \\ +2\overline{\sigma}_{real,i}\cdot\cdot o_{real,i}\Big(\frac{\overline{\sigma}_{attack,i}}{offset_{real,i}} - \frac{o_{attack,i}}{o_{real,i}} - \\ -\frac{\overline{\sigma}_{attack,i}}{\overline{\sigma}_{real,i}} + \frac{o_{attack,i}}{\overline{\sigma}_{real,i}}\Big)\Big]\Big|. \tag{6.18}$$

(6.17–6.18) show that the attack components of both indicators are proportional to the value of the delay, as $\mathit{offset}_{attack,i}$ is proportional to the delay imposed at the i-th RI. Therefore, the choice of the range of delays, i.e., its minimum and maximum values, affects the outcome of thresholding. The range of delays should be dictated by the use case specification, e.g., for CD, the range should be starting from a value close to the minimum one required to break clock synchronization, i.e., $4\cdot$ offset_{max}, and should be going up till when a further increase does not bring any significant change. For instance, if the network has time-slot based channel access, a node being in an unsynchronized state will start accessing the wrong slot. Once this happens, a further increase of the value of the offset does not matter as such. The attack components of both indicators also depend on the window size, however, this dependency is exploited by considering the window size as an indicator.

The attack component of the standard deviation depends on $\overline{\sigma}_{real}$. Therefore each time a route of the messages is reconfigured, the thresholds have to be recalculated. Moreover, as $\overline{\sigma}_{real}$ depends on λ, the thresholds for the standard deviation should be recalculated in case of using the system in a drastically different environment. Tab. 6.4 summarizes identified dependencies: the attack component of mean depends on attack model, i.e., the value and the range of the delay; the attack component of standard deviation depends on the attack model, the communication link, i.e., $\overline{\sigma}_{real}$, and on the environment, i.e., λ. The parameter of nature distribution, λ, also indicates a dependency on a network setting, however as those details are not considered, the parameter is connected to the environment.

Neyman-Pearson Criteria and Detection Coefficient

The problem of attack detection, i.e., a detection of an offset attack component, can be formulated though detection theory and in particular the Neyman-Pearson lemma. The threshold obtained based on the Neyman-Pearson criterion we call Neyman-Pearson threshold. The Neyman-Pearson threshold, T_{NP}, can be calculated via a detection coefficient for a particular type of delay attack based on an adversary model, e.g., for LID it can be calculated as follows:

$$T_{NP} = \Big(\sum_{j=1}^{N_{LID}} (e^{\lambda} \cdot i \cdot d_{LID,j} \cdot p_{LID,j}) \Big)^{W}. \tag{6.19}$$

(6.19) is transformed further in a form allowing to determine how the threshold depends on the size of the considered delay space, i.e., N_{LID}. For simplification, we assume that the probability of choosing one of the delays from the set is uniform, i.e, that $p_{LID,j} = \frac{1}{N_{LID}}$, as there are no obvious reasons to prefer a specific one. To assess the difference between situations when the two various sets of delays are considered, we compare the thresholds for the two following sets: (*i*) Set 1 with delays of size N; (*ii*) Set 2 with delays of size $N+1$ containing all delays from the previous set and one more value, $d_{LID,N+1}$. A comparison of thresholds for these two cases boils down to the following inequality:

$$(1 + \frac{1}{N}) \cdot (e^{d_{LID,1} - d_{LID,N+1}} + \ldots + e^{d_{LID,N} - d_{LID,N+1}}) \lesseqgtr 1. \tag{6.20}$$

If eq.(6.20) is solved with *"less"*, then adding $d_{LID,N+1}$ into consideration lowered the threshold, whereas if it is solved with *"greater"*, then oppositely the threshold was increased. (6.20) cannot be unambiguously solved in a general way, however we can point out several cases when it is possible. If $d_{LID,N+1}$ is greater than all delays from the Set 1, (6.20) is solved with *"greater"*. If $d_{LID,N+1}$ is less than all delays from Set 1, (6.20) is solved with *"less"* for a large enough N. In other words, adding a delay that is above the maximum one from the already considered set, will increase the threshold; adding a delay that is below the minimum one from the already considered set, will lower the threshold. The outcome of the intermediate cases depends on N and the particular value of the delay. However, a similar trend to the one described above for mean and standard deviation can be observed, namely that when choosing a set of delays to consider, it is important to define a valid range of delays. This threshold dependency from the natural distribution is quite obvious, as the greater λ is, the higher the threshold is, which correlates with

Table 6.4: Indicator attack components sensitivity

Indicator	Attack model	Communication link	Environment
Mean	✓	X	X
Standard deviation	✓	✓	✓
Neyman-Pearson	✓	X	✓
Window size	corresponds with sensitivity of the considered for "zum in" indicator		

the fact that the more network disturbance, the higher the malicious deviation needs to be in order to be detected.

The case of CD can be analyzed in a similar fashion, as its detection coefficient is similar. Therefore, we only consider the RD case below. The threshold for the RD case can be calculated based on a detection coefficient for RD:

$$T_{NP} = \Big(\frac{e^{d_{RD}^{min}}}{(d_{RD}^{max} - d_{RD}^{min})} \big(e^{\lambda \cdot (d_{RD}^{max} - d_{RD}^{min}} - 1 \big) \Big)^{W}. \tag{6.21}$$

If the range of RD is fixed, i.e., $(d_{RD}^{max} - d_{RD}^{min}) = const_1$, then the greater the minimum value, the higher the threshold. If the minimum delay is fixed, i.e., $d_{RD}^{min} = const_2$, then the threshold's dependency from the range of delays has the same character as $f(x) = \frac{e^x - 1}{x}$, where $x > 0$. For the defined range of x values, this function is constantly growing and has a limit equal to 1 at $x = 0$. Therefore, for this case we have a minimum threshold that equals $e^{d_{RD}^{min} \cdot W \cdot \lambda}$. The greater the range of delays, the higher the threshold, and the same goes for the dependency of λ. Thus, the Neyman-Pearson indicator as it shown in Tab. 6.4, depends on attack model, i.e., the range of delays, and the enviroment, i.e., λ.

Window Size

Having the window size as an indicator was initially proposed as an additional indicator deployed only in QS. This indicator is used in addition to the ones considered above, as when varying the window size, the monitor still checks the mean, standard deviation or the Neyman-Pearson criterion. Therefore, all dependencies identified for the other indicators are relevant for the window size as well.

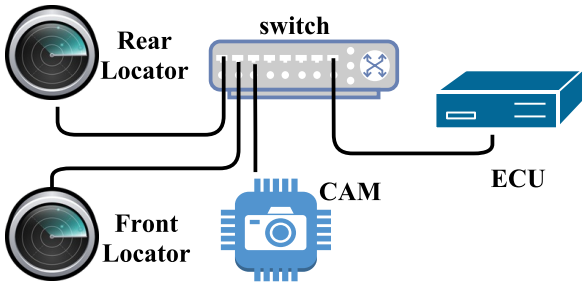

Figure 6.6: A Collision Avoidance System

6.4.3 A Use Case Specific Sensitivity Analysis

An example of an CPS is considered to illustrate thresholds dependencies for a particular use case.

Use case — a Collision Avoidance System

A Collision Avoidance System based on a time-triggered architecture is an example of a CPS from the automotive domain and is used to investigate the dependencies of the indicators with respect to a particular use case. As it is shown in Fig. 6.6, the system consists of (*i*) a switch managing connections and following a schedule, (*ii*) a camera, CAM, located in the front part of the vehicle, (*iii*) two locators - one in the front, and one in the rear part of the vehicle, and (*iv*) an Electronic Control Unit (ECU). The camera sends pictures of the road in front of the vehicle to the ECU, so that the latter can estimate the situation. The ECU also gets information from the locators, that estimate the distance from the vehicle to the nearest object in a certain direction. Locators send data only if an object is detected within a specified range. The camera in its turn sends data constantly within specified time-slots. Once the ECU makes a decision that the current situation is potentially dangerous, it sends a signal to the outer dependency of this system, e.g., a steering control system. The system has a time-triggered architecture, i.e., channel access is time-based and there is a schedule for data transactions. Nodes in the system are using the IEEE 1588 standard for clock synchronization. We assume that a time slot duration is 100 μs [139], RI is 50 ms and that a clock drift is 10^{-2} s/s, i.e., up to 0.1 μs deviation every 1 ms. Therefore, the synchronization boundaries, o_{max}, can be set with a 2 μs safety gap and be equal to 7 μs.

Use Case Dependencies

The particular use case along with the choice of communication structure, set the boundaries and the RI duration. These values define which delay values can be used to break clock synchronization, e.g., the minimum delay for CD and RD or a breaking time for the LID attack. In the considered example, the minimum delay value for CD is greater than $4 \cdot o_{max}$, i.e., 28 μs. The maximum delay that makes sense to consider is defined by the time-slot duration. From an adversary perspective, once the offset exceeds the time-slot duration, there is no gain to exceed further. The node functionality sets the time that the node should be in an unsynchronized state for an adversary to succeed, i.e., at least t_{tar}. In the considered example, the camera sends information with a high frequency, as a situation on the road can change very fast considering the vehicle speed. The assumption that it is critical to miss any message from the camera or locators. Therefore, considering the assumption stated above to have significant consequences, it is enough to miss one time-slot during a vehicle operational phase. We can assume that the camera messages are scheduled every other time-slot, i.e., that camera occupies 50 per cent of the slots, thus an adversary needs to keep it at least 100 μs to certainly break its functional requirement implying that a message was missed . Locators are using on demand time-slots that is not occupied by the camera, only if there is an object detected. Consequently, if there is no detected object, i.e., in case of driving during late night on a highway, a locator in an unsynchronized state would not be noticed by the system. Therefore, for locators, t_{tar} depends on the operational situation, but it is on average longer than for the camera.

