Dipl.-Ing. Lukas Alexander Gressl, BSc.

Towards Security-Aware Design Space Exploration for Embedded Systems

DOCTORAL THESIS

to achieve the university degree of

Doktor der technischen Wissenschaften

submitted to

Graz University of Technology

Supervisor

Univ.-Prof. Dipl.-Inform. Dr. sc. ETH Kay Uwe Römer

Institute of Technical Informatics

Advisor
Ass. Prof. Dipl.-Ing. Dr. techn. Christian Steger

Graz, October 2020

AFFIDAVIT

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present doctoral thesis.

Date	Signature

Acknowledgements

During my Ph.D. studies, I met people who inspired me and with whom I was able to discuss interesting ideas. I was fortunate to work on various topics with some of those. I am grateful to all of them. I do not think that it is even possible to adequately thank all of them, but I want to express my gratitude to the most important ones here.

This thesis has been carried out at the Institute of Technical Informatics at the Graz University of Technology in close cooperation with our industrial partner NXP Semiconductors Austria GmbH. I want to thank Dr. Ulrich Neffe, my industrial partner's advisor, for integrating me into the project at NXP. I want to thank my supervisor, Professor Kay Römer, for guiding me through my Ph.D. studies by raising critical questions, giving detailed feedback, and not losing the big picture. A big shout-out goes to my advisor and mentor, Dr. Christian Steger, who always supported me in topics such as publications and conferences. He challenged me with research topics and always had an open ear for everyday issues. Additionally, I want to sincerely thank Professor Damian Dalton for serving as a second adviser on my thesis. Furthermore, I want to thank the Austrian Research Promotion Agency (FFG) for facilitating my Ph.D. studies.

I also want to say a big ”thank you” to all my colleagues. I was lucky enough to be a member of, in my opinion, the best office at the institute during my Ph.D. studies. Thank you, Michael Spörk, Rainer Hofmann, Alexander Rech, and Martin Erb. You always cheered me up and made me laugh when there was no reason to. Thank you also for the well-spent time at the many ”workshop” sessions throughout the last years. But my thanks do not end at our office's doorstep. Another big shout-out goes to my colleagues Michael Krisper, with whom I had many inspirational sessions and the pleasure to work on two joint publications, and Jürgen Dobaj, with whom I had many interesting technical discussions.

Eventually, I want to thank the most important persons in my life. First of all, my gratitude goes to my parents Franz and Bettina. Without them, I would not have been able to study at all. They always supported and motivated me throughout my whole life, before, during, and after my Ph.D. studies. I'm also thankful for my grandparents Dorothea and Ferdinand. Thank you for always inspiring me and pointing me in the right direction. My greatest thanks, however, go to my wife, Lisa. Thank you for always getting my back, supporting me, motivating me, listening to all my doubts and worries, and making my life so much better!

Graz, October 2020
Lukas Alexander Gressl

Abstract

The advent of the Internet of Things (IoT) and Cyber-Physical Systems (CPS) enabled a new class of connected, smart, and interactive devices. IoT devices and CPS are used in various contexts and often have access to critical information in both the digital and physical worlds. Due to their continuous connectivity to more extensive networks, such as the Internet, and their access to valuable and often private information, security attackers are likely to attack them. Engineers integrate these systems into both the industry and daily used consumer devices. Hence, CPS and IoT devices add yet another attack surface to their respective field of application, posing additional potential threats. These threats make a further consideration of security vulnerabilities necessary. Designers best perform this consideration during the initial design of IoT devices and CPS. Due to their resource-constrained nature, designing secure IoT devices and CPS poses a complex task, as various selectable hardware components and task implementation alternatives must be considered. Researchers proposed a range of automatic design tools to support system designers to find the optimal hardware selection and task implementations. However, these tools usually consider performance and power constraints but do not examine the designed systems' security. Hence, they offer a limited way of modeling attack scenarios targeting the system under design.

In this thesis, we propose a new approach to integrating security constraints into the design space exploration (DSE) of IoT devices, CPS, and embedded systems in general. With our Security aware DSE (SaDSE) framework, we aim to close the gap of missing DSE tools capable of modeling attack scenarios during the early phase of system design. Our framework offers the designer the possibility to model security constraints from the perspective of the potential attackers. It allows the designer to model the probability of successful security attack steps and their impact, resulting in the likelihood of successfully executed security attacks and the security risk they impose. By integrating security mechanisms, the SaDSE framework allows the designer to model mitigation techniques, counteracting them. Furthermore, we integrated the consideration of secret keys used by the chosen cryptographic algorithms into the design process. With these additional inputs and definable security constraints, the framework presents system solutions, meeting performance, power, and security requirements. These system solutions are then used by the SaDSE framework to generate system simulations, offering the designer additional metrics on the system behavior in different environments. We demonstrate the SaDSE framework's capability using two use cases developed in close cooperation with our industry partner.

Kurzfassung

Das Internet der Dinge (IoT) und Cyber-Physische Systeme (CPS) ermöglichen eine neue Art von Geräten miteinander zu verbinden, diese intelligenter und interaktiver zu gestalten. IoT-Geräte und CPS werden in verschiedenen Kontexten verwendet und haben häufig Zugriff auf wichtige Informationen, sowohl in der digitalen, als auch in der physischen Welt. Aufgrund ihrer kontinuierlichen Verbindung zu umfangreichen Netzwerken, wie dem Internet, und ihres Zugriffs auf wertvolle und private Informationen sind sie oft die Zielscheibe für Sicherheitsangriffe. Diese Systeme sind sowohl in die Industrie als auch in die täglich verwendeten Verbrauchsgegenständen integriert. Daher eröffnen IoT-Geräte und CPS in ihrem jeweiligen Anwendungsbereich eine weitere Sicherheitslücke, und stellen somit ein zusätzliches Risiko dar. Diese Sicherheitslücken sollen von den Designern beim ersten Entwurf der IoT-Geräte und CPS berücksichtigt werden. Aufgrund ihrer ressourcenbeschränkten Natur stellt das Entwerfen sicherer IoT-Geräte und CPS eine komplexe Aufgabe dar, da eine Vielzahl an Hardwarekomponenten und verschiedene Möglichkeiten zur Aufgabenimplementierung betrachtet werden müssen. Zur Unterstützung bei der Auswahl der optimalen Hardware und Aufgabenimplementierung gibt es schon eine Reihe automatischer Entwurfswerkzeuge. Diese Tools berücksichtigen normalerweise nur die Laufzeit und den Energieverbrauch der entworfenen Systeme, untersuchen jedoch nicht deren Sicherheit. Somit werden Angriffsszenarien, die auf das entworfene System abzielen, nur in eingeschränkter Form modelliert.

In dieser Arbeit wird ein neuartiger Ansatz zur Integration von Sicherheitsbeschränkungen in die Design Space Exploration (DSE) von IoT-Geräten, CPS und eingebetteten Systemen im Allgemeinen vorgeschlagen. Mit dem entwickelten SaDSE-Framework (Security Aware DSE) wird die Lücke fehlender DSE-Tools, mit denen Angriffsszenarien in der frühen Phase des Systemdesigns modelliert werden können, geschlossen. Das SaDSE-Framework erlaubt dem Designer, Sicherheitsbeschränkungen aus der Sicht potenzieller Angreifer zu modellieren. Der Designer kann die Wahrscheinlichkeit einzelner Schritte erfolgreicher Sicherheitsangriffe modellieren. Daraus können deren Auswirkungen sowie ihr Sicherheitsrisiko berechnet werden. Durch die Integration von Sicherheitsmechanismen ermöglicht das SaDSE-Framework dem Designer, Techniken zur Abschwächung der Bedrohungen zu modellieren die diesen Attacken entgegenwirken. Darüber hinaus wurde die Berücksichtigung geheimer Schlüssel, die von den ausgewählten kryptografischen Algorithmen verwendet werden, in den Entwurfsprozess integriert. Mit diesen zusätzlichen Eingaben und definierbaren Sicherheitsbeschränkungen bietet das Framework Systemlösungen an, die die Anforderungen an Laufzeit, Energieverbrauch und Sicherheit erfüllen. Diese Systemlösungen werden dann vom SaDSE-Framework verwendet, um Systemsimulationen zu generieren, die dem Designer zusätzliche Metriken zum Systemverhalten in verschiedenen Umgebungen bieten. Die Leistungsfähigkeit des SaDSE-Frameworks wird in der Arbeit anhand von zwei Anwendungsfällen, die in enger Zusammenarbeit mit unserem Industriepartner entwickelt wurden, demonstriert.

Contents

List of Figures

List of Tables

Acronyms

AES	Advanced Encryption Standard.
BAG	Bayesian Attack Graph.
BLE	Bluetooth Low Energy.
CAN	Controller Area Network.
CC	Common Criteria.
CP	Constraint Programming.
CPS	Cyber-Physical System.
CPT	Conditional Probability Table.
CVSS	Common Vulnerability Scoring System.
DES	Data Encryption Standard.
DeSyDe	**De**sign space exploration for **Sy**stem **De**sign tool.
DSE	Design Space Exploration.
EAL	Evaluation Assurance Level.
ECC	Elliptic-Curve Cryptography.
ECDSA	Elliptic Curve Digital Signature Algorithm.
HSM	Hardware Secure Module.
I2C	Inter-Integrated Circuit - communication bus.
IoT	Internet of Things.
JDT	Joint Distribution Table.
KDF	Key Derivation Function.
LOS	Line of sight.
MAC	Message Authentication Code.
MCU	Microcontroller Unit.
MSAG	Mapping- and scheduling-aware graph.
PP	Protection Profile.

RSA	Rivest-Shamir-Adleman.
SaDSE	Security aware Design Space Exploration.
SDFG	Synchronous data flow graph.
SE	Secure Element.
SPI	Serial Peripheral Interface - communication bus.
ST	Security Target.
TDMA	Time divions multiple access.
TLM	Transaction Layer Modeling.
ToE	Target of Evaluation.
UWB	Ultra Wideband.
W-LAN	Wireless Local Area Network.
WCCT	Worst Case Communication Time.
WCET	Worst Case Execution Time.
WiFi	Wireless Fidelity.
xls	File extension of Excel files.
XML	Extensible Markup Language.

1 Introduction

This chapter motivates why cyber-security must be considered already in the early phase of the design of Internet of Things (IoT) devices and Cyber-Physical Systems (CPS) and embedded systems in general. It describes the investigated research questions and summarizes the contributions of this thesis. Furthermore, it outlines the structure of the thesis.

1.1 Motivation

The number of IoT devices and CPS has been rapidly increasing in the last decade. Depending on the source of information, the number of IoT devices used by the year 2020 ranges from 26 billion to 50 billion devices [4,5]. For the year 2025, researchers estimate their number to reach 75 billion devices [6]. These IoT devices are a subclass of embedded systems and are usually equipped with sensors and connect to a backend system, such as cloud services. Regarding the term IoT, a clear definition is difficult to grasp, as researchers either describe them from the *Internet-* or the *Thing-*based perspective, depending on what attributes they want to emphasize [7]. In this thesis we stick with the definition coined by the *Information Society* (INFSO), who defines the IoT to be a

> *... a world-wide network of interconnected objects uniquely addressable, based on standard communication protocols [8]*

This definition emphasizes on the *Internet*-based perspective with its continuous connection of IoT devices. With their equipped sensors, they are capable of collecting data, which they can then transmit via their connection interfaces. Depending on the nature of the IoT device, the collected data might contain private information, such as human heartbeat, body temperature, and others [9]. Considering their vast number and their field of application, one can argue that IoT devices infiltrated our daily lives on a huge scale. The introduction of CPS into the industry heralded the next industrial revolution, called Industry 4.0 [10]. Similar to IoT devices, CPS consist of microcontrollers, integrated sensors, and actuators that allow interaction with the physical world. Edward Lee defined them to be

> *... integrations of computation and physical processes. Embedded computers and networks monitor and control the physical processes, usually with feedback loops where physical processes affect computations and vice versa. [11, p. 1]*

CPS process the sensed data and send it to other CPS or larger systems via communication interfaces [12]. In their field of application, CPS sense and process critical data, often not intended to be shared publicly. Their respective definitions show that the term IoT emphasizes the devices' connections to the Internet, whereas the term CPS highlights the systems' impact on physical parameters. From a security perspective, one must notice the large attack surface of IoT devices due to their permanent connection to the Internet, and the security risk induced by CPS, due to their influence on their physical surrounding.

With their advances into the customer market and the industry, IoT devices and CPS play an increasingly important role. Both the vast area of their application and their access to critical data make them attractive targets for security attackers. Numerous security incidents affecting IoT devices have been reported in recent years [13, 14]. Considering the more prominent incidents on critical infrastructure, one must mention the notorious *Stuxnet* attack, in which cyber-security criminals successfully attacked an Iranian atomic power plant in 2010 [15], but also the Maroochy water breach reported in the year 2007 [16]. A. Humayed et al. list various other security incidents considering CPS used in the context of industrial control systems, smart grids, medical devices, and others [17]. Depending on the application domain in which one uses IoT devices or CPS, successfully conducted cyber-security attacks can have a financial impact, or even cost human lives. Hence, the security of such systems is of utmost importance.

However, as described by A. Humayed et al., especially CPS suffer from several security vulnerabilities that originate from different sources. A dominant "security by obscurity" approach taken by the initial designs of CPS, their increasing connectivity among each other and with the Internet, and the heterogeneity of their used components are the most prominent sources of vulnerabilities [17]. Considering IoT devices, the lack of security considerations stems from a short time to market, cost reduction during the design and development phase, or from the emphasis on the device's functionality in favor of its security.

The previous paragraphs highlight how embedded systems in the form of IoT devices and CPS have infiltrated various application areas and how they operate on critical data. Hence, their lack of security has a significant impact on the applications that employ them. Considering the security of embedded systems in general, most of their vulnerabilities stem from bad design choices and architectural decisions. Furthermore, the later mitigation of such fundamental shortcomings becomes increasingly difficult, as many of them only cover up such vulnerabilities but cannot fix the underlying causes [18]. Hence, the integration of cyber-security aspects during the initial design phase is vital to develop secure embedded systems. Security is not the only requirement to which embedded systems must adhere. Considering their integration into more complex systems, they often must conform to timing and power constraints [19]. These constraints are especially crucial for IoT devices and CPS. The various constraints challenge the designers of such systems to find the best possible selection of system components and the optimal task allocation. Besides satisfying all constraints, the system's design must also optimize distinct metrics in many cases. Such optimization goals might concern the system's performance, power consumption, costs, or other metrics. Before describing the design process of embedded systems, some basic terms should be introduced.

Requirements: specify conditions or capabilities a product must meet. Thereby, it represents a contract between the stakeholders and the designers. A requirement can be functional or non-functional. Functional requirements describe a system's anticipated behavior. Non-functional requirements do not concern a system's behavior but describe other metrics obligate to the system [20].

Constraints: are assertions representing a restriction on a distinct metric of the system under design, such as timing, power consumption, or others. These constraints derive from the requirements formulated by the system designers and the stakeholders [21].

Security mechanisms: describe technical tools and methods implementing security services. These mechanisms might operate in cooperation with other security mechanisms or on their own. A system uses these mechanisms to secure certain attributes, such as confidentiality, authenticity, or others. Examples for such mechanisms are cryptographic algorithms, digital certificates, hardware protected areas within the microcontroller to execute secure code, secured memory areas, and many more [22].

The life cycle of embedded systems, in general, spans several phases, ranging from the early concept definition of functional and non-functional requirements, over the development of the approved design, to the final system integration and product support. Figure 1.1 shows these development phases with a focus on the cost-effectiveness of decisions and the commitment costs during the distinct phases. Commitment costs are costs that project managers commit to spending in the future. During the first phase of the system design, designers only spend 20% of the overall project costs. However, in this phase, 80% of the overall costs are committed. This early decision also has the impact that design modifications in later project stages have a significant impact on the overall product costs and its time-to-market. This insight shows that the more decisions designers take in the early design phases, the faster a product can be developed, and the cheaper the overall development becomes. Hence, designers must perform a rigorous analysis of all system requirements [1, 23].

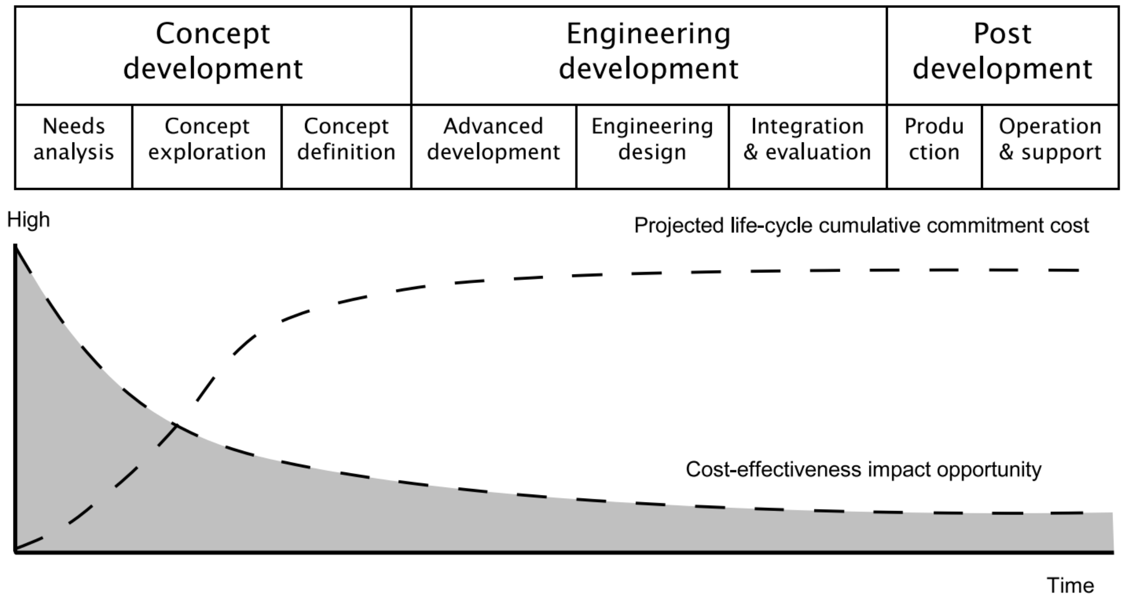


Figure 1.1: Embedded system engineering phases [1].

Based on the overall development of embedded systems, one can argue that the designer must perform these considerations of cyber-security aspects early on, due to several reasons. First, because a later integration increases the overall development costs. Second, because the additional usage of security mechanisms impacts other non-functional requirements, such as performance, power consumption, costs, chip area, and others [24, 25]. This effect is essential when considering time-critical systems as a later integration of these security mechanisms might impact the overall system design, and, hence, lead to unforeseen costs. Third, security mechanisms cannot be simply added to an existing product to increase its security. Instead, one must integrate security mechanisms right at the beginning of the system design [26].

The overall system design demands that the designers consider a multitude of requirements that often interfere with each other. Balancing these requirements poses a task too complex to be performed manually. Design Space Exploration (DSE) tools have been developed to aid system designers in this task. These DSE tools are capable of finding the optimal task allocation and system partitioning based on multiple system constraints [27]. Considering security requirements, however, they cannot model security requirements holistically. Hence, we saw a gap in this area, which we aim at closing in this thesis. We developed a security-aware DSE framework which allows system designers to describe security requirements from the perspective of a potential attacker, and, based on these and other requirements, computes secure system design solutions using both an analytical and a simulation-based approach.