6.4.4 Discussion

The conducted analysis regarding an adversary model, shows that it is a necessity to determine valid limits for the values of the imposed delays. This fact implies a necessity to know a particular use case in order to specify an adversary model. A threat model includes an adversary model as well as a use case specification. However, to calculate thresholds for when the monitor should switch states, a deeper level of use case specification is required, i.e., the network topology should be specified together with the node functionalities. In the best case, a schedule is provided such that the node communication pattern and frequency of sending and receiving messages can be estimated. Moreover, as it was demonstrated in Sec. 6.4.2, the standard deviation attack component depends on the offset mean, which in turn depends on the route

Table 6.5: Indicators classification

Indicator	Main, NS	Additional, QS
Mean	✓	X
Standard deviation	X	✓
Neyman-Pearson	✓	✓
Window size	X	✓

between a GM and a slave used for the PTP protocol message exchange. This limits the flexibility of the method, although it is reasonable to expect a security solution being tightly use case dependent. A way to improve the flexibility is to automate the calculation of thresholds or cardinally relax this connection by imposing requirements on the degree of indicator inter-dependency in the process of indicator development.

Another indirect outcome of the analysis is that it provides grounds for an indicator generalization. By analyzing a specific use case, we can identify which indicators are relevant and which of them can be used efficiently as main or as additional indicators. As identified in Sec. 6.4.2, indicator thresholding is heavily connected to the use case and the adversary model. However, the indicator applicability relates not to a particular value of a threshold, but to a trend in indicator behavior, which is identified by its general analysis in this chapter. Therefore, we can conclude that the results on indicator applicability can be generalized, but not their effectiveness, which should be investigated for each particular use case. For instance, as standard deviation thresholds require knowledge about the communication link between a GM and a slave, it can be used as an additional indicator, since its threshold calculation can be done only when entering the QS and thus a network reconfiguration during the NS would not require its recalculation. The mean in turn requires less amount of information about the specific use case and adversary model, and therefore it is a good candidate for being a main indicator. Finally, the Neyman-Pearson criterion is in between mean and standard deviation regarding required input, as it does depend on the adversary model, but does not require data about the particular communication link. Thus, this indicator can be used in both states. Tab. 6.5 summarizes whether the considered indicators can be used as main, additional or as both.

The monitor analyzed in this work is a type of IDS [86], which can be used for a general application [77]. Even though the proposed monitor use case is narrowed down to clock synchronization handling here, it is possible to extend

it to cover other assets of CPSs following a similar approach, e.g., to develop new relevant indicators or to re-evaluate the applicability of already developed ones. The considered indicators and the different ways to threshold them are based on simple equations, and therefore we analyzed them using standard mathematical approaches and do not formalize the analysis further [140]. Risk evaluation and cost assessment of the monitor [141] are outside the scope of this work, however it can be a logical step in further investigations of its development.

6.5 Answering the Question

In this chapter, a distributed monitoring strategy to detect if an adversary is interfering with the clock synchronization protocol is proposed. The monitor uses certain indicators and a set of rules to decide about switching between normal, quarantine and anomaly detected states. A way to set thresholds for switching between these states based on application requirements and time analysis of indicator performance was proposed. Also an attack exposure problem using detection theory was formulated as well as an additional indicator that can be used in quarantine state: window size. To analyze the results, a time chase between an adversary and the monitor was considered. The results show that the mean values of the offset, as an indicator, reflects the trend change only in the long run and, therefore, can be used for applications that can tolerate a broken synchronization for several resynchronization intervals. Standard deviation in turn, detects changes in the trend faster, i.e., in fewer RIs and, therefore, can be used as an indicator for more critical applications. For the considered system model, an additional indicator was proposed, derived from detection theory, which is able to identify the considered types of attacks. However, a more complex hypothesis, that combines different types of attacks and detects influence on the offset more generally, needs to be considered to be able to generalize the outcome.

Next, a sensitivity analysis was conducted, focusing on indicators and thresholds used by the monitor for switching states when it detects that clock synchronization is under attack. The analysis shows a strong dependency on some particular values of thresholds derived by the use case and the adversary model. For the Standard Deviation indicator, thresholds for its attack component also depend on the network topology, implying that this indicator is best used as an additional one, as then its threshold recalculation is not needed during NS in case of network reconfiguration. Thresholding the Mean indicator requires

less details about the network, and therefore it can be used as a main indicator. Neyman-Pearson thresholding has a higher level of dependency on the use case compared to mean but less compared to standard deviation, therefore it can be used in both roles and assigned depending on other factors that are outside the scope of this work, e.g., a risk assessment.

Chapter 7

Can the Approach Be Applied in Different Settings?

In this chapter the monitoring approach, i.e., the systematic way to design the monitor, is considered when applied to a different scenario, namely for channel reliability estimation on an example of Cooperative CPSs (CO-CPSs). Having a monitor in the network of systems has several benefits; it can support the level of safety in the system, e.g., by detecting component failures, and/or the level of security in the system, e.g., by detecting an attack being deployed. A monitor used for safety reasons detects system states leading to possible hazards, with the goal to prevent or at least mitigate the consequences. A monitor used for security purposes detects malicious actions in the network with the goal to suppress adversary actions and their consequences. Thus, the actions required from a monitor to cover both domains are similar. In this chapter the developed concept of monitoring is tested for its applicability in a bigger scope, the aim to detect clock synchronization being broken is replaced with detecting a transient failure caused by the communication channel with a safety or security related implication and thus provide an estimation of the channel reliability. The following sections provide the motivation for chosen settings to apply the approach and also present the approach refined for the case of reliability estimation of a communicaiton channel in CO-CPSs.

7.1 Motivation of Feasibility Study Settings

Safety assurance remains one of the main challenges for CO-CPSs as wireless communication, along with all its advantages, imposes new challenges in regards to safety. Broadening the system boundary from a single system such as a car to a cooperative system such as, e.g., a platoon with an unknown number of cars raises new challenges in failure logic analysis of the cooperative systems. Besides local failures, the analyses need to take into account also failures in remote systems, as well as in the communication between the systems [142]. Furthermore, the networks in safety-critical systems were traditionally kept closed and isolated suppressing their security concerns in that way. However, the current trend of interconnecting systems by using wireless communication imposes a challenge towards security, as openness of wireless channels can let an adversary receive transmitted messages or interfere with the channel. Security threats can jeopardize reliability and thus affect the safety of the system.

The notion of black channel has been proposed in IEC 61508 [10] as a way of handling the inherent unreliability of wireless links. The black channel concept implies that communication properties cannot be guaranteed and the challenge of reliable communication should be handled by the CO-CPSs. Thus, a methodology for design of a black channel state manager to handle the assessment of communication reliability across CO-CPS is considered. The channel state manager determines the current state of the channel and allows the CO-CPS to react upon this information accordingly. To support analyses of anomalous behaviours across cooperating systems, a common understanding of the anomalies is needed. Hence, such black channel state managers for CO-CPSs need to be developed as reusable components, but IEC 61508 does not detail their development or assurance. The automotive functional safety standard ISO 26262 [55] proposes SEooC as a reuse concept, which can be applicable for the CO-CPS black channel state manager. The SEooC concept enables design of an element outside of the context of a specific system, but upon assumptions about safety relevant properties that need to be validated later during integration of the element. In order to support the high integrity demands such reusable elements may be required to comply with, SEooC development and assurance process has been extended with semi-formal assumption/guarantee contract methodology [56]. The CO-CPS black channel state manager for communication anomaly detection is proposed to be developed aligned with contract-based SEooC guidelines.

The main advantage of developing an element out of context is its future

reusability. If developed as a SEooC, the channel state manager can be reused by different applications requiring cooperative systems (e.g., car platoons or groups of autonomous vehicles working for smart manufacturing) as well as separate elements of these systems of systems (e.g., separate cars in the platoon or vehicles on a factory floor). Taking car platooning as an example, communication which can be described as connections over a black channel is set up between the platoon leader and other cars in the platoon. Communication with the leader vehicle is a crucial requirement at all times, e.g. when joining, driving within or leaving the platoon. As the cars within the platoon drive close to each other and have to cooperate, reliability of the link to the leader is essential and once a car's channel state manager defines that the channel is not reliable anymore, the decision about the car disengaging might be made. The decision about the possible system response could be made based on the channel reliability level that is estimated by the channel state manager. Details of the concrete system reactions are outside of the scope of this chapter, however it is envisaged that the channel state reported by the channel state manager can be used as an input for a system state machine [142]. Ensuring the same understanding of the communication with the platoon leader across the platoon is important for the safety of the platoon. For example, if two vehicles have different conditions for detecting loss of communication to the leader, they may react differently to the same situation, which may be hazardous in some cases. Note common perception of communication is not awareness of other vehicles' current status and actions, but the same logic applied to estimate channel reliability. Thus, introduction of a channel state manager, which is present in every vehicle and responsible for channel state estimations done by all cars within the platoon, might significantly improve safety and efficiency of platooning.

7.2 The CO-CPS Channel State Manager Design

In this section the methodology rationale for designing a black channel state manager as a part of a CO-CPS, which is responsible for detection of communication anomalies and consecutive channel state estimation, is presented [143].