1.2 Problem Statement

As discussed in the previous section, the consideration of security requirements at the early system design phase benefits the whole product development by decreasing costs and development time while improving functionality. Existing DSE tools usually used during this initial phase show a lack of support for considering security requirements and their influence on the system partitioning and task allocation. These tools take a binary approach regarding the security of embedded systems. They do not allow the modeling of security requirements from the attacker's view on the system under design. The analysis of potential attack scenarios gives additional information about what types of attacks the system under design must be able to defend itself against. The attack scenario's description from the attacker's perspective contains valuable information. Its integration into the overall system design benefits the overall system's security. Hence, the attacker's perspective allows for better judgment on the criticality of distinct assets in the system [28]. Furthermore, existing tools either focus on specific details of the system or abstract and thereby disregard important information, such as secret key placement (e.g., the works of Stierand et al. [29] and Xie et al. [30]). Chapter 3 gives a detailed description of these tools. When considering the vast amount of possibilities of how one can increase the security of a system and its impact on system costs, performance, power consumption, and others, a holistic approach is necessary.

Introducing security constraints into automated DSE of embedded systems in a holistic way faces several challenges. First (i), the representation of the security requirements must be chosen in a way that is straight forward to model for system designers. This representation and the description of the security requirements must build on well-defined and widely used methods.

Second (ii), the security requirements must be represented in a way that one can integrate them into an automatic DSE approach for designing secure embedded systems. The DSE tool must reflect the influence of these security requirements on the other system constraints, such as performance, power consumption, and others.

Third (iii), the narrow consideration of single devices is often not enough when designing complex systems. Especially when considering the networking behavior of multiple devices in dynamic environments, such as localization systems, a simulation-based DSE approach is better suited. With these simulations, designers can investigate different environments, including their physical properties and communication channel attributes. Designers can use these simulations to analyze further metrics, such as network throughput, packet reception rate, and others, to determine how the designed systems perform in different, dynamically changing scenarios. Based on the simulated devices' performances, the designers can reassess the overall system design and either approve or reconsider their choices. Also, in a simulation-based DSE, the designers must consider security constraints and model the chosen security countermeasures. The countermeasures' impact on the devices' internal timings is essential, as these might influence network-related metrics.

1.2.1 Hypothesis and Research Questions

Based on the above challenges, we derive the following hypothesis and three related research questions. The contributions described in this thesis aim at answering these questions and solving the problems stated above.

Hypothesis: *The consideration of security constraints based on automated DSE for embedded systems allows us to find secure solutions at an early system design stage without neglecting other non-functional system requirements. These solutions can be used during simulation to assess their network-behavior in dynamic environments.*

Based on this hypothesis, the following research questions arose:

RQ1: *How to model security requirements in a holistic way to use them in DSE?*

RQ2: *How to derive and integrate security constraints usable during automatic DSE of embedded systems?*

RQ3: *How to use the design solutions proposed by the analytical DSE and assess their network-behavior in dynamic environments?*

1.2.2 Contributions

The thesis aims at answering these research questions by introducing the following contributions:

Contribution I: In this thesis, we investigate how to model security requirements for designing embedded systems from the perspective of potential attackers. Thereby, we use different modeling techniques to describe the attacker's options on compromising the system under design, including the risks of the distinct attack scenarios. Furthermore, we describe the security mechanisms mitigating these attacks, reflecting their impacts induced on costs, performance, power consumption, and other metrics, on the overall system. Although we developed this approach to design embedded systems, one can also leverage it to design secure computer systems in general. Within this thesis, we present how to use our approach for designing a secure sensor system, considering security attacks on multiple abstraction layers, including the design of the embedded sensors, but also the high-level design of the server platform hosting the analysis application. This contribution builds on papers [2,31–33].

Contribution II: We show how to integrate these security requirements into the design space exploration for secure embedded systems based on the *Security-aware Design Space Exploration* SaDSE tool. The SaDSE tool is based on a classic DSE framework and models the constraints using a constraint-based programming approach. We show how the SaDSE tool reflects the security requirements by calculating additional constraints for the solver. Thereby, we show how to use these constraints to formalize objective functions towards which the solver optimizes. We describe how the selection of the various security mechanisms influences system metrics, such as performance, power consumption, and costs. We introduce optimization approaches on how to speed up the calculation of secure solutions. In addition to modeling security requirements from the attacker's view, this thesis introduces the automatic consideration of secret key placement during the DSE. This integration aims to support system designers with their decision on what hardware components to use to store the secret keys used by the system. This contribution builds on papers [32, 34–37]. With the SaDSE tool, we present a proof-of-concept implementation. However, the approach presented here can also be integrated into other frameworks.

Contribution III: We use the solutions found during the analytical DSE to generate a system simulation model from them automatically. Thereby, the framework represents each found solution of the analytical phase as a device model within the simulation. Each device model consists of simulation models for the hardware components and tasks mapped to them. The framework generates the tasks as skeletons, offering the designers a structure to implement the task's functionality for the simulation. The generated device models reflect the characteristics of the represented solution, including the influences caused by the selected security mechanisms. The framework connects multiple devices using a network simulation. In this thesis, we use a network simulation developed with our project partner and use the generated simulation for analyzing network metrics, such as channel throughput, packet reception rate, and others. The tool offers the designers the possibility to assess the influence of the security mechanisms on these overall system metrics for each solution based on the simulation results. The designers can use these insights to approve or adapt the found solution. The framework generates the simulation in such a way that it is integrable into further network simulations. Furthermore, based on the simulation generated by the SaDSE tool, we show the further potential of optimizing distinct parts of the designed system. In this thesis, we use the generated simulation to optimize the communication between a secure element and a micro-controller adapting the underlying protocol. This contribution is described in papers [3, 38]. With this generation of a system simulation based on the analytical DSE, the presented framework introduces a novel approach to consider security requirements in the early design of embedded systems.

Figure 1.2 shows the traditional design flow for embedded systems. It shows the levels of the design flow on which the thesis' contributions are located. The system design starts on a high abstraction layer and adds additional details to the design while further proceeding in the design process. At the *System Level*, the designers capture the design of the product as an informal specification, describing the system requirements, the functional behavior, and the architectural components. Based on these descriptions, the next step in the system design flow is to add characteristics describing the implementation of single behavioral elements on distinct hardware components. The main goal at this *Architectural Level* is to find a selection of hardware components and a mapping of the behavioral elements on these components, which satisfies the system requirements described at the *System Level*. Based on the system partitioning and the functional mapping, system designers can investigate different scheduling schemes. Based on the solutions found on the *Architectural Level*, designers can set up co-simulations, which allow the integra-

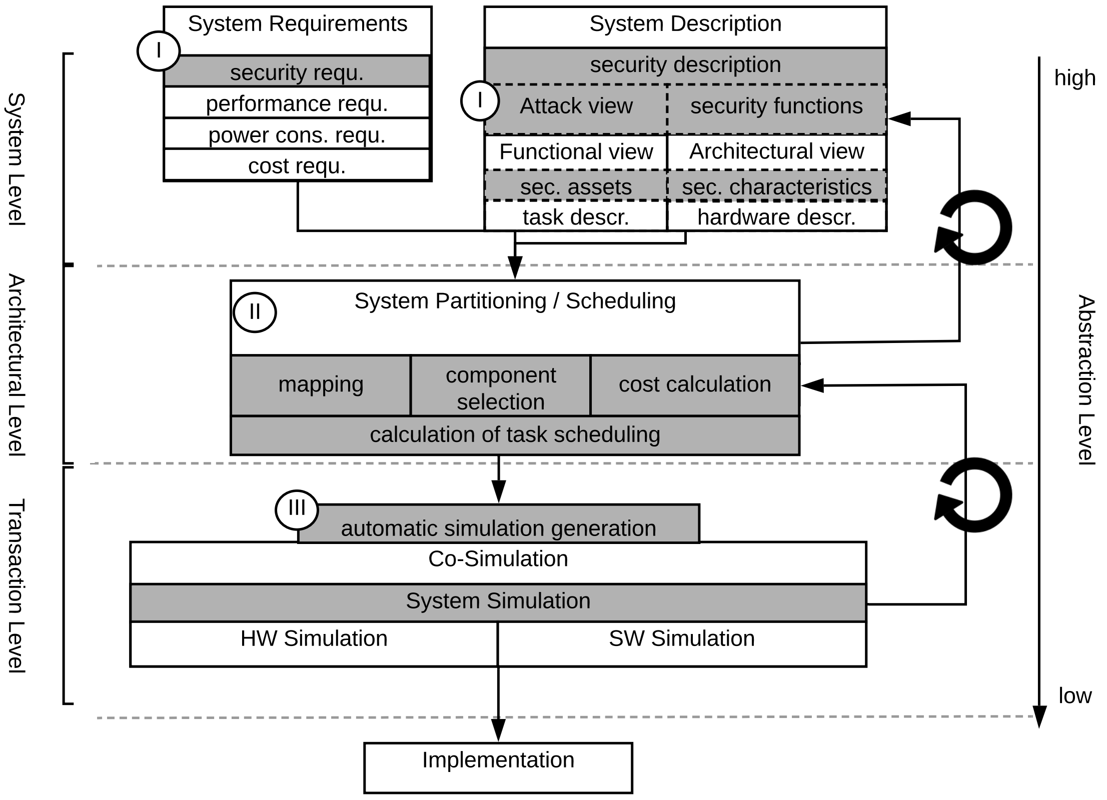


Figure 1.2: Traditional design flow for embedded systems and the phases where the contributions described in this thesis are located.

tion of both software and hardware simulations. System designers then use these simulations to enhance the accuracy of the system model further. At all these different abstraction levels, the designers can take a step back, alter the models of the preceding level, thereby refining the system design. The circular arrows indicate these refinements and the back-tracing arrows at distinct levels. Hence, the overall system design flow is an iterative process [19]. The contributions to this traditional design flow are numerically marked in Figure 1.2 with Roman numerals (I-III). The resulting modifications to this traditional design flow are highlighted in the gray boxes.

In contribution I, we introduced several security-related metrics into the system description and the system requirements. The system description was extended with a security description. We added a model of potential attack scenarios described from the attacker's perspective and security functions describing attack mitigation techniques available to the system under design. Furthermore, we introduced security assets into the functional view, which traditionally describes the system's functionality as a task graph. The architectural view was extended with security characteristics, expanding the traditional description of hardware components with security mechanisms available to them. The system requirements were extended with security requirements. The security requirements describe what security attacks the calculated solutions must be able to withstand.

Contribution II highlights the changes introduced to the calculation of the tasks' mapping to the hardware components. It also describes the selection of the hardware components, and the calculation of the costs. It describes the derivation of the security constraints from the security requirements and how the security mechanisms are selected and influence a solution's security, performance, power consumption, and other metrics.

Contribution III contains the automatic generation of the simulation based on the solutions found in the analytical DSE, thereby supporting the designers with an automatic transition to a simulation-based DSE. It shows how the security mechanisms that were chosen by the analytical DSE influence the overall system behavior in different settings.

1.3 Thesis Structure

In this first chapter, we argued why the integration of security requirements into the DSE of embedded systems is of importance. We discussed the problems occurring when modeling and integrating these requirements into an automatic DSE approach and gave an overview of the contributions with which we aim to solve them. The remainder of this thesis is structured into the following chapters:

Chapter 2 describes the thesis' foundations, introducing the basics of DSE for embedded systems and cyber-security, including their analysis methods.

Chapter 3 gives a detailed discussion of related research in the area of DSE with an emphasis on cyber-security. In this chapter, we describe how the framework presented here differs from state of the art DSE tools.

Chapter 4 presents the overall design of the SaDSE framework, showing how we describe the security requirements on the overall system design and how we derive the constraints from them. It describes the basic concepts with which the SaDSE framework was implemented. It focuses on the security constraints and their effect on the system partitioning, the mapping, and how the framework integrates other system constraints into the DSE.

Chapter 5 describes two use-cases with which the functionality of the SaDSE framework is evaluated. Based on the solutions found for these use-cases and their real-world implementations conducted with our project partner, we review our approach's soundness.

Chapter 6 concludes this thesis, discussing its limitations and giving an outlook on future work.

2 Background

This chapter introduces and describes the fundamental terms used in this thesis. It covers necessary information about DSE, covering its goals and methods. Furthermore, the chapter explains the basic concepts of cyber-security, discussing its basics, and prominent assessment methods.

2.1 Design Space Exploration

Design Space Exploration (DSE) describes the evaluation of possible design alternatives for a system under design. These system designs involve multiple metrics, such as performance, power, component costs, and many more. With the increasing complexity of systems, especially in embedded systems, a designers' task evaluating design alternatives has become increasingly complex. To support system designers in their evaluation, tools for an automatic DSE are used in both industry and scientific research. DSE tools usually follow the popular Y-chart approach. It builds on the co-design of application and platform. It also includes the mapping of the application to the platform [39]. Figure 2.1 shows this Y-chart approach. These tools allow for fast exploration of design alternatives in the initial system design phase [40]. DSE tools can generally be categorized into analytical and simulation-based approaches [41].

2.1.1 Analytical Design Space Exploration

The analytical DSE approach models the design problem abstractly and specifies it as a formal problem. This approach enables the application of analysis methods to search the design space for valuable solutions. The analytical DSE approach allows a first and fast filtering. However, the limitations of the analytical approach largely come from this abstraction, as the modeling of complex systems is a complex task. Especially when considering the dynamics of the environment in which the system under design is deployed, the analytical approach lacks the capability of reflecting this dynamism. An additional drawback of analytical DSE tools is the general worst-case estimation of system metrics such as execution times during the initial design phase. This estimation leads to an overall pessimistic system model [41, 42].

There exist numerous frameworks supporting this analytical DSE approach [43–46]. The analytical DSE must solve the system partitioning, the mapping problem, and the task scheduling. System partitioning describes the selection of the hardware components for the platform. These hardware components represent processing elements and physical communication channels, such as bus systems or connection-less channels [19]. Processing elements come in the form of general-purpose processors, application-specific instruction-set processors, application-specific integrated

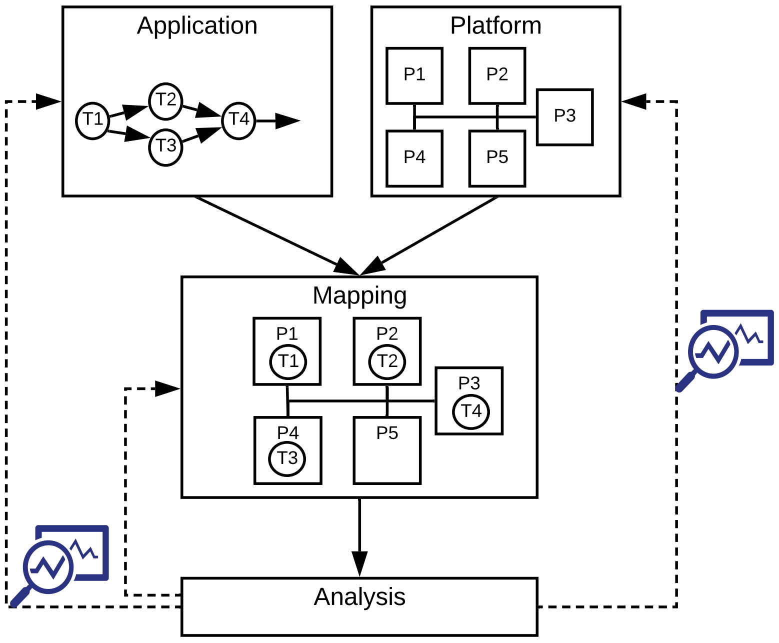


Figure 2.1: Y-chart based DSE

circuits, and many others. Bus systems come in the form of Inter-Integrated Circuits (I2C), Serial Peripheral Interface (SPI), etc. Common contact-less channels for embedded systems are Near Field Communication (NFC), Wireless Local Area Networks (W-LAN), Bluetooth Low Energy (BLE) and others [47].

The mapping problem describes the allocation of the tasks to the selected hardware components. The system partitioning, the task mapping, and the scheduling affect various system parameters, such as power consumption, timing behavior, and others. These influences must be modeled as functions within the DSE tool. Based on defined requirements considering the system's performance, power consumption, and other attributes, the DSE tool finds the valid solutions among the system design alternatives. Most DSE tools also find the solution(s) with one or multiple optimal attributes. This approach turns the DSE problem into a single-/multi-objective optimization problem. There exist various approaches on how to solve these problems. The most prominent methods use heuristic algorithms, or special constraint programming approaches [19,48–51].

2.1.2 Simulation-based Design Space Exploration

In contrast to the analytical DSE approach, the simulation-based DSE is capable of reflecting the dynamic changes of the environment in which the system under design is deployed. System designers use the simulation-based approach to understand how the designed system behaves in various use cases, coming with different environments [41,42]. There exists a range of simulation-based DSE frameworks [52–54]. Simulation-based DSE generally also follows the Y-chart model, similar to the analytical DSE. However, instead of modeling the DSE problem as an abstract model, simulation-based DSE implements the system's behavior and its platform components within a simulation environment. Strictly separating the behavior from the platform implementation allows an independent mapping between tasks and platform components. This mapping defines the tasks' attributes, such as timing behavior, power consumption, and others. Simulation-based DSE allows the iterative refinement of the system model. Compared to the analytical DSE, the simulation-based approach comes with a more complex implementation. However, it allows an evaluation of the system in dynamically changing environments and a steady refinement of the overall model [52]. Such implementations build on simulation environments, such as SystemC, SystemVerilog, and others [55, 56].

2.1.2.1 SystemC Simulation Engine

SystemC is a well-established simulation engine both within research and the industry. Its simulation engine is completely implemented in C++, which makes it usable in different execution environments. The key feature of SystemC is its capability of modeling systems on various abstraction levels. They range from capturing the system requirements (highest abstraction), over the partitioning of the system into hardware and software components, down to the register transfer level (lowest abstraction) [55]. SystemC uses a transaction layer modeling (TLM) approach for defining the interfaces between distinct models in the system simulation. This modeling approach allows the designers to independently refine single modules without changing their communication interfaces to other modules. This approach helps in tackling the problem imminent to all simulation engines: execution time versus simulation accuracy. The TLM approach lets the designers choose in what modules to increase the accuracy without changing the abstraction level of other modules. Therefore, feasible execution times of the system simulation can be maintained.

Its extendability and ability to model the system on multiple abstraction layers make SystemC a popular choice for the implementation of hardware/software co-design for embedded systems. The easy integration of additional C++ libraries and the usage of C modules allows a straight-forward implementation of hardware/software partitionings [57].

2.1.3 Combined Design Space Exploration

Multiple research teams have been driving the idea of combining both analytical and simulation-based DSE to integrate the benefits of both approaches [41, 58, 59]. Several approaches use the output of the simulation-based DSE to feed different configurations to an analytical approach. This second step then finds the optimal configurations using analytical DSE [58,59]. Other approaches, such as the framework of Fernando Herrera et al. [41], perform an analytical DSE as a first step, filtering valuable solutions from which their framework creates simulation models. These models are then tested within various environments.

2.2 Cyber-securty

The terms *cyber-security* and *information security* are used interchangeably in the literature. The International Telecommunications Union defines cyber-security as a collection of various security tools, policies, concepts, but also guidelines, to protect the assets of both organizations and users against relevant security risks [60]. The security assets that need protection and the mechanisms how to protect them are defined in various standards [61].