This chapter is focused on considering the design of the manager itself, while the design of the CO-CPS is abstracted away only to the manager and the rest of the system that uses the input from the manager. Fig. 7.1 presents a reference architecture of such a CO-CPS, where the state manager consists of a monitor and an analyzer. The monitor gets input data and checks the defined set of indicators. Next, the analyzer, based on the indicators values, makes a

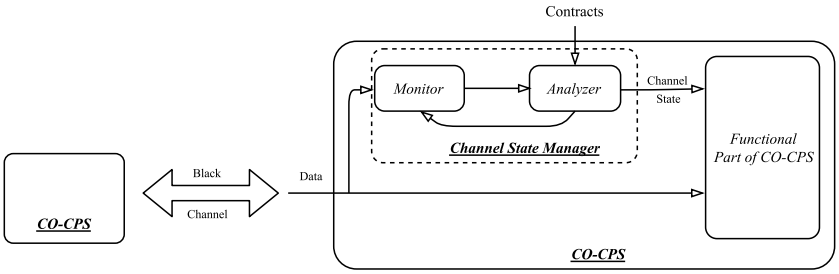

Figure 7.1: The CO-CPS reference architecture

decision about the current reliability level for the communication channel, i.e., the current channel state. The analyzer operates based on the contracts derived for the channel state manager and provides as an output the current channel state.

The design phase of the CO-CPS channel state manager development is split into six steps, from 0 to 5, depicted in Fig. 7.2. Note that while the depicted process may seem sequential, its steps are iterative in nature, where going several steps back may be needed in some cases. In general, after the design, the next phases are the development on hardware and software levels, which are also performed out-of-context. Only the following up integration and maintenance should be performed in-context. However, the focus of the current work is on the design stage and thus the phases of hardware and software development and further integration are not considered.

The design process starts with Step 0, where necessary assumptions are formulated. To make the development of a SEooC possible, the real context is replaced with assumptions that shall be validated later during the integration phase of the element. Therefore, Step 0 includes formulation of the assumptions about the system including the environment it operates in, its external dependencies, particular specifics required for the system along with assumptions about the SEooC and its environment. These assumptions are used as an input for safety and security analyses in Step 1. Based on these assumptions, the assumed hazards for CO-CPS are formulated within Step 1. The purpose of the safety analysis is to identify all the failure contributions of the channel state manager and the unreliable channel being monitored that can contribute to the potential CO-CPS hazards in the assumed context. Similarly, the purpose of the security analysis is to identify all threats enabled by the channel

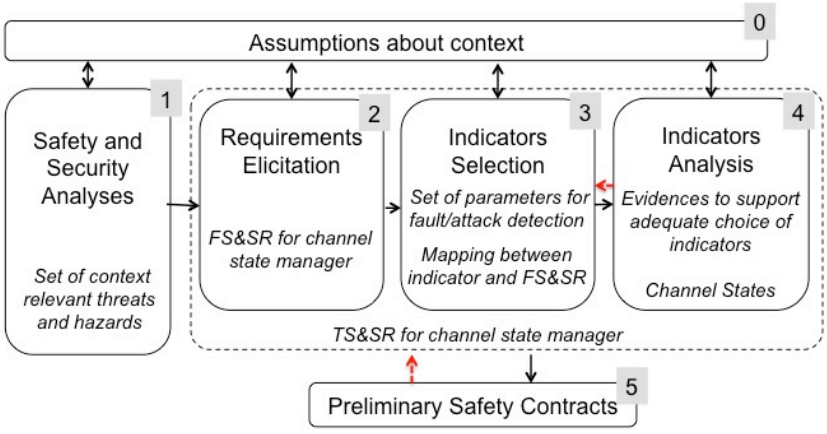

Figure 7.2: Steps of the CO-CPS channel state manager design methodology

state manager and the unreliable channel that can contribute to potential CO-CPS hazards. There are two sets of threats and failures which are produced as an outcome of Step 1. The first set is a set of threats coming from or failures caused by the channel state manager itself that may potentially lead to the assumed CO-CPS hazards. This set is an outcome of analyses performed for the channel state manager in the assumed context of CO-CPS. The second set consists of assumed failures caused by and assumed threats coming from the communication channel and potentially leading to CO-CPS hazards. The second set is derived based on assumptions from Step 0 and the assumed CO-CPS hazards derived also within Step 1.

In Step 2, functional safety and security requirements (FS&SR) for the channel state manager are elicited as an outcome of the analyses in the previous step. As both safety and security are considered at the same time, possible conflicts between requirements coming from both properties need to be resolved in this step. Note that the requirements formulated at Step 2 are valid only for the channel state manager and not for the CO-CPS as a whole, thus, implicitly the channel state manager's functionalities are defined. For example, a related requirement can be formulated as *the channel state manager shall be continuously assessing the quality of the communication link to each individual CO-CPS*.

Step 3 includes selection of the indicators to monitor. Given (*i*) FS&SR,

(*ii*) assumptions about CO-CPS hazards, (*iii*) the set of threats and failures that could contribute to the assumed CO-CPS hazards and which originated from the communication channel, possible indicators are identified based on expert knowledge and corresponding databases. These indicators are chosen so that their monitoring allows detection of faults and attacks which originated in the channel and might lead to assumed CO-CPS hazards. Thus, the indicators are mapped with FS&SR. For example, bit and packet error rates or timing offsets might be needed to monitor to assess the channel reliability to comply with the requirement mentioned above.

In Step 4 the selected indicators are analyzed. Their sensitivity regarding identified threats and failures (both may lead to CO-CPS hazards and coming from or caused by the communication channels) needs to be examined to select the ones that significantly vary in presence of one of these failures or an attack realizing one of these threats. The analysis should show how the indicators respond to these attacks or failures, i.e., how they can be bounded by thresholds. The outcome of Step 4 is a collection of evidences demonstrating adequacy of indicators selection. Based on the analysis, the needed amount of channel states can be reasoned about, i.e., how many situations can be efficiently distinguished based on indicators' values.

Steps 3 and 4 are iterative, so if the chosen indicators are not indicative, others need to be found and analyzed. The overall outcome of Steps 2-4 is elicitation of technical safety and security requirements (TS&SR) for the channel state manager. Technical requirements are mapped with functional requirements from Step 2 and contain thresholds for selected indicators needed to make a decision about an attack or a fault. For example, the quality of a communication link can be determined as degraded once packet error rate exceeds 10%.

In the final step of the design phase of the CO-CPS channel state manager the identified assumptions, the indicators and their thresholds are captured in the corresponding assumption/guarantee contracts. The process of safety contracts development for a SEooC can be divided into several phases [56], however only elicitation of the preliminary safety contracts that correspond to the SEooC assumptions is included in the design phase. These contracts indicate the guaranteed state of the channel under the specified assumptions. Additional contracts indicate the assumptions on the context of the CO-CPS, which are needed for indicator monitoring and estimation.

7.3 Methodology Application Guidelines

In this section the proposed steps of the channel state manager design are discussed and examples on how they can be followed are presented.

7.3.1 Step 0: Assumptions

The methodology described above is proposed for a generic black channel state manager and thus suits a variety of use cases involving CO-CPSs. However, when applying the methodology for development of a concrete channel state manager targeting a set of applications, these applications will dictate the assumptions which are needed to perform safety and security analyses, define functionalities of the manager and requirements to it, and find suitable indicators for monitoring.

Since in cooperative and safety-critical systems, packets transmitted over a channel typically have to be delivered in a real-time manner, i.e., before corresponding deadlines, notion of time has to be present in every cooperative device. Thus one assumption that can be made is the need for time synchronization between the communicating nodes. Also, assumptions can be made about the data sending patterns as nodes can be sending data in time-triggered (e.g., periodic updates by sensors in monitoring systems) or event-triggered (e.g. packets triggered on a particular value of a sensor measurement) manner. When sending packets in a time-triggered manner nodes need to know when the previous packet has been sent and when it is time to send a new one. Regardless of the packet sending pattern, it is crucial to determine the age of the packet and the data it contains at the receiver side. Moreover, looking at the organization of the information exchange, packets can be sent between the nodes in peer-to-peer or centralized manner. However, a receiving endpoint node needs to monitor communication channels to particular nodes as well as the communication link regardless of the exact particular information exchange setting. Furthermore, whether the CO-CPSs are static or mobile (e.g., vehicles or sensors on moving equipment) affects the communication quality and should be considered when designing a channel state manager for CO-CPSs and choosing which indicators to monitor.

Note, that this step includes only the most general assumptions which are required for the design of the channel state manager. The channel state manager only provides additional information about the channel and is not responsible for specifying the CO-CPS's response to a change in channel reliability level. Other details as, e.g., the role of the CO-CPS in which the channel state

manager is to be placed or organization of the wireless network, can be considered, but these specifications do not affect the design of the manager. For example, the role of the node containing the developed manager can dictate particular safety reactions triggered by the device as a response to the decision made by the manager about the channel state, but not the channel state manager's decision itself. Similarly, more detailed knowledge about organization of the network, e.g., scheduling schemes or routing protocols, can be used for, e.g., localization of anomalies detected by the manager, but does not affect the monitoring itself and corresponding decisions.

7.3.2 Steps 1&2: Safety and Security Analyses and Requirements Elicitation

According to the proposed methodology, the first step, after the initial assumptions have been specified, is safety analysis and threat modeling to identify hazards and security threats that are relevant for the assumed context. As both safety and security related causes are considered, a joint safety and security analysis could be performed as well as parallel analyses. Within a classical safety analysis, e.g., HAZOP [144], hazards are identified and possible safety related causes are considered, e.g., with FTA. The safety analysis of the channel state manager in the context of a CO-CPS provides a set of hazards to which the manager can contribute. The system threat modeling, e.g., STRIDE [40], and mapping of system assets with system vulnerabilities and existing attacks, e.g., ATA, provide a set of relevant threats for the system that could be enabled by the channel state manager. Step 1 is detailed in Fig. 7.3, where it is shown that this step includes derivation of assumed CO-CPS hazards based on the provided context and has two sets of failures and threats as an outcome.

The set of general hazards affecting the cooperative system of systems assumed at this stage depends on the application area. Taking platooning as an example, one high level hazard for the whole system which can be named H_1, is *rear collision*. Examples of the failures that can contribute to this general application hazard are loss of the connection to the leader, commands from the leader which arrive too late or with errors. Thus, looking at a separate platooning vehicle, one assumed failure mode that can lead to H_1 is message omission, which can be interpreted as *absence of an expected packet at the receiving side by the required deadline*. Note that for safety-critical applications requiring reliable and timely data delivery not only packet loss, but also packet corruption or delay due an unreliable channel can result in the omission failure. This means that even correct packets arriving after the deadline can be

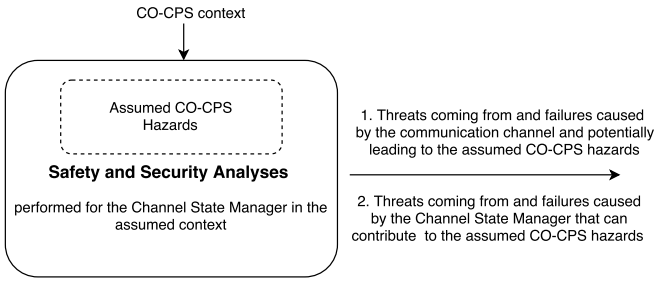

Figure 7.3: Detailed description of Step 1 — safety and security analyses

discarded by applications with hard real-time requirements. In general, there are three potential groups of failures that might manifest as omission failure modes [54]. First, it is hardware failures, which include failures of separate components within the communicating nodes which lead to the corresponding packets not being generated at all (e.g., breakage of sensors sensing the environment wrongly leads to not triggering communication) or not being sent or received (e.g., due to communication hardware failures). Next, it is software failures which are usually caused by errors in the code. The third group of possible failures are communication failures such as packet losses, delays or errors.

Given an initial assumption about required clock synchronization between the cooperating systems, another failure mode that may lead to H_1 is timing inconsistency failure mode where *a CO-CPS is not synchronized with other CO-CPSs*. Lack of timing consistency can lead to disturbances in the transmission schedule, which in their turn lead to packet collisions and thus, losses. Similarly to omission, timing inconsistency could be caused by software, hardware and communication failures as clock synchronization algorithms require information exchange between participants.

A security threat that is possible to assume for a CO-CPS, T_1, is: *data received from another CO-CPS is modified by an adversary*. For example, if a control message from the leading vehicle in a platoon is tampered, the receiving platooning car can start accelerating when it is commanded to slow down.

Taking the assumed CO-CPS hazards into account, safety and security analyses for the manager have to show how the manager can contribute to them. Thus, once safety and security analyses are conducted and the failures and threats that can contribute to CO-CPS hazards are identified, correspond-

ing requirements for the operation of the channel state manager can be elicited. There are generic guidelines for distributed monitoring design that should be considered on top of specific ones elicited from the analyses. For example, A. Goodloe and L. Pike [145] proposed a set of general principles for distributed monitor architectures related to functionality, reliability, schedulability and certifiability that could be used as a guideline for requirements elicitation. The monitor (*i*) shall not change functionality of the monitored system, i.e., a CO-CPS; (*ii*) shall not decrease its reliability; (*iii*) shall not cause a violation of hard real-time deadlines, (*iv*) shall not require unduly modifications of the monitored system. These principles should be applied for the channel state manager to ensure its usefulness for the CO-CPS. Thus, looking at the platooning example, one functional requirement that can be allocated to the black channel state manager is that it reports the link to the leader as unreliable if there are, e.g., five consecutively lost messages from the leader, or if, e.g., a security attack is detected. Also, deadlines for reporting can be set as the current channel state should be reported to the system state machine in a timely manner. Moreover, the channel state manger should not disrupt the normal behaviour of the system, e.g., it should not report that the channel is unreliable when it actually is reliable.

7.3.3 Step 3: Indicators Selection

Taking the omission failure mode as an example, communication failures were found as possibly manifesting into it and thus, by timely detection and reporting of channel quality degradation, the channel state manager may decrease the probability that a communication failure manifests into the omission failure mode. As it was shown in [54], examples of possible indicators to monitor in order to detect omission failure mode are *packet delivery rate* (PDR), which shows the number of correctly received packets out of the total number of sent packets or *the number of consecutive errors*, showing how the errors are distributed in time. Looking at the time inconsistency failure mode, indicators for detecting clock synchronization failures can be derived based on the time offsets which are calculated in the nodes (Chapter 6) and time-stamping of synchronization messages (Chapter 3). They include, e.g., *message inter-arrival time* (its fluctuation or constant trend towards decreasing or increasing indicates a degrading clock synchronization precision and may imply an attack), *mean* and *standard deviation* of the offset which is calculated in each node, and others. Note that the indicators might need to be adjusted to concrete scenarios and assumptions made in Step 0.

7.3.4 Step 4: Indicators Analysis

Once the indicators have been selected, they need to be analyzed in order to evaluate how indicative they are for estimation of the channel reliability and to find suitable thresholds for them. However, the actual values of thresholds can be obtained only in the integration phase of the channel state manager development. For example, the number of consecutive errors required to declare that channel reliability has degraded could be set to ten for one application and to three for another application with a higher criticality level. The parameter analysis also results in a set of channel states which can (based on the set of indicators and their corresponding thresholds) and should (based on the assumed requirements dictated by the applications) be distinguished by the manager. The trust in the correctness of the identification of the channel states which is done during the operation of the manager needs to be supported by evidences collected during the manager design, e.g., results of the sensitivity analysis of the selected indicators demonstrating their effectiveness in anomaly detection.

The monitor concept proposed in Section 5.1 is an example of a possible structure of channel states and transition rules for switching between them. As can be seen from the Fig. 7.4 demonstrating a possible example of channel states, the difference compare to Fig. 5.1 is that QS is renamed to a degraded state (DS), to reflect that now both safety and security are considered. The actual amount of states and details about switching rules need to be set considering the assumed application requirements of the CO-CPS and after the sensitivity analysis of the selected indicators. Sensitivity analysis demonstrates how indicators react on anomalies in the channel and thus how indicative they are and whether one more iteration with Step 3 is required. Within the example demonstrated in Fig. 7.4, there are three channel states, NS, DS and ADS. NS represents a normal situation, when communication channel reliability level is estimated as high enough for normal operation. On the contrary, ADS represents a situation when anomaly is assuredly detected and the system needs to be switched into its safe state, e.g., disengaging for a platooning car. DS is something in between, i.e., a situation when the reliability level of the channel is borderline or when there are not yet confirmed suspicions about an attack targeting wireless communication reliability. DS may imply that an increased minimum distance between platooning cars is needed as more time may be needed to avoid a collision. However, a system response to a channel state is defined by the system state manager which is left out of scope for this work.

A transition between states could be formalized as a rule that is later captured by a contract in Step 5. The decision about a transition can be made

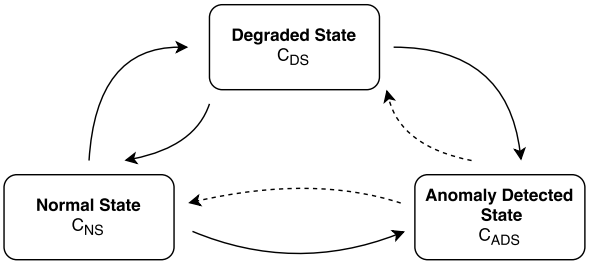

Figure 7.4: Channel states and transitions between them

based on values of one or several (e.g., to increase the overall trust level of decision correctness) indicators. Each parameter can be assigned one or several thresholds corresponding to different degrees of deviation of the observed value from a predefined operational value. For example, the channel reliability could be determined as degraded, i.e., the channel state is switched to DS, if at least one of the indicators has deviated significantly from its average value, i.e., exceeded its threshold. Furthermore, the channel reliability could be determined as not acceptably reliable, i.e., the channel state is switched into ADS, if several indicators have exceeded their thresholds. Also, different indicators could be monitored in different states. As DS corresponds to a situation when an anomaly is detected but more evidence of an attack and/or a further channel quality degradation is needed for the system to be switched into ADS, more indicators can be additionally monitored in DS. For example, as packet delivery rate indicates the channel quality, it can be checked in all states, while the number of consecutive errors can be checked only after at least some errors in a communication channel have been detected, i.e., it can be an additional parameter introduced in DS. Note, that transitions rules from ADS back to NS or DS are not described in Fig. 7.4. Once reliability of communication is again in place, i.e., conditions for entering NS or DS hold true, the channel could be switched from ADS back to NS or DS as it is shown by the dotted arrows in Fig. 7.4. However, note that without an identification of a particular cause for entering ADS it is not recommended to simply reset the system or a node since if the cause came from the security domain, an adversary can gain even more power while being present in the network during its rebooting.