2.2.1 Security Properties

The main purpose of cyber-security is the protection of user and company assets against imminent security threats. To determine the necessity of this protection, these assets must be made tangible, and their value must be determined. An asset's security properties determine against what threats it must be secured [60, 62–65]. The most common security properties are:

Confidentiality: The protected asset is only readable by intended recipients. It is not readable by any unauthorized entities. The protection of the asset must be considered during transmission, but also when it is stored.

Integrity: The protected asset reaches the intended recipient with the same content as sent by the sender. Any modification performed by an unauthorized entity must be noticeable. Modifications mainly come in the form of alteration or corruption of the asset. Unauthorized participants must not alter the asset's content during transmission and storage.

Authenticity: The protected asset is proven to originate from the sender it claims to be its originator. The recipient checks the asset's authenticity, and thereby also its integrity. Depending on what standard, authentication is a part of data integrity [64].

Availability: The protected asset must be reachable by the intended participants whenever they need it. Any prevention of this reachability is considered a successful attack.

2.2.2 Security Threats and Attacks

The terms of security threats and attacks are often used ambiguously when considering various publications that discuss information security in general. Researchers often use these terms interchangeably. This section introduces the most prominent definitions of security threats and attacks. Threats, in general, are actions that take advantage of vulnerabilities within a system [66]. Security threats aim at breaking the security properties of an asset. There exist various models for describing security threats and how they are categorized [67]. In this thesis, we use the most prominent and widest used threat model: The *STRIDE* model [68]. This model categorizes security threats into the following classes:

Spoofing identity: The successful identification as some other entity by falsifying information. This threat aims at breaking an asset's authenticity.

Tampering with data: Intentional and malicious data modification during storage or transmission. Tampering aims at breaking the asset's integrity.

Repudiation: The break of the association between actions/changes and a unique individual. Thereby, repudiation breaks or circumvents the proof of the asset's integrity and its origin.

Information disclosure: The reading of information one is not authorized to read. Information disclosure aims at breaking an asset's confidentiality [69].

Denial of service: The denial of a service to a valid user. This threat breaks an asset's availability.

Elevation of privilege: The unauthorized gain of privileges, such as reading and writing of assets which are normally protected from these operations being performed.

These terms listed in the *STRIDE* model are also referred to in various other information security standards [70–73]. The threats described here result from potential security attacks. Security attacks focus on harming a system and disrupting its intended operation. Attackers use special tools and techniques to find and exploit vulnerabilities in the system to achieve their intended goals. Each security attack comes with certain costs, expressing the attacker's expertise and available resources. Security attacks can come in many forms, ranging from access attacks, over attacks on privacy, to physical attacks [74].

Papp et al. describe a range of documented attacks, vulnerabilities, and threats in the area of embedded system security. The authors build their description on the Common Vulnerability and Exposures database, which provides a comprehensive list of known vulnerabilities with additional standardized identifiers. Attacks on embedded systems come in the form of hijacking the control flow, eavesdropping, packet injection, infection of the software with malicious code, and many more. These attacks lead to integrity violations, information leakage, unauthorized access, malicious code execution, and other unintended effects [75].

2.2.3 Security Mechanisms

Security attacks imply threats to a system by exploiting its vulnerabilities. These vulnerabilities allow an attacker to intrude a system which allows for conducting harmful actions to valuable assets, breaking their security properties [62]. Security mechanisms are methods used by the system to prevent these exploits and, thereby, to mitigate security attacks. Thereby, these mechanisms prevent the breaking of the asset's properties. There exists a vast number of security mechanisms. This section only lists mechanisms which are used throughout the thesis:

Cryptographic primitives: Cryptographic algorithms are, in general, put into two categories, symmetric and asymmetric cryptography. The key difference between these two primitives is the used secret keys [76]. Both primitives build on the secrecy of the used private key. Symmetric cryptography requires both sender and receiver to use the same private (secret) key. Prominent and widely used symmetric cryptography algorithms are the Advanced Encryption Standard (AES) and its predecessor, the Data Encryption Standard (DES) [77, 78]. Said algorithms are either used for encryption, protecting an asset's confidentiality or for authentication, protecting is integrity/authenticity. Authentication is performed by Message Authentication Codes (MACs) [79] which rely on cryptographic hash functions (involving a secret key) [80], or symmetric cryptographic algorithms [81].

Asymmetric cryptography relies on key pairs for the sender and receiver. A key pair consists of a public key and a private key. Depending on the used algorithm, the private key is used for decryption and signing; the public key is used for encryption and verification of a given plain text. As its name suggests, the public key can be shared with everyone; the private key must be kept secret to guarantee the protection provided by the algorithm. Prominent examples for asymmetric cryptographic algorithms are the Rivest-Shamir-Adleman [82] (RSA) and the Elliptic-Curve Cryptography [83] (ECC). Asymmetric cryptography is also used providing authenticity in the form of digital signatures, such as RSA signatures [84] or the Elliptic Curve Digital Signature Algorithm [85] (ECDSA).

The implementation of these algorithms is either performed in software or hardware. Both implementations come with distinct advantages and disadvantages, considering performance, power consumption, chip area size, and others [86]. Furthermore, implementations of cryptographic primitives might be susceptible to various side-channel attacks. These side channels unintentionally reveal information regarding the cryptographic process. This revelation can lead to the disclosure of the used secret keys, rendering the whole cryptographic process insecure [87].

Cryptographic hash functions form another frequently used method in cryptography. Many security functions, such as signatures, rely on the usage of hash values. Hash functions are mathematical functions mapping a value (m) of arbitrary length to a fixed-length value $h(m) = m'$. Hash functions must fulfill several properties: Collision resistance, pre-image resistance, and second pre-image resistance. Collision resistance means the infeasibility of two values having the same hash value ($h(m1) = h(m2)$). Pre-image resistance refers to the infeasibility of finding m when given $h(m)$. Second pre-image resistance means that given a value $m1$ it must be infeasible to find a second value $m2$ where $h(m1) = h(m2)$ [88]. Hash functions are, e.g., used for cryptographic signatures, such as hash-based message authentication codes which calculate the hash value of the message and then sign the hash value instead of the whole message. This approach saves computation time, and because of the hash's second pre-image resistance, it can be assumed the signed hash originates from the received message [89].

Secure hardware: Many cryptographic primitives are fully or partially implemented in hardware [90,91]. In addition to potential side-channel attacks, security-relevant hardware is also susceptible to physical attacks. Tamper-resistant hardware components provide certain mechanisms to mitigate these attacks [92]. They support secure code execution and physical separation of security-relevant tasks [93], for example the *TrustZone* design established by *Arm* [94]. Additionally, tamper-resistant hardware often supports tamper-proof storage that can be used for storing cryptographic keys. These secure memory regions are protected against unauthorized access using firewalls controlling the data transfer over the bus-system [95].

Key derivation: Key derivation functions (KDF) are hash-based methods that allow the derivation of secret keys from a secret value, e.g., password, master key, etc. They are usually used to generate ephemeral keys from a secret master key. A common attack to disclose secret keys used for encryption and authentication builds on differential cryptanalysis [96]. By periodically changing the used encryption keys, this attack can be mitigated. Using a KDF allows renewing the session key when only possessing a single shared secret key. The fundamental requirement of a KDF is to build on a strong cryptographic hash function. This ensures that even the disclosure of the session key does not reveal any information on the secret master key, it has been derived from [97, 98].

2.3 Security Assessment Methods

The assessment of security assets and threats is far developed in both industry and academia. This section lists the most important security assessment methods relevant to this thesis.

Common Vulnerability Scoring System: The NIST Common Vulnerability Scoring System (CVSS) is a method to capture principal characteristics of security vulnerabilities. It uses a numerical representation to reflect a vulnerability's severity, which helps organizations to prioritize their vulnerability management. The CVSS calculations consist of three metrics: the base, temporal, and environmental metrics. The base metric rates the quality, which is intrinsic to a vulnerability. The temporal metric describes characteristics that evolve over the lifetime of a given vulnerability. The environmental metric characterizes vulnerabilities due to the distinct implementation environment. The National Vulnerability Database which stores standardized vulnerabilities uses the CVSS [99, 100].

Attack graphs: Attack graphs model attack scenarios and are a valuable method to assess security threats of systems. They are widely used in both science and industry and have been rapidly adopted for various application areas, such as network security analysis [101]. These models describe attack steps as distinct nodes within the graph. The nodes are connected with edges representing the dependencies between the single steps. The attack graph represents different attack paths a potential attacker can take to harm the system's assets. As these graphs are represented as Bayesian networks, they must be acyclic. Using these attack graphs, designers estimate the probability of certain attack scenarios to be successfully performed by a potential attacker [102–104]. Attack graphs are also used in security risk management [105].

Risk assessment: Risk assessment and management are central parts of the security evaluation for systems and organizations. The main purpose of security risk assessment is the reduction of uncertainty induced by potential threats. Risk is the product of the probability with which an event might occur and its impact. In most cases, the impact constitutes a financial loss. In certain systems, the occurring event's impact can also lead to the loss of lives [106]. Novel approaches in risk assessment consider the usage of risk trees. These trees model the security risks induced by potential attack scenarios. Thereby, they allow the system designers to assess the severity of individual attack paths based on their risks on the overall system [107].

Common Criteria: The Common Criteria (CC) is a standardized method in the industry to evaluate the security measures of secure products. These products are named *Target(s) of Evaluation* (ToE). The main purpose of the CC is to support the evaluation facilities with guidelines on how to assess the security functions implemented by the ToE. This assessment covers what security functions have been implemented, as well as the soundness of the implementation. What security function must be implemented by the ToE is determined by its Security Target (ST). The soundness of the security measures implementing the security functions is stated by the Evaluation Assurance Level (EAL). These levels scale from EAL 1 (lowest) to EAL 7 (highest). The higher the EAL, the more thorough the facility must evaluate the ToE. The Protection Profile (PP) allows distinct groups and communities to capture their security requirements. These PP are used to group multiple STs. Each ST in a PP inherits the PP's security requirements [108].

3 Related Work

The research community has been studying the integration of security requirements into analytical DSE and the combination of analytical and simulation-based DSE approaches for several years. This chapter gives an overview of various projects in these research fields and gives a detailed explanation of their differences to the research presented in this thesis.

The DSE of safety-critical systems considers a problem very similar to security-driven DSE approaches. Similar to security constraints, safety constraints restrict the mapping of critical tasks to especially enhanced components or enforce a redundant implementation of specific components. Thereby, safety-related constraints often contradict performance and power requirements to which the designed system must adhere. DSE frameworks in this realm are in high demand for the industry and especially in the automotive area. Functional safety aspects generally consider the reliability of certain system components. A system's reliability can be improved by adding redundancy and variation to system components, posing a risk to the user's safety. Safety constraints are usually expressed by certain levels to which the selected components must adhere to. These levels are usually defined by large manufacturers, leaving a small scope for decision-making. Therefore, projects integrating safety into the DSE of systems are considered as examples of related approaches integrating other non-traditional requirements into the DSE of embedded systems. These projects solve constraints on traditional characteristics, such as performance and power consumption, but only integrate the safety aspects based on safety levels. Compared to the integration of safety aspects into the DSE, the integration of security spans a wider design space to select solutions from [109–112].

3.1 DSE Tools considering Cyber-Security

The DSE of embedded systems is a well-studied field in both industry and academia. Various research projects have described classical DSE tools for embedded systems that consider the power and performance of the individual implementation alternatives of the system's behavior on distinct components. Based on these two characteristics, they select the optimal system components, functional implementations, and scheduling strategy. Considering DSE tools integrating security requirements into the design space of embedded systems, various research publications have been presented in the last years. Research projects in this area can be generally categorized into approaches focusing on a single aspect of the system design, approaches abstractly considering the design space, and approaches focusing on the security aspects of network systems. The following sections describe related projects within these categories and their approaches to integrating security aspects into the DSE of embedded systems.

3.1.1 Abstract Approaches

Abstract approaches on the security constraint integration into the DSE of embedded systems focus on high-level descriptions of the design space. Research projects in this area focus on abstracting the security aspects of the embedded system design.

Stierand et al. presented their approach to integrating security requirements into the DSE of embedded systems in [29]. Their approach considers the characterization of security assets used by the system under design and the attackers' capabilities. In their model, the attacker's capabilities and the assets' properties match (e.g., an attacker capable of reading messages compromises the confidentiality of the message). The approach also considers hardware components with different security capabilities used to secure the assets. Their approach focuses on the channels used to transfer the security assets and to what architecture modules they are connected to. To satisfy the security of the defined assets, the DSE approach only allows the transmission of the assets between architecture modules capable of protecting them. Hence, the system partitioning is limited by the compulsory protection of security assets.

In [113], the authors present a DSE approach on designing secure processor architectures. The authors use a high-level analytical model to find practical designs considering the processor's security requirements and different trade-offs regarding the integrated security modules and their performance. The high-level model approach allows the early discovery of performance bottlenecks in the design. The authors' approach only considers the design of the security processors themselves, without any integration of potential attacks.

The approach proposed by Zheng et al. [114] focuses on the control performance and the schedulability of CPS under the consideration of security requirements. In their approach, the authors assume an attack model aiming at learning the states of the internal control tasks of a CPS. To learn a task's state, the attacker eavesdrops on the exchanged sensor messages within the CPS. Depending on the amount of successfully disclosed sensor messages, the attacker has a certain probability of guessing the control task's state. The more sensor messages the attacker is able to eavesdrop, the higher the possibility of learning the correct state. Encrypting the sensor messages mitigates the eavesdropping and, hence, decreases the learning possibility. This learning possibility is captured as a security level. The encryption lowers the overall control performance. The approach presented in [114] captures this problem and allows its optimization by altering the number of encrypted sensor messages. Thus, the approach allows the designers to find solutions adhering to a predefined minimum security level and control performance.

Lin et al. [115] describe a methodology on how to formalize security constraints for embedded systems and integrate them into DSE. Their methodology allows the mapping of distinct system functions to architecture components based on their security properties and the security services and mechanisms protecting them. The authors' approach also considers performance metrics of the chosen security services and mechanisms and specific risk values defined by the users. How their methodology is used in practice is described in two different security design use cases, both coming from the automotive industry. In [116], the authors show the usability of their design approach on the security enhancement of a controller area network (CAN). In [117], the authors show their approach applied to the design of a Time Division Multiple Access (TDMA) based real-time distributed systems.

Zhang et al. [118] consider the DSE of security-sensitive mixed-criticality real-time embedded systems. They consider the design of embedded systems given their functional and architectural description. The functionality is described as a task graph. Each task comes with a different execution time, depending on the processor it is implemented on. The system's security is captured as security criticality levels (SCL), ranging from SCL1 to SCL4. Depending on the SCL needed by the task and the SCL provided by the processor it is implemented on, both execution time and power consumption alter. The higher the SCL, the higher the execution time and power consumption. Using a DSE approach, the authors find an optimal system partitioning and task mapping considering the system's SCL, performance, and power consumption.

Information security modeling and threat analysis have attracted much interest in both research and industry in the last decades. To consider security measures in the design phase of products, various modeling tools, and languages, such as the Unified Modeling Language, integrate extensions allowing the modeling of security [119]. Such tools support the designers in reflecting, e.g., security protocols within the system's behavior. However, they do not consider the system's hardware architecture, nor the security's performance overhead, nor the attacker's view on the system under design. The framework described in this paper integrates security attacks and the architecture's security capabilities into the design flow, automatically proposing security operations.

The SysML-sec project [120] integrates cryptographic mechanisms during hardware/software partitioning based on predefined attack scenarios. Their project builds on the SysML-based description of embedded systems. In addition to SysML's design approach, SysML-sec allows the integration of security aspects to the overall system design in the form of attack scenarios and cryptographic mechanisms. The attack scenarios come in the form of textually described attack vectors, characterized by the security properties they aim to break. The cryptographic mechanisms are described with their performance and power costs, and what security properties they protect. Using these characterization approaches, the designers can capture the necessary security mechanisms and integrate them into the general system design.

Compared to the works described in this section, the approach taken by the SaDSE tool allows a more detailed description of the design space. Thereby, the framework still maintains a holistic view of the overall system under design. This balance is achieved by supporting the use of rule sets to describe the impact of the security requirements on the solutions produced by the tool.

3.1.2 Detailed Approaches

The detailed approaches focus on specific parts of the overall system design. Research projects in this area mainly focus on the security protocols and the design of secure communication of embedded systems.

Xie et al. consider the security for the signal packing problem of a CAN-based embedded sensor network. In their work, the authors describe the signal packing problem as an optimization problem in which the optimum between message payload sizes and MAC sizes. The MAC sizes chosen by the system influence the communication's security and affect the payload sizes chosen for the exchanged packets. The bigger the exchanged messages, the worse the communication performance becomes. These contradicting influences on security and performance make the optimal selection of MAC and payload sizes complex. The authors resolve this problem by employing a DSE of the available options [30, 121].

Lukasiewycz et al. [122] also consider the security-aware DSE of a CAN-based embedded system. In their work, the authors focus on the scaling effects of attacks on vehicle fleets sharing the same vehicle platform. The attacks build on compromising the headers of the messages transmitted via the CAN bus systems. Although the transmitted messages' payload is generally encrypted, the message's identifier must be exchanged unencrypted due to the used arbitration method. As the mapping between the payload and the identifier is fixed, compromising the identifier and, hence, its payload mapping can enable the attacker to reverse engineer the used vehicle platform. As these platforms are reused in many different vehicles, the attack can spread among whole vehicle fleets. To counteract this threat, the authors propose the obfuscation of these identifiers. As this obfuscation also influences the communication performance, the authors apply a DSE of different approaches to find the optimal obfuscation technique, which still satisfies the system's performance constraints.

In their work, Jiang et al. [123] focused on finding the optimum between the quality of service (QoS), quality of confidentiality (QoC), and the intrusion detection accuracy (IDA) of embedded systems. In their approach, the QoC depends on the number of rounds used by the encryption algorithms. The more rounds, the higher the confidentiality. The QoS depends on the execution time available to the general tasks. The intrusion detection is performed by dedicated tasks scheduled during the normal execution of the system's functionality. The higher the frequency of their scheduling, the higher the IDA. However, both the intrusion detection and the confidentiality's quality decreases the execution time available for the general tasks and, hence, decreases the QoS. The authors solve this problem by applying a DSE approach to find the optimal solution given the constraints on QoS, QoC, and IDA.

Hasan et al. [124] deal with the integration of security-surveillance tasks into an already existing task schedule on a multicore-system. These security-aware tasks are used to check the system for the intrusion of potential adversaries. As such, checks are time-consuming; the integration of those tasks can break the overall task schedule. As the system comprises real-time tasks, the system must guarantee their timely execution. Hence, the integration of the security-surveillance tasks must be performed so that the other tasks' timelines are not broken. To solve this problem, Hasan et al. use a DSE-based approach, solving the general schedulability of both real-time and security-surveillance tasks.

Kang [125] published a tool that supports system designers in their decisions regarding the security mitigation techniques used for a network system. The mitigation techniques get described by the designers in the form of security policies. Their approach supports both the evaluation of design candidates against a predefined system description and security policies and the enumeration of potential design candidates satisfying both policies and system descriptions. The design candidates are captured in the form of domain models stored in a library, extendable by the designers. Their approach allows both the verification of design candidates as well as finding valid candidates.