Once the indicators have been analyzed, FS&SR can be decomposed to TS&SR. For example, the functional requirement that the channel state manager shall not disrupt the CO-CPS by reporting that the channel is not reliable,

while it is, can correspond to a technical requirement describing a transition rule to ADS.

7.3.5 Step 5: Preliminary Contracts

As it is shown in Fig. 7.4, each channel state has a contract, C_x, that specifies transitions from other states into this state (i.e., a situation when the entering conditions to the considered state are satisfied after conditions for another state have been satisfied) and transitions from this state to other states. Contracts can be derived by defining the states and transitions, e.g., as it was previously done in [54], where an adaptive contract group was defined. Preliminary contracts are finalized once the channel state manager is placed in the context of a particular CO-CPS. For example, depending on if the manager is used for a platoon of trucks or a platoon of cars, different thresholds for the same monitored indicators may be applied in order to identify a change of the channel state. Hence, the thresholds in the preliminary contracts stated out of context may be updated in the actual context.

.

7.4 Answering the Question

In this chapter a refinement of a safety element out-of-context contract-based development process is considered by tailoring the design phase specifically for the development of a black channel state manager. The role of the state manager is to monitor and assess the channel reliability. The manager is designed as a part of a cooperating system and aimed to detect communication anomalies in a black channel and report the assessed channel state. The design phase is split into six steps all of which are performed out of a particular system context as an introduced channel state manager developing process is aligned with the development process of a safety element out-of-context. The steps are detailed and discussed on examples of cooperative systems of systems, e.g., a car platoon. The discussed channel state manager, if being placed within each cooperative system, enables common perception of the channel reliability among cooperating systems, i.e., individual assessment of each CO-CPS channel, but with the same rationale behind, which is important for synchronization of the states and safety tactics amongst cooperating systems.

Thus, answering the question, there is no evidence that the method of addressing monitoring in the way presented in Chapters 3-6 cannot be abstracted

from considering only clock synchronizarion and into a more general purpose. The steps detailed in previous chapters fit as a particular case of design methodology into the steps presented in this chapter.

Chapter 8

Conclusions and Future Work

The era of increased connectivity in cyber-physical systems supported by flexible wireless solutions has inspired highly interconnected systems such as Industry 4.0 and cooperative automotive systems. Such systems have high complexity, outer dependencies, cooperation functionalities and humans in the loop. Thus, security rises as an essential property to consider for such systems, as they are not isolated, not static and often safety-critical. Timing properties for such systems are of high concern as they enable cooperation in a timely manner and support for real-time requirements. This thesis addresses a core asset for systems that have time-triggered architecture or systems which require a coherent notion of time within system and between components, namely clock synchronization. Once this asset was identified, possibilities to break it were investigated. A selective delay attack was identified as the most challenging to counteract. A solution to detect such an attack using distributed monitoring was proposed. The interaction between the monitor and a potential adversary was modeled using game theory. Further, a set of indicators to monitor was proposed and analyzed in order to evaluate their effectiveness. A set of several monitor states was introduced to include the situation when there are suspicions, but an attack is not confirmed and additional evidences are needed. Switching between states is based on indicator values and their corresponding thresholds. The process of indicator thresholding was investigated and their sensitivity in regard to system parameters, its environment and attack model was examined. The monitoring approach and an indicator analysis were eval-

uated using an example of cyber-physicals systems. The complexity level and interconnections of cyber-physical systems make their safety and security requirements interdependent. A system is not safe if it is not secure and vice verse. Thus, safety and security interconnections and implications on each other were considered in clock synchronization. The joint approach for assurance of these properties was argued. From a monitoring point of view, safety and security causes for abnormal behaviour are not distinguishable without additional techniques. Thus, the developed approach was generalized to estimate communication channel reliability on an example with a black channel model for communications between cooperative systems. As an outcome a methodology for the design of black channel state manager was considered.

The research questions are brought up again in the sections below to map conclusions for them.

8.1 Answering Research Question 1

RQ1: What are the main system assets and security objectives for cyber-physical systems?

There are many relevant assets to consider for cyber-physical systems. However one of the crucial assets needed to support real-time requirements and to establish a notion of time is clock synchronization. Cyber-physical systems by their definition imply timing as a functional requirement. Thus, this thesis is focused on clock synchronization as an essential asset to protect. However, it is not straightforward to connect it with a set of security objectives, as those are tightly dependent on the particular threat model and consequently on the considered use case. The correspondence between an asset and the security objectives relevant for it can be established within the proposed threat model.

8.2 Answering Research Question 2

RQ2: How can the main system assets be protected and the needed security objectives provided in the presence of real-time requirements?

Protection could imply detection of a problem in time which allows taking actions to prevent or mitigate its consequences. Hence, one of the ways to protect clock synchronization is to detect it being broken in time, i.e., before the consequences of it being broken become problematic. Monitoring also

has low communication overhead, thus it does not require additional efforts, e.g., rescheduling, for continuous support of real-time requirements. The main challenge in monitoring is to identify what to monitor, i.e., which parameters are indicative enough and allow to determine the presence of an anomaly. In this work, offset calculated according to the considered clock synchronization algorithm, PTP, and messages belonging to the protocol enabling clock synchronization, were considered as input data. Thus, two directions were investigated: (*i*) message patterns, allowing to detect a sudden change if it is significant enough; (*ii*) offset statistical characteristics allowing to detect a new trend in the values. Further, statistical characteristics were analyzed in more details to evaluate their applicability and effectiveness.

8.3 Answering Research Question 3

RQ3: How will protection of the system assets influence system safety and dependability?

Safety and security as system properties are interconnected. Clock synchronization is an example of how security can influence safety. Security causes can contribute to hazards and thus can influence results of the risk assessment and consequently which hazards are classified as important to address. Moreover, as security implies an adversary and malicious intensions, prevention and mitigation techniques may differ. Security and safety solutions could contradict each other as well, hence safety and security analyses should be done jointly or at least considering each other, so that conflicts could be resolved on the requirement level. However, monitoring is a solution beneficial to both, as it does not impose protection techniques that could support one property and jeopardize another. The proposed monitoring approach was developed from security considerations of clock synchronization, however it could be applied to detect an anomaly in clock synchronization coming from a failure as wel as from an attack.

8.4 Future Work

There are many directions to develop the work further. First, an implementation of the state manager that realizes clock synchronization anomaly monitoring to evaluate the effectiveness of the developed solution in a static application and then adopting the solution to support mobile nodes. Next, obtaining

the time reference and investigate different options to include it in a case study. Further, considering the system state machine and its connection to the channel state manager is interesting future work. The next possible direction is developing a joint assurance pattern from what was demonstrated in this work. Including security in safety assurance enables run-time adaptation, as security is a dynamic system property, thus investigating of mapping between security updates and a safety/security case is a challenging task to address. Finally, examining different redundant paths and changing the routing/scheduling based on monitor outputs regarding channel reliability, constitutes an interesting remedy.

Bibliography

[1] M. G. Doyle. How Volvo CE is engineering a quarry run by electric loaders and haulers for big cuts to costs and emissions, 2016. [Online; accessed 18-April-2017].

[2] German Institute of Standartization (DIN). DIN 19 245 - PROFIBUS Standard Part 1 and 2, 1991.

[3] International Organization for Standartization (ISO). ISO 11898 - Road vehicles - Controllerarea network (CAN) , 2015.

[4] W. Steiner, G. Bauer, B. Hall, and M. Paulitsch. TTEthernet: Time-Triggered Ethernet. *Time-Triggered Communication*, 2011.

[5] D. Miorandi, E. Uhlemann, S. Vitturi, and A. Willig. Guest Editorial: Special Section on Wireless Technologies in Factory and Industrial Automation, Part I. *IEEE Transactions on Industrial Informatics*, 3(2):95–98, May 2007.

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

[7] S. Raza and T. Voigt. Interconnecting WirelessHART and Legacy HART Networks. In *6th IEEE International Conference on Distributed Computing in Sensor Systems Workshops (DCOSSW)*, pages 1–8, Jun. 2010.

[8] R. Baheti and H. Gill. Cyber-hysical Systems. *The impact of control technology*, 12:161–166, 2011.

[9] Y. B. Reddy. Cloud-Based Cyber Physical Systems: Design Challenges and Security Needs. In *10th International Conference on Mobile Ad-hoc and Sensor Networks*, pages 315–322, Dec. 2014.

[10] CENELEC. IEC 61508: Functional Safety of E/E/PE Safety-Related Systems. Part 2: Requirements for E/E/PE Safety-Related Systems, 2007.

[11] J. Åkerberg, M. Gidlund, and M. Björkman. Future Research Challenges in Wireless Sensor and Actuator Networks Targeting Industrial Automation. In *9th IEEE International Conference on Industrial Informatics*, pages 410–415, Jul. 2011.

[12] R. Bloomfield and P. Bishop. Safety and Assurance Cases: Past, Present and Possible Future – An Adelard Perspective. In *Making Systems Safer*, pages 51–67. Springer, 2010.

[13] C. B. Weinstock, H. F. Lipson, and J. B. Goodenough. Arguing Security Creating Security Assurance Cases. 2014.

[14] E. Lisova, E. Uhlemann, J. Åkerberg, and M. Björkman. Towards Secure Wireless TTEthernet for Industrial Process Automation Applications. In *IEEE Emerging Technology and Factory Automation (ETFA)*, pages 1–4, Sep. 2014.