Compared to the works presented in this section, the SaDSE framework focuses on the design space spanned by the design of the entire system. Thereby, the SaDSE tool is capable of modeling certain aspects of the internal information exchange using communication buses but does not put its whole focus on the communication part of the system design.

3.1.3 Network Design Approaches

The works presented in this section focus on the security aspects of networked systems. Thereby, they consider security requirements both during design- and run-time.

In their work, Anderson et al. [126] present the NetKAT tool. This tool offers a network programming language that describes network topologies from both an architectural and implementational level. Thereby, the tool offers a semantical and mathematical description approach. The tool allows the designers to evaluate their networks under design against the mathematical model, including security requirements on the used protocols and performance metrics. Designers can check if their modeled networks fulfill these requirements using the NetKat tool.

Nelson et al. [127] present a network firewall analysis tool called Margrave. With their tool, the authors support designers in the firewall configuration of security blocking rules, security goals, the accuracy of security blocking rules, and others. The Margrave tool offers the designers to formalize specific security goals and checks whether a proposed configuration adheres to these goals. Furthermore, it also provides the designers with valid configurations given predefined security goals. Thus, the tool supports the designers in exploring the valid configurations with which a firewall can be configured.

With VeriFlow, Khurshid et al. [128] present a network verification tool usable during design and run time. This tool allows the users to verify performance and security requirements for dynamically changing networks. Furthermore, the VeriFlow tool supports the designers in finding optimal network setups to fulfill given requirements.

Compared to the works presented in this section, the SaDSE framework focuses on ensuring the confidentiality and authenticity of messages passed via potentially insecure channels. However, it does not consider the configuration of distinct access rights or securing a system against unauthorized access.

3.2 Security Attack Analysis Tools

The security attack analysis is a vast research area in which the modeling of possible attack scenarios is widely used. The frameworks presented in this area are mainly used to provide designers the possibility to consider potential attacks on newly designed systems.

Poolsappasit et al. [129] present a framework that is capable of solving the administrator's dilemma. The administrator's dilemma describes the cost-optimal selection of security services to mitigate distinct attack scenarios on a network system. To describe the attack scenarios, the framework offers the designers a model of them as Bayesian Network Attack Graphs. These graphs allow the capturing of the distinct attack steps the potential attacker can take. Also, the probabilities of successfully reaching these steps can be modeled. The security services lower these probabilities but induce implementation costs. Hence, the framework supports the designers to find the optimal selection of security services to mitigate potential attack scenarios.

In their work, Feng et al. [130] propose a novel security risk assessment model building on Bayesian Network-based Attack Graphs. They introduce a three-phase approach, consisting of a Bayesian Network development, a security risk assessment, and a vulnerability propagation analysis. The Bayesian Network-based Attack Graph structure is learned from a historical database in such a way that it optimally fits past security incidents. The graphs are then used to perform a security risk analysis fed by a real-time database that is concurrently updated with new incidents. Based on this analysis, the model allows performing a vulnerability propagation analysis and supports the designers in the creation of risk treatment plans.

Frigault and Wang [131] investigate the usability of Bayesian networks for modeling attack scenarios of potential attackers. In their work, the authors provide a fundamental explanation about the attack representation. They bind their modeling approach tightly to the CVSS, which supports the designers to fill the probabilities of the distinct attack steps within the Bayesian network.

Sun et al. [132] investigate the usability of attack graphs for modeling potential zero-day attacks. Zero-day attacks are generally attacks enabled by yet unknown vulnerabilities in a system. These attacks are modeled as distinct attack paths in an attack graph. The authors argue that a successful attack on a system is composed of both zero-day attacks and known vulnerabilities. The approach of Sun et al. allows the complete capturing of attack scenarios. The attack graphs build on a probabilistic representation provided by Bayesian networks. This representation allows the extension of the attack paths with evidence assigned to nodes where a vulnerability exploit was detected. Using this evidence, zero-day attacks can be learned, helping the security analysts to find the vulnerabilities in the system.

Ammann et al. [133] presented a straight-forward approach to the modeling of network attacks. In their work, the authors aim at simplifying the attack graph representation by omitting unnecessary information produced by classic representations. Their representation is based on the use of effective compression algorithms, which allows a more compact model of the attack scenarios. This representation, however, can be bloated to the traditional representation if needed. Hence, the approach of Ammann et al. supports designers with a concise model of potential attack scenarios.

Ray et al. [134] propose a novel intrusion detection system that builds on risk calculation performed during run time. Their approach builds on the reasonable assumption that attacks on a given system can be enumerated by assessing past attacks. Considering the potential attacker's session scope, probable attack sequences can be limited. Based on these sequences, the authors implemented an estimator to assess the overall attack probability. The estimator allows the intrusion detection to warn the system administrator on an imminent attack and, hence, allows its mitigation before the breach occurred. The authors showed that their approach, in addition to traditional intrusion detection mechanisms, has a great benefit for the overall system's security.

In [135], Phillips et al. present their approach to supporting the security of networks using attack graphs. Their approach uses attack graphs to model security risks on networks. These attack graphs are generated from three inputs: attack templates, configuration files, and profiles describing the potential attackers. The templates provide a generic representation of the attacks and their preconditions. These templates are configured with network characteristics. The attacker profile describes the attacker's capabilities. Based on these inputs, the attack graph with attack success probabilities is generated. The proposed approach supports network administrators evaluate the costs of attack paths and model potential future attack scenarios.

Similar to [135], also Sheyner et al. [136] propose the generation of attack graphs based on the description of attack scenarios and attacker properties. Based on the attack graphs produced by their approach, networks can be checked for stealthy intrusion attacks. Already implemented mechanisms do not detect these stealthy intrusion attacks. This consideration of stealthy attacks allows the network administrators to find the optimal locations where additional mechanisms are placed. Their approach allows the determination of a minimal set of attack steps whose mitigation would guarantee the intruder's failure.

Compared to the presented security attack analysis tools, the SaDSE integrates the security modeling techniques presented here into the automatic DSE of systems. Thus, while these tools do not consider security aspects in combination with other system constraints, such as performance, they provided us with an inspiration to integrate security constraints into DSE.

3.3 Analytical and Simulation-based DSE

The tools presented in this section focus on the integration of analytical and simulation-based DSE. Most of these tools perform an initial simulation of the system design and use the retrieved simulation results for an analytical optimization of the system parameters. Other tools start with an analytical DSE and use the resulting solutions for a simulation-based feasibility check.

In [41], Fernando Herrera and Ingo Sander describe an approach combining analytical- and simulation-based DSE. Their approach contains a first analytical DSE and a second simulation-based DSE for finding suitable design candidates for embedded systems. The design candidates proposed by the analytical part are considered to meet safety-relevant timing constraints. These candidates are then used in executable performance models that are stimulated by likely environment scenarios. Using the executable performance models, the most efficient design candidate is evaluated by the designers.

Fornaciari et al. [137] present a framework allowing the simulation-based exploration of potential memory architectures on a system level. The authors show how their approach can be used to find a near-optimal cache architecture configuration, varying the parameters on cache size, block size, and associativity. Their approach aims to prohibit the exhaustive exploration of all design variants by applying heuristic approaches known from the analytical DSE. Their approach builds on the dynamic profiling of memory references. These traces are obtained by capturing the software execution focusing on the transition activity on system-level buses.

Künzli et al. [58] present a new method that allows designers to mix simulation-based and analytical DSE. With their approach, they reduce the execution time of simulation-based approaches. These long run-times are the major drawback of simulation-based approaches. To realize this hybrid approach, the authors propose a component-based design. This design allows the integration of simulation-based and formal models and allows the reuse of the components. Thus, the mixed approach can be utilized to improve the performance of the simulation-based exploration by gradually exchanging simulation with formal components.

In their work, Lahiri et al. [59] present a combined DSE approach used for the performance analysis of communication architecture designs. The approach focuses on system-level performance analysis. The authors' approach fills a gap in existing frameworks. Existing tools are either too slow because they simulate the complete system or are not accurate enough because they only perform static analysis. The approach of Lahiri et al. widely applicable, as it uses a very general communication architecture that allows the designers to model their customized protocols. Furthermore, the approach allows the modeling of various dynamic effects on communication. Based on the modeled communication architecture, the designers trace the communication. These traces are then used to analyze the performance of the communication architecture.

Kempf et al. [138] present a combined analytical and simulation-based DSE approach for modeling software-defined radios. Thereby, the authors present a solution for solving the very complex task of designing these radios. Their key contribution is the pre-simulation of the mathematical analysis based on synchronous data flow graphs. Their approach seamlessly integrates into an electronic system level based simulation framework. Hence, the approach enables the transition of the pure mathematical analysis to the overall system simulation.

The main difference between the tools presented in this section and the SaDSE framework is the transfer of the security constraints' influence on the resulting solutions into the system simulation. Thus, the designers can evaluate the impact of the security aspects on the overall system's behavior.

3.4 Differentiation

Considering the related projects regarding the security-driven DSE, the security attack modeling, and the combined DSE approaches, the SaDSE framework presented here differs in various aspects. This differentiation towards the related approaches is described in the following sections.

3.4.1 Security-driven DSE

The SaDSE approach differs from other security-driven DSE tools by integrating the attacker's perspective directly into the model describing the designed embedded system. This integration allows a direct representation of the attacks' implications on the system's security and its risk. The integration of the attacker's perspective in the form of the attack graphs and the risk trees allows a more detailed investigation of the security impacts on the overall system. The other approaches in this research domain only offer a scale-based approach describing security levels, often leading to unclear interpretations.

Regarding the abstract approaches for the integration of security requirements into the DSE, the SaDSE approach offers a much more detailed description model without losing the holistic view on the overall system design. Abstract approaches on the integration of security requirements into the DSE of embedded systems can lead to ambiguous definitions of system designs (e.g., [29]). By explicitly modeling the security assets and mechanisms, and linking them via a rule set, the SaDSE omits this inaccuracy.

The detailed DSE approaches considering security requirements focus on specific parts of embedded systems. Thus, these approaches are not capable of depicting the design of the overall embedded system. Furthermore, the view on the security aspects of these approaches is mostly narrowed towards encryption and authentication mechanisms. The SaDSE is, in contrast, capable of depicting the overall design of embedded systems. The used ruleset allows the definition of various security aspects considering the assets, attacks, and mechanisms. Thus it allows a holistic view of the designed system and allows a detailed consideration of distinct aspects.

Considering the design approaches for secure networks, the main difference regarding the SaDSE framework is the ability to regard the influence of security mechanisms on other system characteristics during design time. The SaDSE, however, does not offer any means of verifying the correct implementation of them.

3.4.2 Security Attack Analysis Tools

The SaDSE framework differs from the projects described in the research area of the attack analysis tools in several aspects. First, the SaDSE framework allows the consideration of the influences of security mechanisms on the overall system's power consumption and performance overhead. Furthermore, our tool allows the evaluation of the key placements' influence on the overall system's security. Lastly, the SaDSE tool supports the evaluation of found solutions regarding their behavior and network characteristics in dynamic environments. The related projects in this research area inspired the SaDSE framework's description of the attack scenarios as attack graphs. We extended this representation by also adding risk trees to the potential threat descriptions [107].

3.4.3 Analytical and Simulation-based DSE

Regarding the related works published in the research area of combined analytical and simulation-based DSE frameworks, the SaDSE framework is best comparable to the work of Fernando Herrera and Ingo Sander [41]. Other works in this research area mostly use traces accumulated during an exhaustive simulation-based DSE and analytically evaluate them regarding their performance and power consumption. Similar to [41], the SaDSE framework uses a first analytical DSE followed by a simulation-based DSE. However, in contrast to the work published by Fernando Herrera and Ingo Sander, the SaDSE framework also models the influence of the security mechanisms on the simulated system's execution time and power consumption. Furthermore, the SaDSE framework automatically generates an executable simulation of the designed system, offering the designers a simulation framework in which they can add further details of the system.

4 The Security-Aware Design Space Exploration Framework

This chapter describes our approach to the introduction of security requirements into the DSE of embedded systems. It gives the big picture of the approach described in this thesis. It explains the distinct design perspectives, from which the system under design can be seen, the computation of the system's characteristics, and the formulation of the constraints the solutions must adhere to. The chapter furthermore describes the transition from the analytical to the simulation-based DSE, as well as the framework's implementation.

4.1 Overview

Figure 4.1 shows the design of the overall security-aware DSE (SaDSE). SaDSE consists of two main parts, the analytical and the simulation-based DSE. During the analytical DSE, the designers describe the system under design from several distinct perspectives: the functional-, the architectural-, and the attack-perspective, as well as the description of the selectable security features. Furthermore, the designers describe the requirements the system under design must fulfill. These descriptions are fed as inputs to the analytical part of the DSE. Based on these inputs, the analytical DSE finds solutions satisfying the given requirements. The designers can define certain goals, which the analytical DSE optimizes. These solutions conclude the analytical DSE step and serve as input to the simulation-based DSE step.

The simulation-based DSE generates simulation models from the solutions found by the analytical DSE. The simulation-based DSE uses these models within a simulation environment. The simulation environment can be configured with different use cases and contains a network simulation. The simulation models communicate with each other using interfaces to this network simulation. The simulation environment is configurable by the system designers. Thus, the simulation models generated from the solutions can be used in different environment setups. Our SaDSE approach allows designers to analyze the solutions found during the analytical DSE and the overall system during the simulation-based DSE.

The analysis of the solutions calculated by the analytical DSE focuses on the hardware component selection, and what tasks map to them. It also focuses on the chosen security mechanisms and how their selection affects the attack scenarios. It also considers the security mechanisms' influence on the solution's timing behavior, power consumption, and costs. The simulation-based DSE provides metrics on the overall system, focusing on its networking behavior. These metrics depend on the used network simulation environment. The system designers can use the results of both the analytical and the simulation-based DSE to adapt the input descriptions, leading to different solutions.

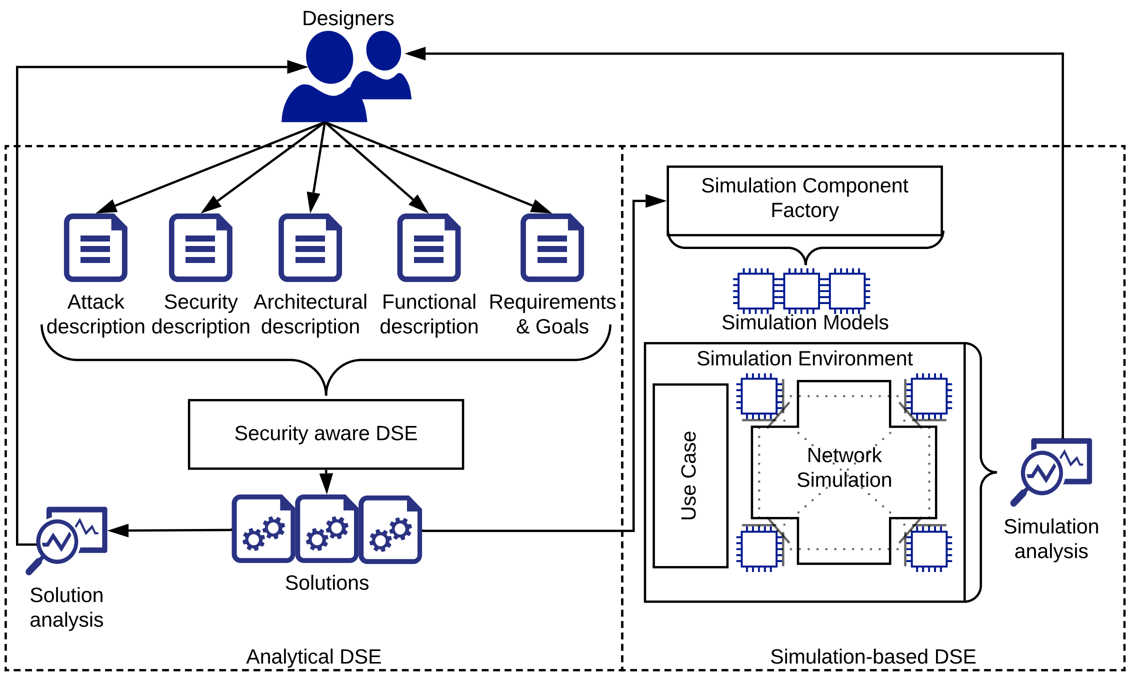


Figure 4.1: Overview on the SaDSE approach.

The following sections describe the details of the analytical and the simulation-based DSE part of the SaDSE framework. Section 4.2 explains the analytical DSE approach, including its system description models.

4.2 Analytical DSE Approach

This section discusses the analytical part of the SaDSE framework. It describes the models the framework offers designers to describe the different perspectives of the system under design. Furthermore, it discusses how the SaDSE framework describes the system partitioning and the task mapping problem from these perspectives and how it calculates the influence of the security constraints on the system's performance, power consumption, and others.

4.2.1 Design Perspectives

The inputs to the SaDSE consist of several perspectives that describe the system under design. These perspectives consist of description models that are interlinked with each other, spanning multiple perspectives. These perspectives include the description of the system's functionality, its architecture, potential security attacks on the system, and the security functions the system can use to mitigate the threats induced by the possible attacks.

4.2.1.1 Functional Perspective

The SaDSE framework supports the description of the system's behavior in the form of a task graph. The nodes within this graph represent the tasks encapsulating the system's functionality. The edges represent the data transfers between the tasks and generally describe the schedule of the tasks. The task graph is a directed graph, consisting of multiple nodes (T) and edges (ϵ_{T_x,T_y}), where each ϵ_{T_x,T_y} connects two nodes T_x and T_y. A task graph contains a starting node (T_s) and an end node (T_e). A node can be both T_s and T_e. A path η through the graph is a set of nodes $N = (T_s, T_x, ..., T_y, T_e)$ and a set of edges $\varepsilon = (\epsilon_{T_s,T_x}, ..., \epsilon_{T_y,T_e})$ [39].

Figure 4.2 shows an example task graph. Each task in the graph might operate on a set of data entities (D_x). These entities are characterized by their size and optional security properties, denoting the data entity as a security asset. The designer additionally describes what operations a task performs on a data entity. The SaDSE framework offers the designer to formalize the operations and the security properties. The operations and security properties are then used to formalize security mapping constraints according to a rule set, also described by the designer. A base set of task operations is denoted $OP = (op_{rx}, op_{tx}, op_r, op_w, op_s, ...)$, consisting of the operations: Receive (op_{rx}), transmit (op_{tx}), read (op_r), write (op_w) and store (op_s). The base set of security properties is described as $SP = (sp_{conf}, sp_{int}, sp_{auth}, ...)$, consisting of the properties: Information disclosure (sp_{conf}), integrity (sp_{int}), and authenticity (sp_{auth}).