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

[16] H. Kopetz and W. Ochsenreiter. Clock Synchronization in Distributed Real-Time Systems. *IEEE Transactions on Computers*, C-36(8):933–940, Aug 1987.

[17] N.C. Audsley, A. Burns, R.I. Davis, K. Tindell, and A.J. Wellings. Fixed Priority Pre-emptive Scheduling: An Historic Perspective. *Real-Time Systems*, 8(2/3):173–198, 1995.

[18] L. Sha, T. Abdelzaher, K.-E. Årzén, A. Cervin, T. P. Baker, A. Burns, G. Buttazzo, M. Caccamo, J. P. Lehoczky, and A. K. Mok. Real Time Scheduling Theory: A Historical Perspective. *Real-Time Systems*, 28(2/3):101–155, 2004.

[19] L. T. Heberlein, G. V. Dias, K. N. Levitt, B. Mukherjee, J. Wood, and D. Wolber. A Network Security Monitor. In *IEEE Computer Society Symposium on Research in Security and Privacy*, pages 296–304, May 1990.

[20] P. Johnson, D. Gorton, R. Lagerstrom, and M. Ekstedt. Time Between Vulnerability Disclosures: A Measure of Software Product Vulnerability. *Computers and Security*, 62:278–295, 2016.

[21] E. Denney, G. Pai, and I. Habli. Dynamic Safety Cases for Through-life Safety Assurance. In *37th International Conference on Software Engineering (ICSE)*, pages 587–590, May 2015.

[22] IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems. *IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002)*, pages 1–269, Jul. 2008.

[23] A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuellar, P. Hankes Drielsma, P. C. Heam, O. Kouchnarenko, J. Mantovani, S. Modersheim, D. von Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Vigano, and L. Vigneron. The AVISPA Tool for the Automated Validation of Internet Security Protocols and Applications. In *The 17th International Conference on Computer Aided Verification*, pages 281–285, Jul. 2005.

[24] R. Snieder and K. Larner. *The Art of Being a Scientist: A Guide for Graduate Students and their Mentors*. Cambridge University Press, 7st edition, 2015.

[25] G. Dodig-Crnkovic. *Constructive Research and Info-computational Knowledge Generation*, pages 359–380. 2010.

[26] Hilary J. Holz, A. Applin, B. Haberman, D. Joyce, H. Purchase, and C. Reed. Research Methods in Computing: What Are They, and How Should We Teach Them? In *Working Group Reports on ITiCSE on Innovation and Technology in Computer Science Education*, pages 96–114. ACM, 2006.

[27] OMNeT++. http://www.omnetpp.org/, 21 January 2015.

[28] M. Shaw. The Coming-of-age of Software Architecture Research. In *23rd International Conference on Software Engineering (ICSE)*, May 2001.

[29] Jean-François Bonnefon, Azim Shariff, and Iyad Rahwan. Autonomous Vehicles Need Experimental Ethics: Are We Ready for Utilitarian Cars? *CoRR*, 2015.

[30] P. Ohm. Broken promises of privacy: Responding to the surprising failure of anonymization. *UCLA law review*, 57:1701, 2010.

[31] I. Šljivo, E. Lisova, and S. Afshar. Agent-Centred Approach for Assuring Ethics in Dependable Service Systems. In *IEEE World Congress on Services (SERVICES)*, pages 51–58, Jun. 2017.

[32] D. R. Forsyth. A Taxonomy of Ethical Ideologies. *Journal of Personality and Social Psychology*, 39(1):175–184, 1980.

[33] R. Kissel. *Glossary of key information security terms*. U.S. Dept. of Commerce, National Institute of Standards and Technology, 2006.

[34] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr. Basic Concepts and Taxonomy of Dependable and Secure Computing. *IEEE Transactions on Dependable Secue. Computing*, 1(1):11–33, Jan. 2004.

[35] N. Kuntze, C. Rudolph, B. Brisbois, M. Boggess, B. Endicott-Popovsky, and S. Leivesley. Security vs. Safety: Why Do People Die Despite Good Safety? In *Integrated Communication, Navigation, and Surveillance Conference (ICNS)*, Apr. 2015.

[36] P. Johnson, D. Gorton, R. Lagerstrm, and M. Ekstedt. Time Between Vulnerability Disclosures: A Measure of Software Product Vulnerability. *Computers and Security*, 62:278–295, 2016.

[37] F. Swiderski and W. Snyder. *Threat Modeling*. Microsoft Press, Redmond, WA, USA, 2004.

[38] SAEJ3061. Cybersecurity Guidebook for Cyber-Physical Vehicle Systems. SAE International, 2016.

[39] ISO/IEC 15408. Common Criteria for Information Technology Security Evaluation. 2012.

[40] Microsoft Corporation. The stride threat model, 2005.

[41] A. Shostack. *Threat Modeling: Designing for Security*. Wiley Publishing, 1st edition, 2014.

[42] B. Schneier. Attack Trees. *Dr. Dobb's Journal*, 24(12):21–29, 1999.

[43] E. Ruijters, S. Schivo, M. Stoelinga, and A. Rensink. Uniform analysis of fault trees through model transformations. In *Annual Reliability and Maintainability Symposium (RAMS)*, pages 1–7, Jan 2017.

[44] Kim Guldstrand Larsen, Paul Pettersson, and Wang Yi. UPPAAL in a Nutshell. *Springer International Journal of Software Tools for Technology Transfer*, 1(1+2):134–152, May 1997.

[45] R. Kumar and M. Stoelinga. Quantitative Security and Safety Analysis with Attack-Fault Trees. In *IEEE 18th International Symposium on High Assurance Systems Engineering (HASE)*, pages 25–32, Jan. 2017.

[46] W. Young and N. Leveson. Systems Thinking for Safety and Security. In *29th Annual Computer Security Applications Conference*, Dec. 2013.

[47] N. G. Leveson. *Engineering a Safer World: Systems Thinking Applied to Safety (Engineering Systems)*. The MIT Press, 1st edition, 2012.

[48] CENELEC. *IEC 61508: Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems. Part 4: Definitions and abbreviations*. UK Ministry of Defence, 2007.

[49] D.F. Haasl, N.H. Roberts, W.E. Vesely, and F.F. Goldberg. *Fault Tree Handbook*. Jan. 1981.

[50] R.E Melchers. On the ALARP Approach to Risk Management. *Reliability Engineering and System Safety*, 71(2):201 – 208, 2001.

[51] T. Kelly. *Arguing Safety — A Systematic Approach to Managing Safety Cases*. PhD thesis, University of York, York, UK, 1998.

[52] T. Kelly and R. Weaver. The Goal Structuring Notation – a Safety Argument Notation. In *Dependable Systems and Networks Workshop on Assurance Cases*, Jun. 2004.

[53] A. Benveniste, B. Caillaud, D. Nickovic, R. Passerone, J.-B. Raclet, P. Reinkemeier, A. Sangiovanni-Vincentelli, W. Damm, T. Henzinger, and K. G. Larsen. Contracts for System Design. Research Report RR-8147, INRIA, Nov. 2012.

[54] S. Girs, I. Šljivo, and O. Jaradat. Contract-Based Assurance for Wireless Cooperative Functions of Vehicular Systems. In *43rd Annual Conference of the IEEE Industrial Electronics Society (IECON)*, Oct. 2017.

[55] International Organization for Standardization (ISO). ISO 26262: Road Vehicles - Functional Safety, 2011.

[56] I. Šljivo, B. Gallina, J. Carlson, and H. Hansson. Using Safety Contracts to Guide the Integration of Reusable Safety Elements within ISO 26262. In *The 21st IEEE Pacific Rim International Symposium on Dependable Computing*, Nov, 2015.

[57] J. L. Welch and N. Lynch. A New Fault-tolerant Algorithm for Clock Synchronization. *Information and Computation*, 77(1):1–36, Apr. 1988.

[58] H. Kopetz and W. Ochsenreiter. Clock Synchronization in Distributed Real-Time Systems. *IEEE Transactions on Computers*, C-36(8):933–940, Aug. 1987.

[59] L. Lamport and P. M. Melliar-Smith. Byzantine Clock Synchronization. In *3rd Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 68–74, Aug. 1984.

[60] D. L. Mills. Internet Time Synchronization: the Network Time Protocol. *IEEE Transactions on Communications*, 39(10):1482–1493, Oct. 1991.

[61] D. Mills and U. Delaware. Network Time Protocol Version 4: Protocol and Algorithms Specification. 2010.

[62] M. Ullmann and M. Vogeler. Delay Attacks - Implication on NTP and PTP Time Synchronization. In *International Symposium on Precision Clock Synchronization for Measurement, Control and Communication (ISPCS)*, pages 1–6, Oct. 2009.

[63] T. Neagoe, V. Cristea, and L. Banica. NTP versus PTP in Computer Networks Clock Synchronization. In *IEEE International Symposium on Industrial Electronics (ISIE)*, volume 1, pages 317–362, Jul. 2006.

[64] IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems. *IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002)*, pages 1–269, Jul. 2008.

[65] M. Lipiski, T. Wostowski, J. Serrano, and P. Alvarez. White rabbit: a PTP application for robust sub-nanosecond synchronization. In *IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication*, pages 25–30, Sep. 2011.

[66] B. Hirschler and A. Treytl. Validation and Verification of IEEE 1588 Annex K. In *International IEEE Symposium on Precision Clock Synchronization for Measurement Control and Communication (ISPCS)*, pages 44–49, Sep. 2011.

[67] T. Mizrahi. A Game Theoretic Analysis of Delay Attacks Against Time Synchronization Protocols. In *International IEEE Symposium on Precision Clock Synchronization for Measurement Control and Communication (ISPCS)*, pages 1–6, Sep. 2012.

[68] IEEE Standard for Local and metropolitan area networks– Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks– Corrigendum 1: Technical and Editorial Corrections. *IEEE Std 802.1AS-2011/Cor 1-2013 (Corrigendum to IEEE Std 802.1AS-2011)*, pages 1–128, Sep. 2013.

[69] M. Gutiérrez, W. Steiner, R. Dobrin, and S. Punnekkat. Synchronization Quality of IEEE 802.1AS in Large-Scale Industrial Automation Networks. In *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 273–282, Apr. 2017.

[70] M. Levesque and D. Tipper. ptp++: A Precision Time Protocol Simulation Model for OMNeT++ / INET. *CoRR*, abs/1509.03169, 2015.

[71] A. Treytl and B. Hirschler. Security Flaws and Workarounds for IEEE 1588 (Transparent) Clocks. In *International Symposium on Precision Clock Synchronization for Measurement, Control and Communication (ISPCS)*, pages 1–6, Oct. 2009.

[72] E. Lisova, E. Uhlemann, W. Steiner, J. Åkerberg, and M. Björkman. Risk Evaluation for Clock Synchronization from ARP Poisoning attack in Industrial Applications. In *IEEE International Conference onIndustrial Technology (ICIT)*, Mar. 2016.

[73] IEC Industrial Communication Networks - Profiles - Part 3: Functional Safety Fieldbuses - General Rules and Profile Definitions. *IEC Std 61784-3-2016*, pages 1–166, May 2016.

[74] S. H. Lee, K. S. Son, W. Jung, and H. G. Kang. Risk Assessment of Safety Data Link and Network Communication in Digital Safety Feature Control System of Nuclear Power Plant. *Annals of Nuclear Energy* , 108:394 – 405, 2017.

[75] P. Gaj, J. Jasperneite, and M. Felser. Computer Communication Within Industrial Distributed Environment . *IEEE Transactions on Industrial Informatics*, 9(1):182–189, Feb. 2013.

[76] D. K. Nilsson and U. Larson. A defense-in-depth Approach to Securing the Wireless Vehicle Infrastructure. *Journal of Network*, 4:552–564, 2009.

[77] Ozgur Depren, Murat Topallar, Emin Anarim, and M. Kemal Ciliz. An intelligent intrusion detection system (IDS) for anomaly and misuse detection in computer networks. *Expert Systems with Applications*, 29(4):713 – 722, 2005.

[78] M. Wooldridge. Does Game Theory Work? *IEEE Intelligent Systems*, 27(6):76–80, Nov 2012.

[79] S. Roy, C. Ellis, S. Shiva, D. Dasgupta, V. Shandilya, and Q. Wu. A Survey of Game Theory as Applied to Network Security. In *43rd Hawaii International Conference on System Sciences*, pages 1–10, Jan. 2010.

[80] V. Srivastava, J. Neel, A. B. Mackenzie, R. Menon, L. A. Dasilva, J. E. Hicks, J. H. Reed, and R. P. Gilles. Using Game Theory to Analyze Wireless Ad Hoc Networks. *IEEE Communications Surveys Tutorials*, 7(4):46–56, 2005.

[81] W. Yu and K. J. R. Liu. Game Theoretic Analysis of Cooperation Stimulation and Security in Autonomous Mobile Ad Hoc Networks. *IEEE Transactions on Mobile Computing*, 6(5):507–521, May 2007.

[82] A. Agah, S. K. Das, K. Basu, and M. Asadi. Intrusion Detection in Sensor Networks: a Non-cooperative Game Approach. In *3rd IEEE International Symposium on Network Computing and Applications*, pages 343–346, Aug. 2004.

[83] T. Alpcan and T. Basar. A Game Theoretic Approach to Decision and Analysis in Network Intrusion Detection. In *42nd IEEE International Conference on Decision and Control*, volume 3, pages 2595–2600, Dec. 2003.

[84] S. Buchegger and T. Alpcan. Security Games for Vehicular Networks. In *46th Annual Allerton Conference on Communication, Control, and Computing*, pages 244–251, Sep. 2008.

[85] M. H. Manshaei, Q. Zhu, T. Alpcan, T. Bacşar, and J.-P. Hubaux. Game Theory Meets Network Security and Privacy. *ACM Computing Surveys*, 45(3):25:1–25:39, Jul. 2013.

[86] M. Garuba, Chunmei Liu, and D. Fraites. Intrusion Techniques: Comparative Study of Network Intrusion Detection Systems. In *5th International Conference on Information Technology: New Generations* , pages 592–598, Apr. 2008.

[87] S. E. Chodrow, F. Jahanian, and M. Donner. Run-time Monitoring of Real-time Systems. In *12th Real-Time Systems Symposium*, pages 74–83, Dec. 1991.

[88] P. Pop, D. Scholle, I. Šljivo, H. Hansson, G. Widforss, and M. Rosqvist. Safe Cooperating Cyber-Physical Systems using Wireless Communication. *Elsevier journal of Microprocessors and Microsystems* , 53:42–50, Jul. 2017.

[89] A. J. Ramirez, B. H. C. Cheng, N. Bencomo, and P. Sawyer. *Relaxing Claims: Coping with Uncertainty While Evaluating Assumptions at Run Time*, pages 53–69. Oct. 2012.

[90] X. Duan, J. Zhang, Q. Bao, R. Ramachandran, T. J. Lee, S. Lee, and L. Pan. Linking Design-Time and Run-Time: A Graph-Based Uniform Workflow Provenance Model. In *IEEE International Conference on Web Services (ICWS)*, pages 97–105, Jun. 2017.

[91] A. Kane. *Runtime Monitoring for Safety-Critical Embedded Systems*. PhD thesis, Carnegie Mellon University, 2015.

[92] D. Arora, S. Ravi, A. Raghunathan, and N. K. Jha. Secure Embedded Processing Through Hardware-assisted Run-time Monitoring. In *Design, Automation and Test in Europe*, pages 178–183 Vol. 1, Mar. 2005.

[93] M. Stoicescu, J.-C. Fabre, and M. Roy. *Architecting Resilient Computing Systems: Overall Approach and Open Issues*.

[94] M. Gutiérrez, W. Steiner, R. Dobrin, and S. Punnekkat. A Configuration Agent Based on the Time-triggered Paradigm for Real-time Networks. In *IEEE World Conference on Factory Communication Systems (WFCS)*, pages 1–4, May 2015. Best Work-in-Progress Paper Award.

[95] M. Gutiérrez, W. Steiner, R. Dobrin, and S. Punnekkat. Learning the Parameters of Periodic Traffic Based on Network Measurements. In *IEEE International Workshop on Measurements & Networking (M&N)*, 2015.

[96] M. Eslahi, M.S. Rohmad, H. Nilsaz, M.V. Naseri, N.M. Tahir, and H. Hashim. Periodicity Classification of HTTP Traffic to Detect HTTP Botnets. In *IEEE Symposium on Computer Applications Industrial Electronics (ISCAIE)* , pages 119–123, Apr. 2015.

[97] G.A. Fink, B.L. Chappell, T.G. Turner, and K.F. O'Donoghue. A Metrics-based Approach to Intrusion Detection System Evaluation for Distributed Real-time Systems. In *Parallel and Distributed Processing Symposium (IPDPS)*, Apr. 2002.

[98] M. Anand, E. Cronin, M. Sherr, M. Blaze, Z. Ives, and I. Lee. Sensor Network Security: More Interesting Than You Think. In *1st USENIX Workshop on Hot Topics in Security (HOTSEC)* , Jul. 2006.

[99] T. Mizrahi. Time Synchronization Security Using IPsec and MACsec. In *IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication (ISPCS)*, pages 38–43, Sep. 2011.

[100] J. A. Clark, J. Murdoch, J. A. Mcdermid, S. Sen, H. R. Chivers, O. Worthington, and P. Rohatgi. Threat Modelling for Mobile Ad Hoc and Sensor Networks. In *Second International Confrence on Internet Technologies and Application*, Wrexham, Nort Wales, UK,, Sep. 2007.

[101] D. Bruschi, A. Ornaghi, and E. Rosti. S-ARP: a Secure Address Resolution Protocol. In *19th Annual Computer Security Applications Conference*, pages 66–74, Dec. 2003.

[102] F. T. Sheldon, J. M. Weber, S. M. Yoo, and W. D. Pan. The Insecurity of Wireless Networks. *IEEE Security Privacy*, 10(4):54–61, Jul. 2012.

[103] E. Lisova, E. Uhlemann, J. Åkerberg, and M. Björkman. Delay Attack Versus Clock Synchronization — A Time Chase. In *IEEE International Conference on Industrial Technology (ICIT)*, pages 1136–1141, Mar. 2017.

[104] N. Tripathi and B. M. Mehtre. Analysis of Various ARP Poisoning Mitigation Techniques: A Comparison. In *International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT)*, pages 125–132, Jul. 2014.

[105] S. Kumar and S. Tapaswi. A Centralized Detection and Prevention Technique against ARP Poisoning. In *International Conference on Cyber Security, Cyber Warfare and Digital Forensic (CyberSec)*, pages 259–264, Jun. 2012.

[106] F. A. Barbhuiya, S. Biswas, and S. Nandi. An Active DES Based IDS for ARP Spoofing. In *IEEE International Conference on Systems, Man, and Cybernetics*, pages 2743–2748, Oct. 2011.