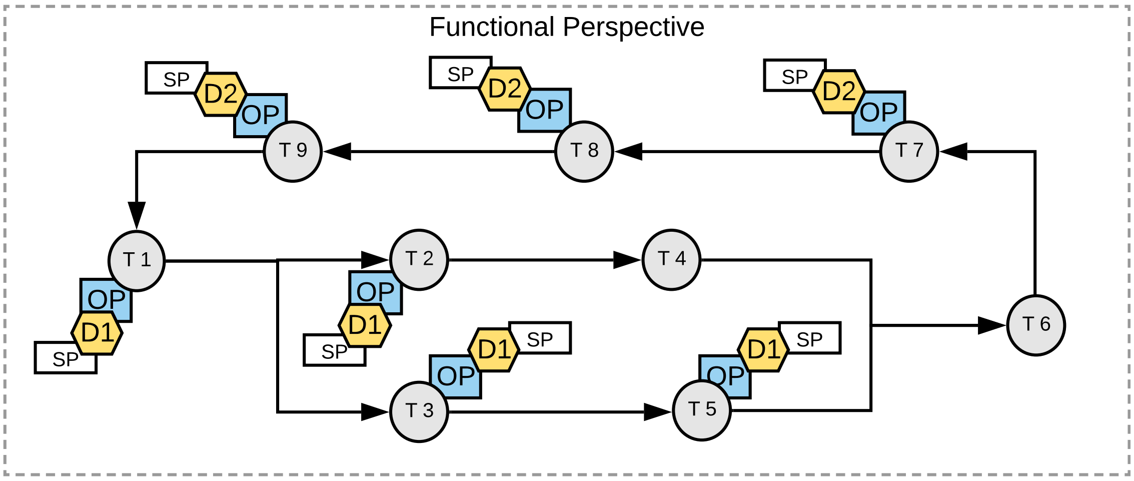


Figure 4.2: Example task graph consisting of multiple tasks (T) operating (OP) and sharing different data entities (D). The data entities are characterized with security properties (SP). T_1 is T_s (start-task) and T_e (end-task).

The tool calculates what security mechanisms the system must provide to the task to protect the data entities it operates on based on the operations of a task on a data entity and a data entity's security properties. This calculation is described in Section 4.2.2.2.

4.2.1.2 Architectural Perspective

The SaDSE framework lets the designers describe the system architecture as hardware components connected via distinct physical channels. These channels can represent wired connections, such as communication buses (e.g., I2C, SPI, Ethernet etc.), but also wireless communication channels (e.g., BLE, W-LAN, WiFi etc.). Both hardware components and physical channels are described by multiple classical and security-based characteristics. The traditional characteristics of the hardware components comprise monetary costs, chip-area size, and dynamic and static power consumption. The security characteristics of a hardware component comprise its security mechanisms and its vulnerability assessment factor.

The physical channels are characterized by transmission speed, power dissipation, and, if applicable, chip-area size. Figure 4.3 shows an example system architecture description. It consists of multiple hardware components connected via physical channels. The physical channels represent communication buses or wireless communication channels. The hardware components are described with their traditional characteristics (trad. char.) and their security mechanisms (SM). Traditional characteristics describe power consumption, execution delay, chip area, costs, and other characteristics crucial for resource-constrained systems. Traditional characteristics also describe physical communication channels.

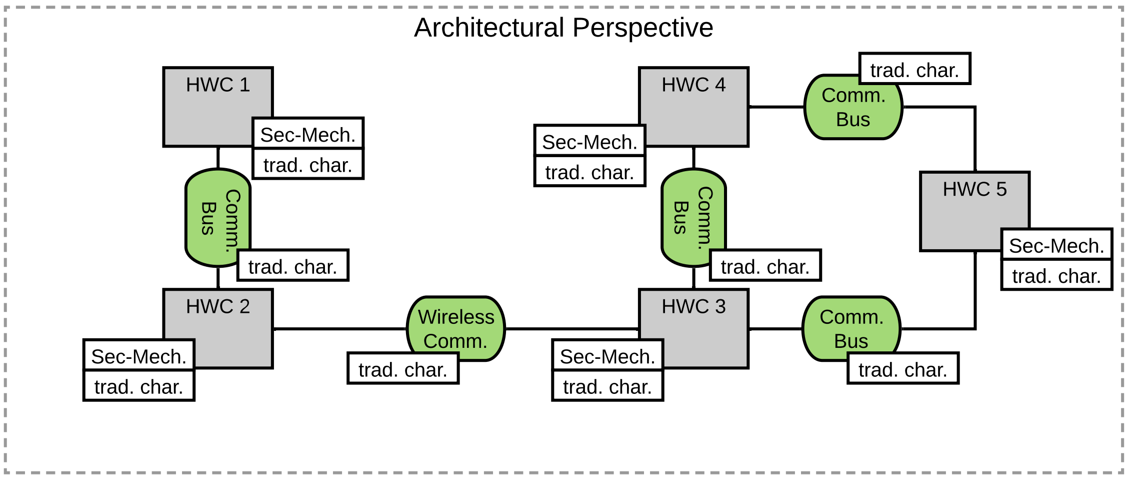


Figure 4.3: Example system architecture description depicting hardware components (HWC) connected with physical channels (e.g. communication buses (Comm. Bus) and wireless communication channels (Wireless Comm.)).

The SaDSE framework offers the designers to freely declare the set of security mechanisms. A base set of security mechanisms is denoted as $SM = (sm_{enc}, sm_{decr}, sm_{sign}, sm_{ver})$, comprising encryption (sm_{enc}), decryption (sm_{decr}), sign (sm_{sign}), and verification (sm_{ver}). In addition to these basic descriptions, the designers can add information about the used key type (symmetric or asymmetric) and the key length to the security mechanisms. Furthermore, the framework offers the designers to determine the link between what security mechanisms protect what security properties using a dedicated rule set. This mapping is explained in detail in Section 4.2.2.2.

4.2.1.3 Attack Perspective

The SaDSE framework offers the designers to describe the potential attackers using two different perspectives, either as a Bayesian Attack Graph (BAG) or as a risk tree.

Bayesian Attack Graphs

The most important perspective for modeling the security requirements is the attack perspective. This perspective builds on modeling potential attack scenarios based on BAG representation. The BAG represents the attacker's opportunities to attack the system as a Bayesian graph, where each node represents a distinct attack step. The conditional probability table (CPT) of a node states with what probability the attacker is capable of successfully performing the attack step. The probabilities stated by the CPT express the estimated possibility that an attacker can successfully conduct the attack step. If an attack step depends on one or more preceding attack steps, the CPT states the attack success possibilities ($asp \in \mathbb{R} : asp \in [0, 1]$) of the current step based on the state of its predecessors. The BAG only knows two states for an attack step, successfully performed (1), or not successfully performed (0). The attack's probability of being unsuccessful is simply computed as the complementary probability of its asp, which is, again, computed depending on the outcome of the attack's predecessors. An example BAG is illustrated in Figure 4.4. This example attack graph shows the link between the attacks and the tasks they are aiming at and the attacks' dependencies reflected in their CPTs. Each attack path aims at least at one attack goal. Attack goals do not have to aim at a task. The description of attacks aiming at distinct tasks also allows the modeling of attacks on the communication between multiple tasks.

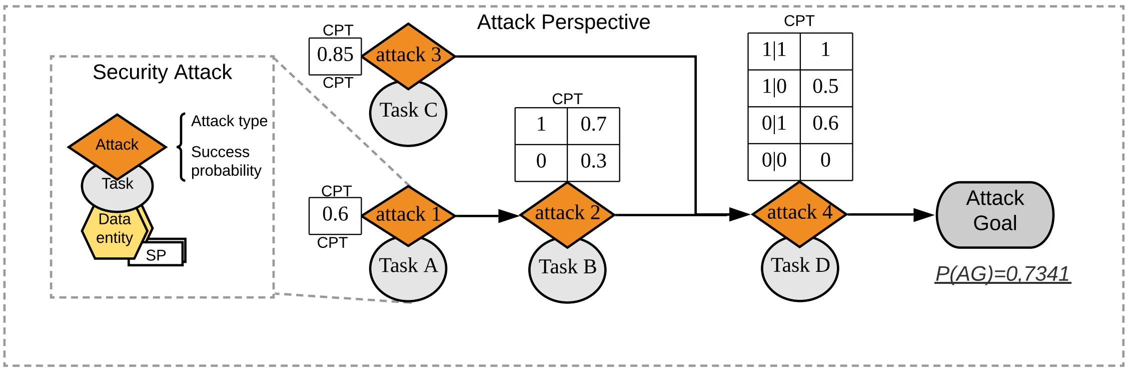


Figure 4.4: Example attack graph represented as a BAG with the attack steps aiming at distinct tasks.

The valuable information of the BAG are the unconditional probabilities of the attacker successfully reaching the attack goals ($P(AG)$). The unconditional probabilities of a node can be obtained by merging the marginal cases of the joint distribution table (JDT) for this node. The framework calculates the JDT applying the Bayesian chain rule (4.1), where $Pa[an_i]$ denotes the conditional probability of the parent of attack node i [32, 139].

$$P(an_1, ..., an_n) = \prod_{i=1}^{n} P(an_i | Pa[an_i]) \tag{4.1}$$

Additionally to the CPT of an attack step, each step is linked to a distinct task in the task graph of the functional perspective and is further characterized by a distinct attack type. This attack-type describes what security property of the task's data entities the attack aims at breaking. The designers describe the attack types. A base set of these attack types $AT = (at_{id}, at_s, ...)$ contains information disclosure (at_{id}) and spoofing (at_s). What attack type aims at what security property is described by the designer using an extendable rule set. This mapping and the influence of task allocation is further described in Section 4.2.2.2.

Risk Trees

The second approach for describing the attacks on the system under design uses risk trees. This approach builds on the *RISKEE* method presented by Krisper et al. [107]. The RISKEE approach builds on a graph representation of potential security attack scenarios, similar to BAGs. The main differences between the *RISKEE* and the BAG based approaches are:

Attack success probability: The *RISKEE* approach uses probability distributions to model the asp (referred to as vulnerability), instead of the discrete probabilities used by BAGs. This probability distribution used in *RISKEE* allows a better representation of the uncertainty of the experts rating the system's susceptibility to a distinct attack.

Impact: Each attack node in the *RISKEE* approach can also be rated with an impact. This impact denotes the monetary loss inflicted by the successful performance of the attack described by the node. The BAG does not allow the direct representation of the attack's monetary impact.

Frequency: The frequency determines how often the attack is attempted in a predefined time. Only the initial attack nodes of a path within the risk tree are further described with the frequency. Also, this aspect is not supported by the BAG method.

Based on the vulnerability, the impact, and the frequency, the *RISKEE* method calculates the monetary risk induced by each successfully performed attack.

At the time of writing this thesis, the *RISKEE* method does not allow the description of a node being dependent on the successful exploit of more than one child node. This limitation stems from the assumption of *RISKEE* that a potential attacker only chooses one route through a path. Hence, a one-to-one translation of a BAG to a *RISKEE*-based risk tree is not possible. This problem is solved using a graph-unrolling algorithm that splits each path in the BAG for each node having more than one predecessor. Thereby, the path is duplicated, with the newly created paths representing all possible routes described by the BAG.

4.2.1.4 Security Functionality Perspective

The security functionality perspective allows the designers to describe security functions usable by the system under design to protect its security assets against potential attacks. This perspective is optional. A security functionality represents a primitive with which to protect a security asset. Thus, before the attacker is capable of attacking the asset itself, he must first break the security functionality. This precondition is modeled by adding an attack to the security function itself. The SaDSE framework describes the protection of the security functionality by adding its attack as a parent to the attack it aims to mitigate. This extension is reflected in the CPT of the original attack. This extension reflects that the execution of the original attack is only executable if the security functionality's attack has been successfully performed.

Depending on the nature of the security functionality, it may depend on the usage of secret keys. The designers must also take into consideration attacks aiming at disclosing the used secret keys. Hence, also secret keys are linked with disclosure attacks. Disclosing the secret key used by a security function automatically renders the security function insecure. For example, the encryption of a security asset to protect its confidentiality is only secure as long as the encryption itself is not broken, or the secret key used in the encryption process is not revealed to the attacker. To model this dependency, the framework links the attack on the secret key as a parent to the security function's attack using the said key. The framework extends the CPT of the attack aiming at the security function in such a way that disclosing the used secret key automatically renders the attack on the security function a success.

The secret keys are characterized by their lifetime, denoted as k_{lt}. This characteristic determines how long the secret key is in usage until it is renewed. This information allows the designer to distinguish between, e.g., session keys and master keys. Furthermore, the SaDSE framework allows the modeling of key derivation chains. A session key used by a system is usually derived from the previously shared master key using a key derivation function (KDF) [97]. The disclosure of a secret key also renders its derived keys insecure. This effect is modeled by adding the key's disclosure attack as a parent to the attack on the keys derived from it. This extension is performed so that the successful disclosure of the parent key automatically renders the attack on the derived keys a success. Figure 4.5 shows the possible extensions of the security functionality on the attack graph modeled in the attack perspective.

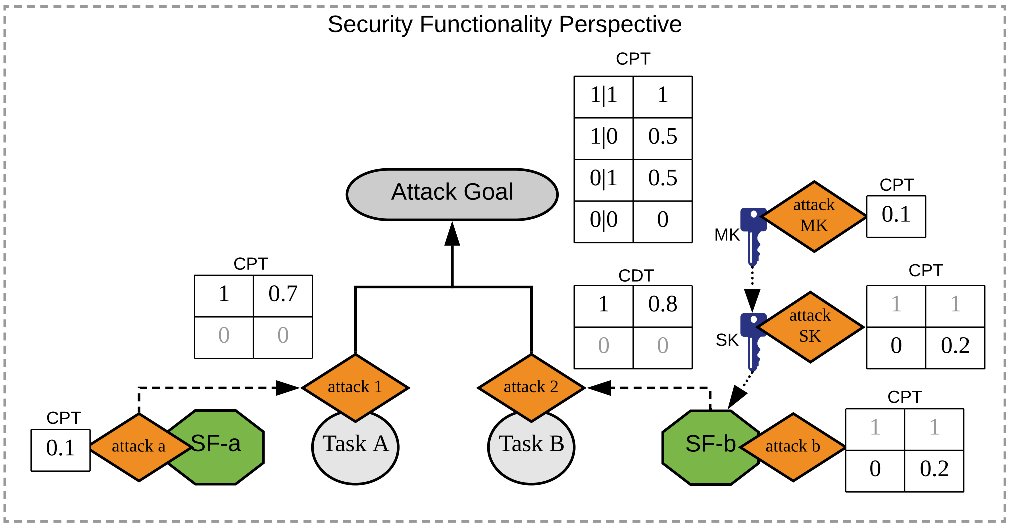


Figure 4.5: Example usage of security functions (SF) with and without the usage of secret keys (e.g., session key (SK) and master key (MK)). In addition, the figure depicts the linkage of their attacks to the original BAG and the extension of the attacks' conditional probability tables (CPTs).

The designer determines what security functions the system under design can choose from to secure its assets. This set of security functions is freely describable by the designer. The framework supports a base set of security functions $SF = (sf_{crypt}, sf_{auth}, ...)$, with sf_{crypt} denoting cryptographic functionality and sf_{auth} denoting authentication functionality. The link between the security functions and the security mechanisms offered by the component is further discussed in Section 4.2.2.2.

4.2.2 System Mapping

The selection of the hardware components and the mapping of the tasks to these components influence the system's performance and power consumption. Such a selection is represented as a solution $S = (P, M)$, with $P = (hwc_1, ..., hwc_P)$ representing the selected hardware components (hwc) and $M = (m(t_1), ..., m(t_M))$ denoting the mappings of tasks (t) to hwc. The security description given to the framework calculates a set of restrictions for the task to component mappings. The following sections explain these security constraints the system under design must fulfill and the influence of the system partitioning and the task mapping on performance, power consumption, and the system's security. In this work, we focus on these system characteristics. However, other characteristics, such as overall system costs or chip area size, can be calculated similarly to the presented ones. The general purpose of the SaDSE framework is to find feasible system partitioning and task mappings that satisfy all constraints imposed by the designers.

To better describe the system partitioning and task mapping and their influence the system's security, performance, and power consumption, we consider the following example of a very simple sensor system, consisting of a sensor node and a gateway. The sensor node accumulates data and sends it to the gateway. The gateway receives the data, analyzes it and stores it for later statistical operations. The system's functionality is described by the tasks: sensor data accumulation (t_{acc}); sending of sensor data (t_{tx}); receiving of the sensor data (t_{rx}); sensor data analysis (t_{sda}); store sensor data (t_{store}). We denote the sensor data accumulated by the sensor node and stored by the gateway d_{sensor}. The task t_{acc} writes (op_w) and transmits (op_{tx}) d_{sensor}. The tasks t_{tx} and t_{rx} only transmit and receive d_{sensor}. The task t_{sda} receives, reads (op_r), and sends d_{sensor}. The task t_{store} receives, reads, and stores (op_s) d_{sensor}. The task t_{acc} sends d_{sensor} to t_{tx} which further transmits d_{sensor} to t_{rx}. The task t_{rx} transmits d_{sensor} to t_{sda} which finally sends it to t_{store}. The sensor node and the gateway are connected via a wireless communication channel. Both the sensor node and the gateway can be realized using various hardware components (hwc), e.g., components supporting encryption (sm_{enc}), decryption (sm_{decr}), and other security mechanisms. What components and task mappings the SaDSE chooses is based on mapping rules described by the designer. The different mappings are defined by distinct WCETs. We use this simple example to give a more descriptive explanation of the system mapping performed by the SaDSE framework. The SaDSE's mapping mechanism is explained in the following sections.

4.2.2.1 Task Mapping and Calculation of Performance and Power Consumption

The mapping of the tasks to the selected hardware components influences the system's execution time and power consumption. This section explains how the SaDSE calculates the performance and power consumption of a distinct system partitioning and task mapping. The mapping of tasks to hardware components must fulfill basic task reachability. Task reachability means that communicating tasks can only be mapped to the same hardware component or different hardware components that are directly linked to one another via a physical channel. The SaDSE framework automatically ensures this property, discarding all solutions, not meeting this basic requirement. When considering the example defined in the section above, all five tasks must be mapped to hardware components directly connected.

Lets assume the that the overall system's architecture consists of three hardware components, one for the sensor node (hwc_{sensor}) and two for the gateway (hwc_{gw1} and hwc_{gw2}). The components hwc_{sensor} and hwc_{gw1} represent BLE radios, hwc_{gw2} represents an extended storage system. The BLE radios hwc_{sensor} and hwc_{gw1} are connected via a BLE communication channel, hwc_{gw1} and hwc_{gw2} are connected with an I2C communication bus. If the SaDSE framework decides to map t_{acc} and t_{tx} to hwc_{sensor}, the basic task reachability only allows the mapping of t_{rx} to hwc_{gw1}, hence, prohibiting its mapping to hwc_{gw2}. This limitation is caused by hwc_{sensor} and hwc_{gw1} not being directly connected. Figure 4.6 shows a valid task mapping based of the simple sensor system example. The data entity d_{sensor} is depicted as D.

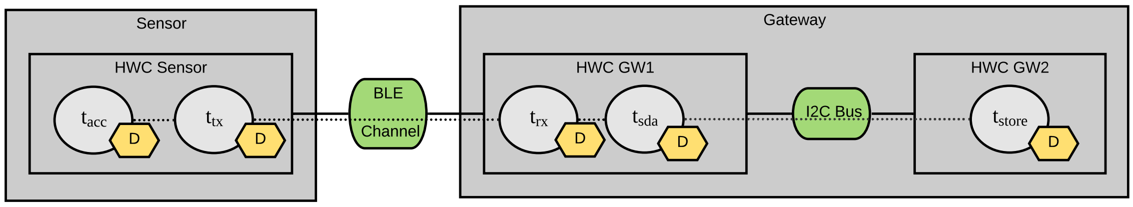


Figure 4.6: Reachable task-to-hardware-component mapping based on the simple sensor system example. The data entity d_{sensor} is depicted as D.

The potential mapping of a task to a hardware component is characterized by the estimated worst-case execution time (WCET) of the task implementation on the said component. This WCET must be estimated by domain experts. A task-to-hardware-component mapping is only possible if a WCET has been previously estimated for this mapping. The calculation of the system's performance and power consumption is based on [48]. To calculate the solution's delay λ_S, the delays of all possible paths through the process graph ($\lambda_H = (\lambda_{\eta_1}, ..., \lambda_{\eta_H})$) must be calculated. The most time-consuming path dictates the system's overall delay, which is calculated as $\lambda_S = max(\lambda_H)$.

The delay of a path in the process graph is calculated as $\lambda_\eta = \lambda_{proc} + \lambda_{comm}$, which is the sum of the path's processing time λ_{proc} and the its communication time λ_{comm}. This calculation is performed for each path in each found solution given its system partitioning and task mapping. The mapping of a task (t_x) to a hardware component (hwc_y) in a given solution is determined by the function $hwc_y = m_{t_x}$. The delay of the task execution is calculated according to definition 4.2.1.

Definition 4.2.1. A path's processing time λ_{proc} is calculated as $\lambda_{proc} = \sum_i^{n_\eta} \delta(t_i, m_{t_i})$, where the function $\delta(t_x, hwc_y)$ returns the WCET of task t_x being mapped to the hardware component hwc_y and n_η denotes the number of all tasks on the path.

The communication delay $\lambda_{comm} = \sum_{i=1}^{k} \lambda_{pcc_i}$ is calculated based on the data entities transmitted via the physical communication channels ($Pcc = (pcc_1, ..., pcc_k)$), with k being the number of all pcc used on the path. What data is transferred on what pcc and how time consuming the transmission is defined in 4.2.2.

Definition 4.2.2. The transmission time λ_{pcc_x} is calculated as $\lambda_{pcc_x} = \frac{v_{pcc_x}}{S(D_{pcc_x})}$ where v_{pcc_x} determines the transmission speed of pcc_x, D_{pcc_x} represents all data entities transmitted via pcc_x, and the function $S(D) = \sum_{i=1}^{D} s_{d_i}$ determines the size s_d of all data entities d in the set $D = (d_1, ..., d_D)$. To determine what data entities are being transmitted via a physical channel, the framework finds all tasks (N) directly connected by the path and allocated on different hardware components connected to pcc_x. The framework then checks for all tasks in N the set of commonly used data entities D. This set D must be transferred between the tasks, and, hence via pcc_x.

The task mapping influences not only the system's performance but also its power consumption. The power consumption of a hardware component consists of a static and a dynamic part. The static power consumption is independent of the task mapping. The dynamic part of the power consumption scales with the tasks being mapped to the hardware component, as it influences its active execution time. The power consumption of the process execution (ρ_{proc}), the system's static power consumption (ρ_{static}) and of the communication (ρ_{comm}) are defined in 4.2.3.

Definition 4.2.3. The static power consumption is calculated as $\rho_{static} = \sum_{i=1}^{m} \rho_{hwc_i}$, where ρ_{hwc_i} denotes the static power consumption of hardware component i and m is the number of all hardware components chosen by the solution. The power consumption caused by the process execution is described as $\rho_{proc} = \sum_{i=1}^{n} \delta(t_i, m_{t_i}) * pwr(m_{t_i})$, where $pwr(hwc_x)$ returns the dynamic power consumption of hwc_x. The power consumption of the communication depends on the amount of data transferred via the physical communication channels of the system. Each physical communication channel is characterized with its power dissipated during data transmission (pwr_{pcc}). The overall power consumption $\rho_{pcc_x} = \lambda_{pcc_x} * pwr_{pcc}$ of a channel pcc_x depends on its data transmission time λ_{pcc_x}. The system's communication power consumption is calculated as $\rho_{comm} = \sum_{i=1}^{k} \rho_{pcc_i}$

The solution's power consumption $\rho_S = \rho_{proc} + \rho_{static} + \rho_{comm}$ is the sum of the static power consumption of the selected hardware components, the dynamic processing power dissipation and the power consumption of the communication system.

This section shows how the SaDSE framework calculates the system's performance and its power consumption regarding the traditional system characteristics. This calculation does not consider integrating the security mechanisms used by the system to meet the security requirements. The following section describes how these mechanisms affect the overall system's performance and power consumption.

Lets exemplify this with our sensor-gateway example. Lets assume the tasks t_{acc} and t_{tx} are mapped to hwc_{sensor}, t_{rx} and t_{sda} mapped to hwc_{gw1}, and t_{store} is mapped to hwc_{gw2}. The implementations of t_{acc} and t_{store} come with an WCET of 5ms, the implementations of t_{tx} and t_{rx} come with an WCET of 2ms, and the implementation of t_{sda} comes with an WCET of $3s$ms. All WCET describe the execution times of the tasks mapped to their respective hardware components. The data entity d_{sensor} contains 1kB of information. The BLE channel connecting the radios of hwc_{sensor} and hwc_{gw1} transmits data with 2MB/s. The I2C communication bus connecting hwc_{gw1} and hwc_{gw2} supports a transmission speed of 1MB/s. All hardware components come with a static power consumption of 1µW/s and a dynamic power consumption of 3µW/s. The I2C communication bus comes with a power consumption of 1µW/s and the communication via the BLE channel consumes 3µW/s. This setup would lead to an overall system latency of $\lambda_S = \lambda_{proc} + \lambda_{comm} = 17\text{ms} + 1.5\text{ms} = 18.5\text{ms}$ and a power consumption of $\rho_S = \rho_{proc} + \rho_{static} + \rho_{comm} = 51\text{nW} + 18.5\text{nW} + 2.5\text{nW} = 72\text{nW}$.

4.2.2.2 Security Mapping

The approach presented in this thesis builds on the formulation of specific security constraints, a task mapping, and system partitioning must adhere to in order to be considered secure. This formulation includes the generation of security operations, the mapping of attacks to security assets, the automatic assessment of secure channels within the solution, the selection of appropriate security functions, and the utilization of the provided security mechanisms.

The general idea is that data entities comprising security properties define security assets. These security assets are target to specific attack types, and, hence, must be protected. Each task operating on the security assets must perform certain security operations to protect the used asset. These operations must be supported by the security mechanisms of the hardware components the task is mapped to. The security mechanisms mitigate the attacks aimed at the tasks using the security assets, thus, influencing the probability of the attacker successfully reaching the attack goal. Furthermore, the designers can define functions which realize the security operations, thus, adding additional flavor to the security selection performed by the framework. These aspects are detailed in the following sections.

Security Operations

The framework calculates the security operations ($SecOp$) for each task based on its operations OP performed on the data entities and the entities' security properties SP. This dependency is determined by the designers of the system using a rule set offered by the SaDSE framework. This rule set, denoted SOR, combines operations $op \in OP$ and security properties $sp \in SP$ using Boolean expressions. This set of rules is mapped on the set of security operations $SOR \mapsto SecOp$. The relation between op and sp are described with Boolean operators. An example mapping of $SOR \mapsto SecOp$ could be $f((op_w \vee op_r) \wedge sp_{conf}) = so_{enc}$, where so_{enc} denotes the security operation of encryption. To assess the usage of the $SecOp$ in the establishment of secure channels, they must be further characterized with information considering their counterpart operation. For example, so_{enc} is intended to protect an asset's sp_{conf} when transmitted from a sending task to a receiving task. Therefore, so_{enc} of the sender has a counterpart on the receiver side. In contrast, a security operation that checks an asset's integrity might only be used by a task itself. The designer must define a set of counterpart operations $CP = (so_1, ..., so_{CP})$ that contains all $SecOp$ having a counterpart operation. This characteristic is used during the DSE to determine secure channels and, further, the utilization of the right security mechanisms.

Lets consider the data entity d_{sensor} from our simple example. If the designer decides that d_{sensor} must be confidential (sp_{conf}) and adds the rule $f((op_w \vee op_r) \wedge sp_{conf}) = so_{enc}$, then all tasks performing $(op_w \vee op_r)$ on d_{sensor} must perform an encryption operation so_{enc} to protect d_{sensor}. In our example this includes the tasks t_{acc}, t_{sda} and t_{store}. How this encryption operation can be realized is stated by the security mechanism mapping, described below.

Security Asset and Attack Mapping

The designer defines the relationship between the security assets used by the designed system and the potential attack scenarios. The designer describes this relation by a rule set associating an attack node's attack type (AT) to an asset's security property $AT \mapsto SP$. For example, the rule $f(at_{id}) = sp_c$ maps an information disclosure (at_{id}) attack to an asset which needs its confidentiality sp_c protected.

Lets consider an attack aiming at our simplified sensor system. This attack targets the confidentiality of the exchanged sensor data and, hence, is determined by the rule $f(at_{id}) = sp_c$. Such an attack aiming at a system's task compromises all data entities coming with the security property confidentiality sp_c said task operates on. Lets assume that an information disclosure attack aims at the data accumulation task t_{acc}. This compromises the confidentiality of d_{sensor}. Hence, the SaDSE framework enforces t_{acc} to perform an encryption of d_{sensor} (performed by so_{enc}) to protect the confidentiality of d_{sensor}.

Secure Channel Determination

Selecting the appropriate security mechanisms to secure the system's assets requires the determination of secure channels. The assessment of secure channels builds on the $SecOp$ a task performs and the security asset (sa) the operation is performed on. Lets denote a function $\chi(t, sa)$ that tells us what security operations the task t performs on the security asset sa. A secure channel sc_{t_s,t_d} spans between a source task t_s and destination task t_d, where $\chi(t_d, sa) \cap \chi(t_s, sa) \neq \emptyset$. To find these secure channels, all paths in the process graph must be traversed for each task performing a security operation listed in the set CP. For example, a task t_x performing an encryption operation $so_{enc} \in CP$ on an asset sa_x spans a secure channel to a succeeding task t_y performing so_{enc} on the same asset sa_x. If the same tasks performed another $so' \notin CP$ on the same asset, no secure channel would spanned between them.

In our simple example, secure channels would be spanned between t_{acc} and t_{sda} ($sc_{t_{acc},t_{sda}}$), and t_{sda} and t_{store} ($sc_{t_{sda},t_{store}}$). These channels are spanned as all these tasks perform encryption operations so_{enc} as defined by the designer. One can notice, that the tasks t_{tx} and t_{rx} are enclosed by the channel $sc_{t_{acc},t_{sda}}$. Therefore, the communication via the BLE channel is encrypted as well.

Security Function Selection

As already mentioned earlier, the description of potential security functions that can be used by the designed system is optional. What security functions are selected for a task to protect its assets is determined by the task's $SecOp$. The system designers define this mapping as $SecOp \mapsto SF$.

In our small example, the designer could add the rule $f(so_{enc}) = sf_{crypt}$, with sf_{crypt} denoting a cryptographic function. Hence, the designer adds another detail to the system description, stating that all tasks using an encryption operation so_{enc} perform this operation using a cryptographic function sf_{crypt}. This rule would enforce for the tasks t_{acc}, t_{sda}, and t_{store} to use sf_{crypt} to protect d_{sensor}.

Security Mechanism Mapping

The security mechanisms mapping describes what security mechanisms implement the security operations and security functions. Depending on the chosen representation of the security features used in the system design, the security mechanisms of the system's architecture map to the $SecOp$ or the selected security functions SF. The mapping rules $\kappa^{SecOp} \mapsto \kappa^{SM}$ and $\kappa^{SF} \mapsto \kappa^{SM}$ determine these mappings, where κ^{SecOp} denotes a set of Boolean expressions combining environmental attributes and security operations, and κ^{SF} denotes a set of Boolean expressions combining environmental attributes and security functionalities. κ^{SM} denotes a set of expressions combining security mechanisms using Boolean operators. For example, the rule $f(so_{enc} \wedge ext) = sm_{enc,env_{ext}}$ states that an encryption operation must be implemented by an encryption mechanism. They are, again, definable by the system designer. The system partitioning and the task mapping, hence, influence what security mechanisms each task can select from. Both the mappings of security options and the security functions to their respective mechanisms are further defined by two fixed attributes coming with our approach: the directional and the environmental attributes. Figure 4.7 shows the simple sensor system with the selected security operations, the spanned secure channels, the used security mechanisms and the attacks aiming at the sensor data, shown as D in the depiction..

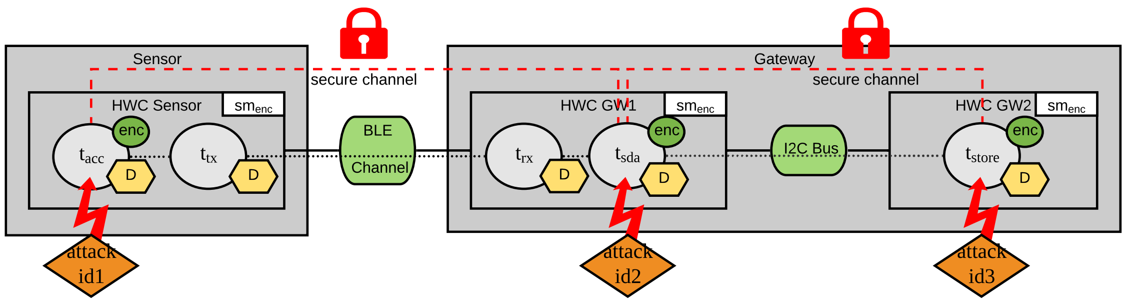


Figure 4.7: Simple sensor system example showing the secure channels between spanned t_{acc} and t_{sda}, and t_{sda} and t_{store}. It also shows the usage of so_{enc} and the information disclosure attacks targeting the distinct tasks and the used security mechanisms. The data entity d_{sensor} is shown as D.

The environmental attributes define in what circumstances the framework should select what security mechanisms. These attributes categorize security mechanisms into internally ($sm_{env_{int}}$) and externally ($sm_{env_{ext}}$) used mechanisms. These attributes refer to the allocation of the secure channel sc_{t_s,t_d} between the tasks t_s and t_d. If both t_s and t_d are allocated to the same hardware component, the framework selects internal mechanisms. If these tasks are allocated to different components, external mechanisms are used. Lets consider two mechanisms, encryption (sm_{enc}) and trusted memory management (sm_{tmm}), with their respective attributes $sm_{enc,env_{ext}}$ and $sm_{tmm,env_{int}}$. For a secure channel with $m_{t_s} = m_{t_d}$ the framework would select sm_{tmm}, whereas for a secure channel with $m_{t_s} \neq m_{t_d}$, it would select sm_{enc}.

Considering our simple example, the SaDSE framework has different choices for selecting appropriate security mechanisms considering the secure channels spanned between t_{acc} and t_{sda}, as well as between t_{sda} and t_{store}. When mapping t_{acc} on hwc_{sensor} and t_{sda} on hwc_{gw1}, the SaDSE framework would select the encryption mechanism sm_{enc} due to its external attribute. For the secure channel between t_{sda} and t_{store}, the SaDSE framework would select sm_{tmm} if both tasks would be mapped to hwc_{gw1}, and sm_{enc} if t_{sda} would be mapped to hwc_{gw1} and t_{store} to hwc_{gw2}. This selection also enforces that all hardware components must be able to offer the respective security mechanisms. Otherwise, such a mapping would be infeasible.

The directional attributes defined by the SaDSE framework characterize when the security mechanisms are utilized. These directions define if a mechanism is used on a security asset before or after the task operates on it. It distinguishes between input (dir_{in}), output (dir_{out}) and bi-directional (dir_{bi}) mechanisms. Lets consider a system with a secure channel sc_{t_s,t_d} with $m_{t_s} \neq m_{t_d}$, where both the tasks perform so_{enc}. If the hardware components on which t_s and t_d are allocated, provided sm_{crypt} with dir_{bi}, the framework would select this mechanism for both t_s and t_d. If the components, however, provided sm_{enc} with dir_{out} and sm_{dec} with dir_{in}, the framework would select sm_{enc} for t_s and sm_{dec} for t_d. The definition of a mechanisms direction lets the designers differentiate between, e.g., a component's encryption versus decryption delay and power consumption.

Lets consider the directional attribute in the simple example. Lets assume the encryption mechanism defined as sm_{crypt} with dir_{bi} for hwc_{gw1} and hwc_{gw1}, and t_{sda} mapped to hwc_{gw1}, and t_{store} mapped to hwc_{gw2}. In this configuration, both t_{sda} and t_{store} would use sm_{crypt} to secure the confidentiality of d_{sensor}. If the encryption mechanism of hwc_{gw1} were to be defined as sm_{enc} with dir_{out} and sm_{dec} with dir_{in}, t_{sda} would use sm_{enc} to protect d_{sensor} before transmitting it to t_{store}. In the other direction t_{sda} would use sm_{dec} to decrypt d_{sensor} when receiving it from t_{rx}.

The directional and environmental attributes apply to both the mapping of security options and security functions to mechanisms. The latter mapping further depends on keys used by the functions. If a selected security function uses secret keys, their types also dictate what security mechanisms can be used. The type mapping is again definable by the designers. The attributes of a secret key can, for example, characterize it as a symmetric or asymmetric key. Also, the length of a key can be described by the designers. This mapping is defined with the rule set $SF_{key_{attrib}} \mapsto SM_{attrib}$. For example, an AES encryption algorithm can only be mapped to an encryption security function using a symmetric key with key sizes of 128, 192, or 256 bit.

When considering our simple sensor system example, the mapping of $f(so_{enc}) = sf_{AES,k_{256}}$ ($sf_{AES,k_{256}}$ denotes AES encryption using a 256 bit key) would lead to the usage of $sf_{AES,k_{256}}$ by the tasks t_{acc}, t_{sda}, and t_{store}. The rule $f(sf_{AES,k_{256}}) = sm_{AES,k_{256}}$ would force t_{acc}, t_{sda}, and t_{store} to be mapped to hardware components supporting $sm_{AES,k_{256}}$. Hence, if one of the components hwc_{sensor}, hwc_{gw1}, and hwc_{gw2} would not support $sm_{AES,k_{256}}$, a mapping to it would be infeasible. Furthermore, if any hardware component would also offer another encryption mechanism, e.g., $sm_{AES,k_{128}}$, the framework would enforce $sm_{AES,k_{256}}$ to be used by the mapped task. Figure 4.8 shows the simple sensor use case with the used security functions and the selected security mechanisms based on their direction. It also shows the keys used by the security functions.

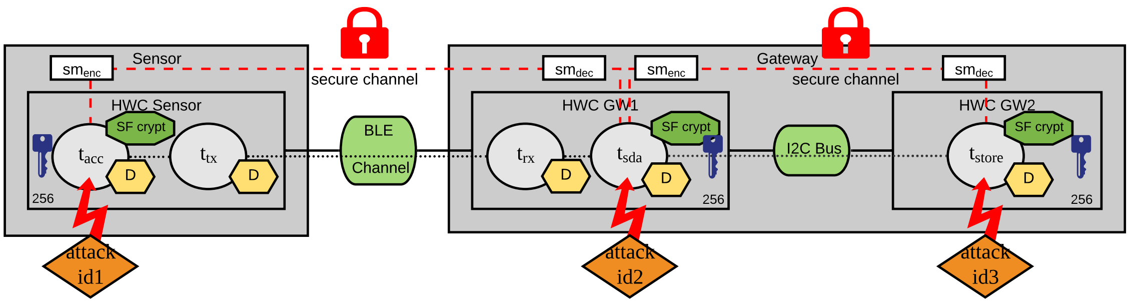


Figure 4.8: Simple sensor system example showing the secure channels spanned between t_{acc} and t_{sda}, and t_{sda} and t_{store}. It also shows the usage of sf_{crypt} and information disclosure attacks aiming at the distinct tasks and the used security mechanisms based on their direction. Furthermore, it shows how the system uses the key k_{256} to realize the cryptographic functions.