[107] A. P. Ortega, X. E. Marcos, L. D. Chiang, and C. L. Abad. Preventing ARP Cache Poisoning Attacks: A Proof of Concept using OpenWrt. In *Latin American Network Operations and Management Symposium*, pages 1–9, Oct. 2009.

[108] C. Onal and H. Kirrmann. Security Improvements for IEEE 1588 Annex K: Implementation and Comparison of Authentication Codes. In *International IEEE Symposium on Precision Clock Synchronization for Measurement Control and Communication (ISPCS)*, pages 1–6, Sep. 2012.

[109] TCXO, Temperature Compensated Crystal Oscillator.

[110] Y. Glouche, T. Genet, O. Heen, and O. Courtay. A Security Protocol Animator Tool for AVISPA. In *ARTIST-2 Workshop*, Mar. 2006.

[111] D. Basin, S. Mödersheim, and L. Viganò. OFMC: A Symbolic Model Checker for Security Protocols. *International Journal of Information Security*, 4(3):181–208, Jun 2005.

[112] INET Framework. https://inet.omnetpp.org/, 21 Jun. 2015.

[113] E. Lisova, M. Gutiérrez, W. Steiner, E. Uhlemann, J. Åkerberg, R. Dobrin, and M. Björkman. Protecting Clock Synchronization: Adversary Detection through Network Monitoring. *Journal of Electrical and Computer Engineering*, pages 1–13, 2016.

[114] A. Burns, J. McDermid, and J. Dobson. On the Meaning of Safety and Security. *Computer Journal*, 35(1), 1992.

[115] David F. C. Brewer. *Applying Security Techniques to Achieving Safety*. Springer London, Feb. 1993.

[116] J. Rushby. Critical System Properties: Survey and Taxonomy. Technical Report SRI-CSL-93-1, International Computer Science Laboratory (SRI), 1994.

[117] D. P.R. Eames and J. Moffett. *The Integration of Requirements*. Sep. 1999.

[118] C. W. Axelrod. Applying Lessons from Safety-critical Systems to Security-critical Software. In *IEEE Systems, LISAT*, May 2011.

[119] Terje Aven. A Unified Framework for Risk and Vulnerability Analysis Covering both Safety and Security . *Reliability Engineering and System Safety*, 92(6), 2007.

[120] C. Ponsard, G. Dallons, and P. Massonet. *Goal-Oriented Co-Engineering of Security and Safety Requirements in Cyber-Physical Systems*. Trondheim, Norway, Sep. 2016.

[121] T. Srivatanakul, John A. Clark, and Fiona Polack. *Effective Security Requirements Analysis: HAZOP and Use Cases*. Sep. 2004.

[122] R. Winther, O.-A. Johnsen, and B. A. Gran. *Security Assessments of Safety Critical Systems Using HAZOPs*. Sep. 2001.

[123] D. Schneider, E. Armengaud, and E. Schoitsch. *Towards Trust Assurance and Certification in Cyber-Physical Systems*. Springer, 2014.

[124] H. Kopetz and W. Ochsenreiter. Clock Synchronization in Distributed Real-Time Systems. *IEEE Transactions on Computers*, C-36(8):933–940, Aug. 1987.

[125] M. Bouzeghoub. A Framework for Analysis of Data Freshness. In *International Workshop on Information Quality in Information Systems (IQIS)*, pages 59–67, Jun. 2004.

[126] R. David. Productivity Improvements in Construction Site Operations through Lean Thinking and Wireless Real-Time Control, Dec. 2014.

[127] V. Chiprianov, L. Gallon, M. Munier, and V. Lalanne. Challenges in Security Engineering of Systems-of-Systems, Dec. 2014.

[128] D. G. Lubas. Department of Defense System of Systems Reliability Challenges. In *Annual Reliability and Maintainability Symposium (RAMS)*, pages 1–6, Jan. 2017.

[129] D. Mills. Network Time Protocol (Version 3) Specification, Implementation. 1992.

[130] E. Lisova, A. Čaušević, E. Uhlemann, and M. Björkman. Clock Synchronization Considerations in Security Informed Safety Assurance of Autonomous Systems of Systems. In *43rd Annual Conference of the IEEE Industrial Electronics Society (IECON)*, Oct. 2017.

[131] PJ Redmond, 2007.

[132] E. Troubitsyna. An Integrated Approach to Deriving Safety and Security Requirements from Safety Cases. In *The 40th IEEE Computer Society International Conference on Computers, Software and Applications (COMPSAC)*, volume 2, pages 614–615, Jun. 2016.

[133] S. H. Lin, T. C. Lee, and M. F. Gardina. Diversity Protections for Digital Radio-summary of Ten-year Experiments and Studies. *IEEE Communications Magazine*, 26(2):51–63, Feb. 1988.

[134] J. D. Perezgonzalez. Fisher, Neyman-Pearson or NHST? A tutorial for teaching data testing. *Frontiers in Psychology*, 6:223, 2015.

[135] H. L. Van Trees. *Classical Detection and Estimation Theory*. John Wiley and Sons, 2001.

[136] G. Gaderer, R. Holler, T. Sauter, and H. Muhr. Extending IEEE 1588 to Fault Tolerant Clock Synchronization. In *IEEE International Workshop on Factory Communication Systems*, pages 353–357, Sep. 2004.

[137] P.A.S. Ralston, J.H. Graham, and J.L. Hieb. Cyber Security Risk Assessment for SCADA and DCS Networks. *ISA Transactions*, 46(4):583 – 594, 2007.

[138] A. Moore, R. Ellison, and R. Linger. Attack Modeling for Information Security and Survivability. Technical Report CMU/SEI-2001-TN-001, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 2001.

[139] W. Steiner. An Evaluation of SMT-Based Schedule Synthesis for Time-Triggered Multi-hop Networks. In *31st IEEE Real-Time Systems Symposium*, pages 375–384, Nov. 2010.

[140] H. H. Feng, J. T. Giffin, Yong Huang, S. Jha, Wenke Lee, and B. P. Miller. Formalizing Sensitivity in Static Analysis for Intrusion Detection. In *Proceedings IEEE SSP*, pages 194–208, May 2004.

[141] W. Lee, W. Fan, M. Miller, S. J. Stolfo, and E. Zadok. Toward Cost-sensitive Modeling for Intrusion Detection and Response. *Journal of Computer Security*, 10(1-2):5–22, Jul. 2002.

[142] I. Šljivo, B. Gallina, and B. Kaiser. Assuring Degradation Cascades of Car Platoons via Contracts. In *6th International Workshop on Next Generation of System Assurance Approaches for Safety-Critical Systems*, volume 10489, pages 317–329, Sep. 2017.

[143] E. Lisova, S. Girs, and I. Šljivo. Designing a Reusable Black Channel State Manager for Cooperative Systems. *submitted to The 23rd International Conference on Reliable Software Technologies Ada-Europe 2018*, 2018.

[144] *A Guide to Hazard and Operability Studies*. Chemical Industry Safety and Health Council of the Chemical Industries Association, 1977.

[145] A. Goodloe and L. Pike. Monitoring Distributed Real-Time Systems: A Survey and Future Directions. Technical Report NASA/CR-2010-216724, NASA Langley Research Center, Jul. 2010.

Appendices

Abbreviations

ADS	Anomaly Detected State
AFT	Attack-Fault Trees
ALARP	As Low As Reasonable Practicable
ARP	Address Resolution Protocol
ATA	Attack Tree Analysis
AVB	Audio Video Bridging
AVISPA	Automated Validation of Internet Security Protocols and Applications
BMC	Best Master Clock
CD	Constant Delay
CIA	Confidentiality, Integrity and Availability
CLAtSe	Constraint-Logic-based Attack Searcher
CO-CPS	Cooperative CPS
CPS	Cyber-Physical System
DoS	Denial of Service
DS	Degraded State
ECU	Electronic Control Unit
E/E/PE	Electrical/Electronic/Programmable Electronic

FS&SR	Functional Safey and Security Requirements
FTA	Fault Tree Analysis
GM	Grand Master
GSN	Goal Structuring Notation
HAZOP	Hazard and Operability Study
HLPSL	High-Level Protocol Specification Language
ICV	Integrity Check Value
IDS	Intrusion Detection System
IoT	Internet of Things
IT	Information Technologies
LAN	Local Area Network
LID	Linearly Increasing Delay
MIM	Man-In-the-Middle
NS	Normal State
NTP	Network Time Protocol
OFMC	On-the-fly Model Checker
OSI	Open Systems Interconnection
OT	Operational Technologies
PTP	Precision Time Protocol
QS	Quarantine State
RD	Random Delay
PDF	Probability Density Function
PDR	Packet Delivery Rate
RI	Resynchronization Interval

RQ	Research Question
SA	Security Associations
SATMC	SAT-based Model Checker
SEooC	Safety Element out-of-Contex
SoS	Systems of Systems
SPAN	Security Protocol Animator
STAMP	Systems Theoretic Accident Modeling and Processes
STPA	System-Theoretic Process Analysis
STPA-Sec	System-Theoretic Process Analysis for Security
STRIDE	Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elicitation of Privilege
TA4SP	Automatic Approximations for the Analysis of Security Protocols
TDMA	Time Division Multiple Access
TSN	Time-Sensitive Networking
TS&SR	Technical Safety and Security Requirements
UDP	User Datagram Protocol
VANET	Vehicular Ad hoc Networks
WR	White Rabbit