A task can utilize different security functions to protect its assets, which might not be implementable on each hardware component. This selection of the security functions adds another flavor to the found solutions by the SaDSE tool, extending the solution's representation ($S = (P, M, SF_{sel})$) by the set of selected functions SF_{sel}. The mapping of $SecOp$ and SF to SM influences the system's security, performance, and power consumption. These influences are described in the following section.

4.2.2.3 Security Influence on Performance, Power Consumption, and Attack Mitigation

The mapping of the tasks to the hardware components and the resulting selection of the security mechanisms influence, additionally, the system's security, its performance, power consumption, and other characteristics.

Influence on Attack Mitigation

The mapping of the tasks to the hardware components and the selection of what security mechanisms to perform the $SecOp$ and functions, respectively, influence the probability of successful attacks. The selected security mechanisms mitigate each attack aiming at a task's assets. To quantify this mitigation, the designers must assess the vulnerability of the security mechanisms implementation provided by the hardware component. This vulnerability is captured as $iv \in \mathbb{R} : iv \in [0, 1]$ and assessed for each hardware component selectable for the system partitioning. The influence of the security mechanism's iv on the mitigated attacks is reflected in both the attack scenarios modeled as BAGs and as risk trees. If modeled as a BAG, the iv reduces the asp in the attacks' CPTs. For each attack mitigated by a mechanism, the framework computes $asp * iv$ for all entries in the attack's CPT. If modeled as a risk tree, the same reduction of the asp is calculated for the mitigated attacks.

The security mechanisms are only capable of mitigating attacks if the mechanism's type matches the attack's type. This matching between mechanisms and attacks is defined by the designer using rules that describe $SM \mapsto AT$. Furthermore, the definition of security functions influences what attacks get mitigated. If no security functions have been defined, the security mechanisms mitigate the attacks aiming at the tasks using them. If security functions have been defined, the mechanism' mitigation does not apply to the attack aiming at the task using the function, but on the attack aiming at the security function itself. Hence, the iv states the vulnerability of the mechanisms implementing the used security function. The task's assets are already protected by using the security function.

The keys used by the security functions also influence the system's security. Their placement on the different components and the components' provided security mechanisms with their iv reduce the disclosure attacks aiming at them. However, not all security mechanisms are capable of protecting the stored secret keys. What secret key can be protected by what mechanism depends on the key's lifetime ($k_{lt} \in K_{lt}$). This mapping is defined by the system designers using the rule set $K_{lt} \mapsto SM$.

Lets exemplify the attack mitigation and the key storage using our simple example. Lets assume that an information disclosure attack at_{id} aims at t_{acc} and has an asp of 0.7. The task t_{acc} is mapped to hwc_{sensor}, which offers sm_{enc} and comes with an iv of 0.1. This mapping would reduce the asp of at_{id} to 0.07, making the attack much less likely to be successful. However, this reduction is only performed if the designer defines the security mechanism sm_{enc} to counteract at_{id}. Considering the key usage in our example, lets assume that the designer defines that the security operation so_{enc} uses $sf_{AES,k256}$ and hwc_{sensor} supports $sm_{AES,k_{256}}$. In this case, the attack on $sf_{AES,k256}$ would be defined as a parent to at_{id} and mitigated by iv of hwc_{sensor}. As $sf_{AES,k256}$ uses a 256-bit key, the attack aiming at the disclosure of the key is defined as the parent of the attack aiming at the security function $sf_{AES,k256}$. This key disclosure attack can also be mitigated by hwc_{sensor}. If the designer, e.g., defines the 256-bit AES key to be a long-term key that can be secured by secure storage (sm_{tss}), hwc_{sensor} can protect the key if it supports sm_{tss}. Otherwise, the key disclosure attack is not mitigated. Figure 4.9 shows the different approaches for modeling the mitigation of potential attacks based on the simple sensor example.

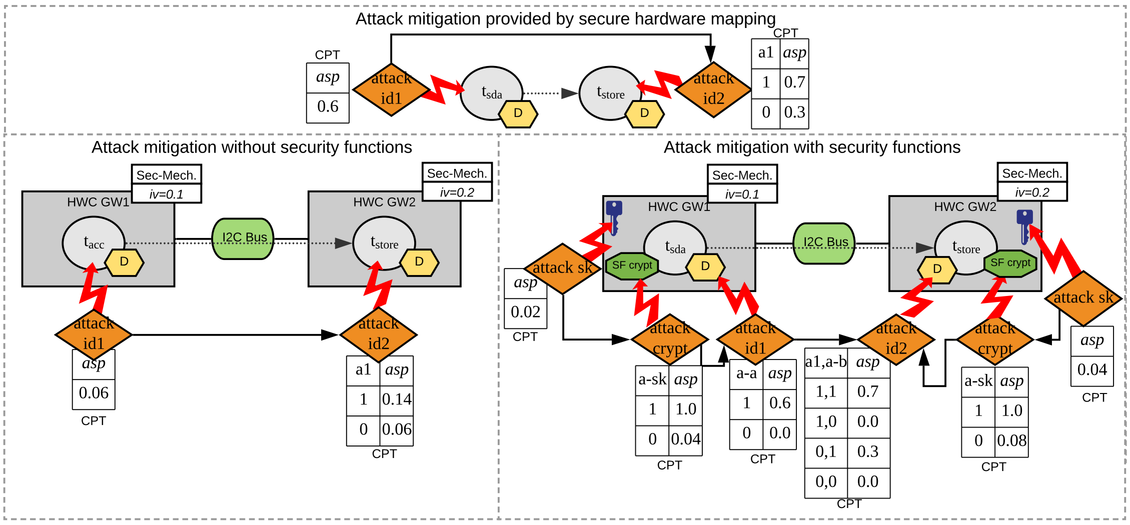


Figure 4.9: Attack mitigation of security mechanisms (Sec-Mech.), with and without the definition of security functions. The depicted example refers to the simplified sensor system example.

Influence on System Performance and Power Consumption

The usage of security mechanisms to protect viable assets of systems always comes at the expenses of the overall system's delay. This additional delay caused by the usage of the securiy mechanisms is added to the delay of the tasks utilizing them. Hence, the overall performance of the system is influenced. This security delay is calculated for each solution found by the tool. For any task t_i we consider its used security assets $SA_{t_i} = (sa_{1,t_i}, ..., sa_{n,t_i})$ on which t_i performs the $SecOp$. Lets denote a function $SM_{t_i,sa_{j,t_i}} = sm(t_i, m_{t_i}, sa_{j,t_i})$ which returns the set of security mechanisms ($SM_{t_i,sa_{j,t_i}}$) used by task t_i being mapped to m_{t_i}, and protecting sa_{j,t_i}. If the designers also specify security functions to be used by the found solutions, the set of security mechanisms $SM_{t_i,sa_{j,t_i},sf_{t_i}} = sm(t_i, m_{t_i}, sa_{j,t_i}, sf_{t_i})$ is additionally dependent on the security functions sf_{t_i} used by t_i. Hence, a security mechanism sm_k is either defined as $sm_k \in SM_{t_i,sa_{j,t_i}}$ or $sm_k \in SM_{t_i,sa_{j,t_i},sf_{t_i}}$. The delay induced by a sm_k depends on whether its delay (γ_{sm_k}) is given as an absolute measure or in relation to the data size. Eq. 4.2 defines the overall computation delay of sm_k in relation to the asset's size s_{sa} ($\lambda_{sm_k}(\gamma_{sm_k}, sa_{j,t_i})$).

$$\lambda_{sm_k}(\gamma_{sm_k}, sa_{j,t_i}) = \begin{cases} \gamma_{sm_k} * s_{sa_{j,t_i}}, & \text{if } \gamma_{sm_k} \text{ given in relation to the data unit.} \\ \gamma_{sm_k} & \text{otherwise.} \end{cases} \tag{4.2}$$

The overall security computation of a task is determined as $\lambda_{sec}(t_i) = \sum_{k=1}^{|SM|} \sum_{j=1}^{|SA|} \lambda_{sm_k}(\gamma_{sm_k}, sa_{j,t_i})$, where SM denotes the set of all security mechanisms used by t_i and SA denotes the set of all assets of t_i.

The power consumption induced by the usage of security mechanisms is computed similarly to their performance overhead. Similar to the delay, also, the overall power consumption of a security mechanism $\rho_{sm_k}(\omega_{sm_k}, sa_{j,t_i})$ depends on the determination of its power consumption ω_{sm_k} in relation to the size of the secured asset, as defined in Eq. 4.3.

$$\rho_{sm_k}(\omega_{sm_k}, sa_{j,t_i}) = \begin{cases} \omega_{sm_k} * s_{sa_{j,t_i}}, & \text{if } \omega_{sm_k} \text{ given in relation to the data unit.} \\ \omega_{sm_k} & \text{otherwise.} \end{cases} \tag{4.3}$$

The overall power consumption induced by the used security mechanisms is then computed as $\rho_{sec}(t_i) = \sum_{k=1}^{|SM|} \sum_{j=1}^{|SA|} \rho_{sm_k}(\omega_{sm_k}, sa_{j,t_i})$, where SM denotes the set of all security mechanisms used by t_i and SA denotes the set of all assets of t_i. The performance delay and the power consumption of the security mechanisms is added to the performance characteristics of the distinct tasks as defined in Section 4.2.2.1. Hence, the calculation of the processing time of each path within a found solution is extended to $\lambda_{proc} = \sum_i^{n_\eta} \delta(t_i, m_{t_i}) + \lambda_{sec}(t_i)$, with n_η denoting all tasks within the path. The solution's power consumption is calculated as $\rho_{proc} = \sum_{i=1}^{n} (\delta(t_i, m_{t_i}) * pwr(m_{t_i})) + \rho_{sec}(t_i)$, with n being the number of all tasks in the system.

Lets take the simple example again and consider that the tasks t_{acc}, t_{sda} and t_{store} perform so_{enc} that is implemented by sm_{enc} by all hardware components with $\gamma_{sm_{enc}} = 2\text{ms}$ and $\omega_{sm_{enc}} = 1\text{nW}$. This would increase λ_S from $\lambda_S = 18.5\text{ms}$ to $\lambda_S = 18.5\text{ms} + 3 * \gamma_{sm_{enc}} = 24.5\text{ms}$ and ρ_S from $\rho_S = 72\text{nW}$ to $\rho_S = 72\text{nW} + 3 * \omega_{sm_{enc}} = 75\text{nW}$.

4.2.3 Constraint and Optimization Goal Definition

The designers can formalize constraints with regards to the system's performance, power consumption, and overall security, additionally to the system's description from distinct perspectives. Additionally, they can determine characteristics to be optimized for. Based on these constraints and optimization goals, the SaDSE tool finds system solutions fulfilling the defined constraints and optimization goals.

4.2.3.1 Traditional Constraints and Optimization Goals

In this thesis, we focus on the two traditional constraints on the system's power consumption and performance, as for our work, those are the most important ones. Further constraints can be defined similarly. The prerequisite for applying these constraints is to obtain feasible solutions. Any solution $S = (P, M)$ can be considered feasible if it fulfills two basic requirements. First, the solutions can only consist of task mappings M that are realizable, meaning that for each task mapping m_{t_x} a WCET must be defined ($\delta(t_i, m_{t_i}) > 0$). Second, the task mappings of S must be reachable. Neighboring tasks in the task graph are only allowed to be mapped to either the same hwc or different hwc connected by a pcc. This feasibility check results in finding the set of feasible solutions $\mathbb{S}_f \subseteq \mathbb{S}$ among all solutions $\mathbb{S}$ [19].

The goal of the design space exploration is to find valid solutions among the set of feasible solutions. To determine what solutions are valid, we focus on two constraints that a solution must fulfill: a performance (c_λ) and a power consumption constraint (c_ρ). To determine if a solution's performance and power consumption fulfill these constraints, we represent these characteristics as objective functions $f_\lambda(S) = \lambda_S$ and $f_\rho(S) = \rho_S$, respectively. For a valid solution the following two assumptions must be true: $f_\lambda(S) \leq c_\lambda$ and $f_\rho(S) \leq c_\rho$. When aiming at a performance or power consumption optimal solution, $S_{opt_\lambda} = \arg\min_{S \in \mathbb{S}}\{f_\lambda(S)\}$ and $S_{opt_\rho} = \arg\min_{S \in \mathbb{S}}\{f_\rho(S)\}$, respectively. As both $f_\lambda(S)$ and $f_\rho(S)$ depend on the various combinations between system partitioning and the task mapping, finding an optimal solution becomes a complex problem to solve. This problem can be tackled using heuristic approaches [19] or constraint programming [48]. The SaDSE framework takes a constraint programming approach.

4.2.3.2 Security Constraints

In addition to the traditional constraints, we define security-related constraints that must be fulfilled by the solutions. Only solutions that fulfill these constraints are considered to be secure solutions. Considering the process flow of the SaDSE framework, the calculation of secure solutions is performed before the computation of the solutions that are valid regarding the traditional constraints. Hence, security constraints are considered for the set of feasible solutions only. For selecting secure solutions, we introduced two constraints, the first one regarding the feasibility of the task mapping, the second one regarding the solutions attack susceptibility.

Security Mapping Constraint

Applying the security mapping constraint to the set of all solutions $\mathbb{S}$ results in obtaining its subset of security-wise feasible solutions $\mathbb{S}_{sf} \subseteq \mathbb{S}$. These security mapping constraints are formalized based on the rule sets for $\kappa^{SecOp} \mapsto \kappa^{SM}$ and $\kappa^{SF} \mapsto \kappa^{SM}$, respectively. In these mappings κ^{SecOp} describes the combination of security operations and environmental attributes using Boolean operators, κ^{SM} describes the combination of security mechanisms using Boolean operators, and κ^{SF} describes the combination of security functions using Boolean operators. Naturally, these combinations can also contain only one single element, respectively.

Considering the mapping of security operations to security mechanisms, $SecOp \mapsto SM$ must be checked for each solution $S(P, M) \in \mathbb{S}$. For each solution $S(P, M)$, the task mappings $M = (m_{t_1}, ..., m_{t_M})$ must be checked for compliance to the defined rules. For example, lets assume we have a rule $f(so_{enc}) = sm_{crypt}$ within the rule set, any security feasible solution $S(P, M) \in \mathbb{S}_{sf}$ must not contain a mapping $hwc_y = m_{t_x}$, where so_{enc} is not performed by t_x and sm_{crypt} is not supported by hwc_y. The same approach goes for the mapping of security functions to mechanisms defined by the rule set $SF \mapsto SM$. The framework determines, based on the rules in $SF \mapsto SM$, what solutions are allowed in $\mathbb{S}_{sf}$.

Attack Potential Constraint

The attack susceptibility is calculated for the solutions found to be feasible regarding their security mapping. This prerequisite reduces the number of solutions for which the attack success probability must be calculated. To find the set of secure solutions $\mathbb{S}_{sec} \subseteq \mathbb{S}_{sf}$ the attack scenarios, represented as BAGs or risk trees, must be calculated for each $S(P, M) \in \mathbb{S}_{sf}$. For both approaches, the designers must define certain thresholds the attack goals must not exceed, further characterizing each attack goal. For the BAG-based attack scenario description, this threshold would limit the unconditional probabilities of the attacker successfully reaching the attack goal. The thresholds used in the risk-tree-based approach have a different meaning. For the risk-tree-based approach, these thresholds limit the mean risk value (e.g., given as \$ amount) of the attacker exploiting the whole attack path resulting in the final attack goal. Hence, the threshold applied to the attack scenarios modeled with the risk-tree-based approach includes the risk induced by all attacks within the attack path, not only the risk of reaching the attack goal.

For the BAG-based attack description the attack potential constraints are defined as follows: For each solution $S(P, M) \in \mathbb{S}_{sf}$ the unconditional attack success probabilities $uasp_{ag}$ of the attack goals are checked against their thresholds $t_{uasp,ag}$. Only if the unconditional success probability of each attack goal is below its respective threshold ($uasp_{ag_1} < t_{uasp,ag_1}, \cdots, uasp_{ag_{|AG|}} < t_{uasp,ag_{|AG|}}$, with AG being the set of all attack goals), the attack potential constraint for the solution is met and hence $S(P, M) \in \mathbb{S}_{sec}$.

For the risk-tree-based approach, the constraints are similar to the BAG-based approach, with the difference that the thresholds of the distinct attack goals limit the risk of the whole attack path. Therefore, a solution is only secure if $mrv_{ag_1} < t_{mrv,ag_1}, \cdots, mrv_{ag_{|AG|}} < t_{mrv,ag_{|AG|}}$, where mrv_{ag_i} denotes the mean risk value of the attack path resulting into ag_i and t_{mrv,ag_i} its respective threshold.

4.3 Analytical DSE Implementation

For the implementation of the SaDSE framework we used the DeSyDe tool as a basis. The DeSyDe tool is open source[1] and has been described in various publications [48–51]. The DeSyDe tool itself builds on the description of the DSE problem using a constraint programming (CP) approach. It captures the input descriptions of the system's functionality and architecture within XML documents.

Finding a valid or optimal task mapping and system partitioning constitutes a typical combinatorial problem. CP is a well-proven technique to solve such problems. It can be applied for both checking a combination of predefined constraints or finding an optimal combination regarding the defined requirements. CP approaches model the problem's components as decision variables characterized by a domain of possible values. The decision variables are put into relationship with each other. This relationship is described in the form of constraints. For example, lets denote two variables $x \in \mathbb{Z} : [1, 10]$ and $y \in \mathbb{Z} : [1, 10]$, then the statement $x > y$ defines the constraint the values of the two variables must satisfy. Based on the problem described by the decision variables, the constraints, and the optional optimization goals, a constraint solver finds satisfactory and optimal solutions using intertwined steps of propagation, branching and searching [140]:

- **Propagation**: The propagation step removes all infeasible values within the domain of the decision variables regarding their constraints.
- **Branching**: The branching step builds up a search tree for the remaining possible value combinations of the decision variables' domains.
- **Search**: The search tree built up by the branching step is used for searching for new possible solutions again, satisfying all constraints. CP solvers employ various search algorithms, such as a depth-first search for finding valid solutions or branch-and-bound search for finding an optimal solution.

Heuristic algorithms are a great tool to solve complex problems for satisfactory and optimal so-

[1] https://github.com/forsyde/DeSyDe

lutions. However, the application of the CP approach comes with several advantages compared to heuristic approaches. In contrast to the heuristic approach, describing the problem space using CP does not split it into sub-problems, but captures it as a whole. This approach makes a formulation of interdependencies between the sub-problems unnecessary [49].

The representation of the traditional characteristics of the DSE problem is presented in [48, 49]. The DeSyDe tool represents the designed system's functionality as Synchronous Data Flow Graphs (SDFGs). Based on its convertibility into a single activation graph, an SDFG can be used to denote a task graph, as described in Section 4.2.1.1 [141]. According to [49], the DeSyDe tool's functional system description is based on SDFGs, denoted as $G(A, C)$. They consist of actors $a \in A$ (task nodes) and channels $c \in C$ (edges). As the DeSyDe tool allows the definition of multiple SDFGs mapped to the same system, the set $\mathcal{A}$ comprises the actors of all graphs, and the set $\mathcal{C}$ comprises the channels of all graphs. This SDFG representation gets mapped on a set of processing nodes P. The processing nodes $p \in P$ represent the system platform on which the actors are allocated. Each $p \in P$ is characterized by a set of modes $\mathcal{M}_p$. Each mode $M \in \mathcal{M}_p$ comes with distinct characteristics describing the processing node's power consumption, etc. The DeSyDe framework represents the communication system of the platform using a time division multiple access (TDMA) bus. The decision variables used by the CP model of the DSE problem capture the mapping and schedulability problem in the form of a mapping- and scheduling-aware graph (MSAG). This MSAG is used to calculate the delay (period) of an SDFG, which is performed using throughput (the inverse of the period) propagators [142]. The main decision variables capturing the problem are:

$\texttt{proc}_a$: captures the processor assignment of actor a and is defined for $\forall a \in \mathcal{A}$. This decision variable effectively represents the mapping $a \mapsto p$.

$\texttt{proc\_mode}_p$: captures the mode assignment of processor p and is defined for $\forall p \in P$. The domain of this variable is $\{0...|\mathcal{M}_p| - 1\}$.

$\texttt{wcet}_a(\texttt{proc}_a, \texttt{proc\_mode}_{\texttt{proc}_a})$: captures the WCET of an actor a mapped to a processor $\texttt{proc}_a$ running in a specific mode $\texttt{proc\_mode}_{\texttt{proc}_a}$. This variable is again defined for $\forall a \in \mathcal{A}$.

$\texttt{wcct\_s}_c$: Captures the worst case communication time (WCCT) for each channel $c \in \mathcal{C}$.

$\texttt{period}_g$: The iteration period expresses the overall delay of an SDFG represented as $G(A, C)$. The period is calculated as the inverse of an SDFG's throughput, obtained by the throughput propagator operated on the MSAG.

$\texttt{dynPower}_p$: represents the dynamic power consumption of a processor $p \in P$. It depends on the WCET, and, hence, on the mapping of actors to processors. $\texttt{dynPower}_p = \sum_{a \in \mathcal{A}}(\texttt{wcet}_a(\texttt{proc}_a, \texttt{proc\_mode}_{\texttt{proc}_a})) * \texttt{dynPow}(\texttt{proc\_mode}_{\texttt{proc}_a})$, with $\texttt{dynPow}(\texttt{proc\_mode}_{\texttt{proc}_a})$ returning the processor mode's dynamic power of the processor on which actor a is mapped.

$\texttt{statPower}_p$: represents the static power consumption induced by a processor $p \in P$.

$\texttt{system\_power}$: is calculated as $\sum_{p \in P} \texttt{dynPower}_p + \texttt{statPower}_p$ and represents the overall power consumption of a system solution.

The following sections describe what changes we implemented for the analytical part of the SaDSE framework, how we formalize the security constraints using the constraint programming approach, and the structure of the resulting solutions.

4.3.1 Realizing SaDSE in DeSyDe

We used the DeSyDe tool's representation of a system's functionality and architecture to map it with the SaDSE's approach to the functional and architectural perspective. The SDFG with its actors and channels represents the tasks graph of the functional perspective. The processors with their processing modes represent the hardware components of the architectural perspective. We extended the actor's representation with the data entities (including their SR) and the operations the actors perform. To reflect the security mechanisms, we adapted the processing modes accordingly. These adaptations were included in the XML representation and further used for calculating the security mapping constraints.

A range of adaptations were performed to reflect the security overhead affecting a solution's processing and communication delay and their power consumption. These adaptations affect the model of the communication system used in the architectural representation and the performance and power overhead that the used security mechanisms induce. These overheads change the calculation of the $\texttt{period}_g$ and the $\texttt{system\_power}$.

4.3.1.1 Communication System

The DeSyDe tool uses a TDMA bus as a base representation of the communication model for the system's platform. This representation is not feasible for capturing the physical communication channels used in our design (see Section 4.2.1.2). Hence, we extended this representation by introducing the physical communication channel system of the architectural representation, with its description added to the XML document describing the system's architecture. To introduce the physical channels into the CP model, we add the following set of decision variables $\texttt{pcc}_{p_x,p_y} \in PCC$, with $p_x, p_y \in P$ denoting the hardware components the physical channel connects. The inclusion of the communication system has three effects on the solutions found by the framework, namely the task mapping and the calculation of the WCCT.

The mapping of tasks to hardware components represented in the CP model with the variable $\texttt{proc}_a$ is constrained by the physical communication system linking the processors. To reflect this constraint, we introduced the decision variable $\texttt{conn}_{\texttt{proc}_a}$ into the CP model. This variable is defined for all processors connected to $\texttt{proc}_a$, including $\texttt{proc}_a$ itself. Hence, the constraint $\texttt{proc}_{\texttt{next}_a} \neq \neg\texttt{conn}_{\texttt{proc}_a}$ restricts the mapping of a task $\texttt{next}_a$ to a hardware component connected to $\texttt{proc}_a$, where $\texttt{next}_a$ is the successor task of a.

The communication delay induced by the transmission via these channels is added to the calculation of the WCCT by introducing further decision variables of the CP model. The decision variable $\texttt{ts}(\texttt{pcc}_{\texttt{proc}_a,\texttt{proc}_{\texttt{next}_a}})$ captures the transmission speed of the $\texttt{pcc}$ connecting the hardware components on to which a and $\texttt{next}_a$ are mapped. It is 0 if $\texttt{proc}_a = \texttt{proc}_{\texttt{next}_a}$. The decision variable $\texttt{ds}(c_{a,\texttt{next}_a})$ captures the overall data size exchanged between on the channel c connecting a and $\texttt{next}_a$. The overall communication delay is, therefore, computed as $\sum_{a\in\mathcal{A}}(\texttt{ds}(c_{a,\texttt{next}_a}) * \texttt{ts}(\texttt{pcc}_{\texttt{proc}_a,\texttt{proc}_{\texttt{next}_a}}))$. This delay is added to the WCCT in the CP model, and, hence added to the calculation of $\texttt{period}_g$.

Furthermore, the power consumption induced by the communication system is added to the calculation of $\texttt{system\_power}$. The communication system's power consumption is calculated as $\sum_{a \in \mathcal{A}}(\texttt{ds}(c_{a,\texttt{next}_a}) * \texttt{pc}(\texttt{pcc}_{\texttt{proc}_a,\texttt{proc}_{\texttt{next}_a}}))$, where $\texttt{pc}(\texttt{pcc}_{\texttt{proc}_a,\texttt{proc}_{\texttt{next}_a}})$ gives the dynamic power consumption of the physical channel connecting the hardware components to which a and $\texttt{next}_a$ are mapped. Similar to $\texttt{ts}$, it is 0 if a and $\texttt{next}_a$ are mapped to the same component. With this computation approach the power consumption of the communication bus systems is based on the time it is actively used, which depends on the transmission speed and the amount of data being transmitted. This fits general observations on the power consumption of communication bus systems, such as presented in [143].

4.3.1.2 Integration of Security Aspects

To integrate the security characteristics of our approach into the exploration performed by the SaDSE framework, we extended the CP model with the security mechanisms $sm \in SM$, the security assets $sa \in SA$, and the security functions $sf \in SF$, describable for the system under design. These extensions are used to select the security mechanisms used for protecting the system against potential attackers. The CP model performs the calculation of the overhead on the performance and the power consumption induced by the selected mechanisms.

The following paragraphs explain how the security aspects defined in Section 4.2.2.2 were integrated into the CP model, and how they influence the calculation of the system's performance and power consumption. These aspects comprise the security assets, the security operations $SecOp$, security functions SF, and the security mechanisms SM. These aspects were integrated into the CP model using decision variables.

Security Mechanism Selection

The calculation of the performance and power overhead induced by the security mechanisms depends on the security mechanisms the solution uses to protect security assets. The selection depends on whether the system under design defines the usage of possible security functions. If the system defines no security functions, the selection solely depends on the security assets that need protection. What security mechanisms protect which assets are defined by the used security operations. If the system defines a range of security functions to be selected from, the selection of the mechanisms depends on the selected security functions.

This selection is described in the CP model using the decision variable $\texttt{sm}_{a,sa} = \texttt{sec\_mech}(\texttt{so}_a(sa), \texttt{proc}_a, \texttt{proc\_mode}_{\texttt{proc}_a}, sa)$, where $\texttt{so}_a(sa)$ defines what security operations are used by a to protect sa, and $\texttt{sec\_mech}$ returns the security mechanism to realize so_a. The variable $\texttt{sm}_{a,sa}$ is defined for $\forall a \in \mathcal{A}, \forall sa \in SA_a$, with SA_a denoting the set of security assets a interacts with. In case that the system under design defines a set of security functions SF to be selected by the framework, the selection of the security mechanisms is redefined by the decision variable $\texttt{sm}_{a,sa,\texttt{sf}} = \texttt{sec\_mech}(\texttt{sf}(\texttt{so}_a(sa)), \texttt{proc}_a, \texttt{proc\_mode}_{\texttt{proc}_a}, sa)$, where $\texttt{sf}(\texttt{so}_a(sa))$ denotes the security function realizing the mapping of $\texttt{so}_a$ and sa. The decision variable $\texttt{sm}_{a,sa,\texttt{sf}}$ is calculated for $\forall a \in \mathcal{A}, \forall sa \in SA_a, \forall sf \in SF$. These decision variables are used to calculated the performance overhead.

Security Performance and Power Overhead Calculation

The security overhead on performance and power depends on the selected security mechanisms and what data entities they secure. A security mechanism's overhead further depends on whether its performance is characterized in relation to the data size or as an absolute value (see Section 4.2.2.3). The selection of the security mechanisms is described with the decision variables $\mathtt{sm}_{a,sa}$ and $\mathtt{sm}_{a,sa,\mathtt{sf}}$, depending on whether or not the system selects from a predefined set of security functions. Based on these decision variables, the security performance overhead is calculated as $\sum_{a\in\mathcal{A},sa\in SA_a} \mathtt{sm\_perf}_a(sm_{a,sa}, size_{sa})$ or $\sum_{a\in\mathcal{A},sa\in SA_a,sf_a\in SF_a} \mathtt{sm\_perf}_a(sm_{a,sa,sf_a}, size_{sa})$, respectively, where $size_{sa}$ defines the size of the security asset. Similar to the calculation of the performance overhead, the power overhead is calculated as $\sum_{a\in\mathcal{A},sa\in SA_a} \mathtt{sm\_pwr}_a(sm_{a,sa}, sa_{size})$ or $\sum_{a\in\mathcal{A},sa\in SA_a,sf_a\in SF_a} \mathtt{sm\_pwr}_a(sm_{a,sa,sf_a}, size_{sa})$, respectively. The performance overhead is added to the calculation of the overall system's delay $\mathtt{period}_g$, the power overhead to $\mathtt{system\_power}$.

4.3.1.3 Security Constraints

The calculation of the security constraints builds on two separate steps. First, the calculation of the security mappings' validity. Second, the calculation of the system's attack success probability or security risk. The first step yields the security-wise feasible solutions, on which the second step calculates the secure solutions. The first step is based on security mapping constraints, the second step is based on attack potential constraints. The implementation of the security constraints ensures that all selected security operations and security functions used by the tasks are supported by the hardware components they are mapped to and that no attack goal exceeds its threshold on attack success probability and mean risk.

Security Mapping Constraints

The security mapping constraints prohibit the mapping of tasks to hardware components which do not support their operations on security assets. This mapping is modeled in the CP model with the decision variable $\mathtt{smap}_a = \mathtt{sec\_map}(a, \mathtt{proc}_a, \mathtt{proc\_mode}_{\mathtt{proc}_a})$, where $\mathtt{sec\_map}$ performs the rule-based check of the mapping $SecOp \mapsto SM$. When considering the selection of security functions, the mapping is represented by the decision variable $\mathtt{smap}_{a,sm} = \mathtt{sec\_map}(a, sm, \mathtt{proc}_a, \mathtt{proc\_mode}_{\mathtt{proc}_a})$, checking the mapping against the rules described by $SF \mapsto SM$. Both $\mathtt{smap}_a$ and $\mathtt{smap}_{a,sm}$ are set to 1 if the mapping adheres to the defined rules, and 0 if not. The constraints $\mathtt{smap}_a > 0, \forall a \in \mathcal{A}$ and $\mathtt{smap}_{a,sm} > 0, \forall a \in \mathcal{A}, \forall sm \in SM$ restricts all task mappings considering the hardware component's security mechanisms.

Attack Potential Constraints

The attack potential constraints are calculated based on the attack models captured as BAGs or risk trees. For implementing the BAG we used the *Dlib* library[2] for the realization of the underlying Bayesian networks. The risk trees were integrated using the implementation of *RISKEE*, provided by Krisper et al. [107]. The BAGs and risk trees are provided to the framework in the form of XML files.

The BAG-based approach supports the calculation of the attack success probability with and without the definition of the security functions. Without the definition of these functions, the framework changes the asp values of the BAGs' attack nodes with the iv values depending on the mappings $\mathtt{proc}_a, \mathtt{proc\_mode}_{\mathtt{proc}_a} : \forall a \in \mathcal{A}$. With the definition of the security functions,

[2] `http://dlib.net/`

the framework adds the corresponding attack nodes to the BAGs and changes the asps of the added attack nodes according to the ivs based on the task mappings. Hence, the changes for the BAG creation depends on $\texttt{proc}_a, \texttt{proc\_mode}_{\texttt{proc}_a}, sf : \forall a \in \mathcal{A}, \forall sf \in SF$. Furthermore, the framework performs a mapping of the keys used by the sf to the hardware components. For this allocation, the framework checks $\texttt{proc\_mode}_{\texttt{proc}_a} : \forall a \in \mathcal{A}$.

To integrate the attack potential constraint based on the asp of the BAG's attack goals, we added the decision variable $\texttt{asp}_{ag} = \texttt{calcBAG}(ag)$ to the CP model, where $\texttt{calcBAG}$ returns the unconditional attack success probability of ag. This variable is constrained by $\texttt{asp}_{ag} < \texttt{t\_asp}_{ag}$, where $\texttt{t\_asp}_{ag}$ defines the threshold of the attack goal ag. Both $\texttt{asp}_{ag}$ and $\texttt{t\_asp}_{ag}$ are defined $\forall ag \in \mathcal{AG}$, where $\mathcal{AG}$ defines the set of all ag of all BAGs.

The security risk constraint was integrated into the CP model in a similar way. The influence of the iv on an attack node's asp is calculated for all attack nodes of all risk trees based on the task mappings. The risk-based attack potential constraint is modeled in the CP model as $\texttt{mrv}_{ag} = \texttt{calcRISK}(ag)$, with $\texttt{mrv}_{ag}$ denoting the mean risk value of ag, and $\texttt{calcRISK}$ calculating this mean risk. This variable is constrained by $\texttt{mrv}_{ag} < \texttt{t\_mrv}_{ag}$, where $\texttt{t\_mrv}_{ag}$ defines the mean risk value threshold of the attack goal ag. Again the variables are defined for $\forall ag \in \mathcal{AG}$.

We optimized both the $\texttt{asp}_{ag}$ and the $\texttt{mrv}_{ag}$ computation of the attack potential constraints by introducing a task mapping limitation based on the resulting asp or mrv values of the attack scenarios. This optimization performs an ordering of $\texttt{proc\_mode}_{\texttt{proc}_a}$, ascending according to their iv. For each mapping $\texttt{proc}_a$, the framework only calculates the $\texttt{asp}_{ag}$ or $\texttt{mrv}_{ag}$, permuting $\texttt{proc\_mode}_{\texttt{proc}_a}$, until the constraint on $\texttt{asp}_{ag}$ or $\texttt{mrv}_{ag}$ is not met for at least one ag. Any further calculation based on consecutive $\texttt{proc\_mode}_{\texttt{proc}_a}$ can be automatically rendered insecure.

4.3.2 Resulting Solutions

The solutions found by the SaDSE framework are provided to the designers in the form of an `xls` file. Each solution comprises the following information:

Task mapping: task to processor mapping, given in the form denoted by $\texttt{proc}_a$.

System partitioning: hardware component selection, given in the form denoted by $\texttt{proc\_mode}_{\texttt{proc}_a}$.

Period: Overall system delay of the solution, given as $\texttt{period}_g$.

Power consumption: Overall power consumption of the solution, given as `system_power`.

Security mechanism selection: The output contains the used sm for each solution.

Security function selection: If the system under design defines sf to select from, the framework adds the selected sf and the used keys, including their component allocation to the output.

Based on the output solutions, the designers can reevaluate the suitability of the selectable hardware components, their security mechanisms, the selectable security functions, and the key placement, etc., for the overall system design. Furthermore, they can evaluate the suitability of the hardware component's security mechanisms on the attack mitigation, considering the attacker's probability to bypass all security measures, and the risk induced by a successful attack.

4.4 Simulation-based DSE Approach

The solutions found by the analytical part of the DSE are used to generate a system simulation from them, as depicted in Figure 4.1. Figure 4.10 gives a more detailed overview on the simulation-based part of the SaDSE framework. The simulation-based part includes the solutions found by the analytical part of the SaDSE, the architectural description, the functional description, and a configuration file that describes the use case for the simulation. The solutions, the architectural and functional descriptions are used by the component factory to generate the device components, as well as synthesize and instantiate the devices. These devices are used in the simulation environment, simulating the devices' communication via external channels and according to a predefined networking behavior. This network simulation is analyzed regarding the network's packet loss, its throughput, etc. Furthermore, the simulation can be used to show the device's internal timings, which can be further analyzed to identify performance bottlenecks. This analysis allows the evaluation and the further optimization of the device's internal behavior, e.g., usage of alternative protocol designs.

The following sections explain the transformation of the solutions found in the analytical part of the DSE to the system simulation. Furthermore, they explain the implementation of the simulation-based part of the SaDSE.

4.4.1 Solution Transformation to System Simulation

The SaDSE framework integrates the analytical part and the simulation-based part by generating system simulation models from the solutions found by the analytical DSE. The basic idea is to use the different views of the system design combined with the solutions found by the analytical part of the DSE to generate simulation models of the devices they represent. Based on the tool's configuration, a number of these devices are instantiated. Their interactions are simulated within an environment defined by the designer.

The focus here lies on the evaluation of the interaction of the devices in different environments. The internal communication within the devices themselves can already be assessed using the analytical part of the SaDSE framework. The interesting part of the evaluation is how the selection of the security mechanisms to counter the attacks, influences the behavior of the overall system, consisting of the multitude of devices. The following sections give an overview of the simulation models generated by the simulation-based part of the SaDSE framework, how they are composed to form the device models, and how they are connected. The SaDSE tool focuses on a high-level system simulation. It shows the influence of the task mappings, the selected security mechanisms, and the chosen communication channels on the system's performance and power consumption. Thereby, the simulation model reflects the task mappings WCETs, the hardware components' power consumption, and the communication channels' transmission speed and energy dissipation. Furthermore, the simulation models allow the designer to integrate function blocks, extending the behavioral model of the designed system. The simulation-based DSE was realized using SystemC.

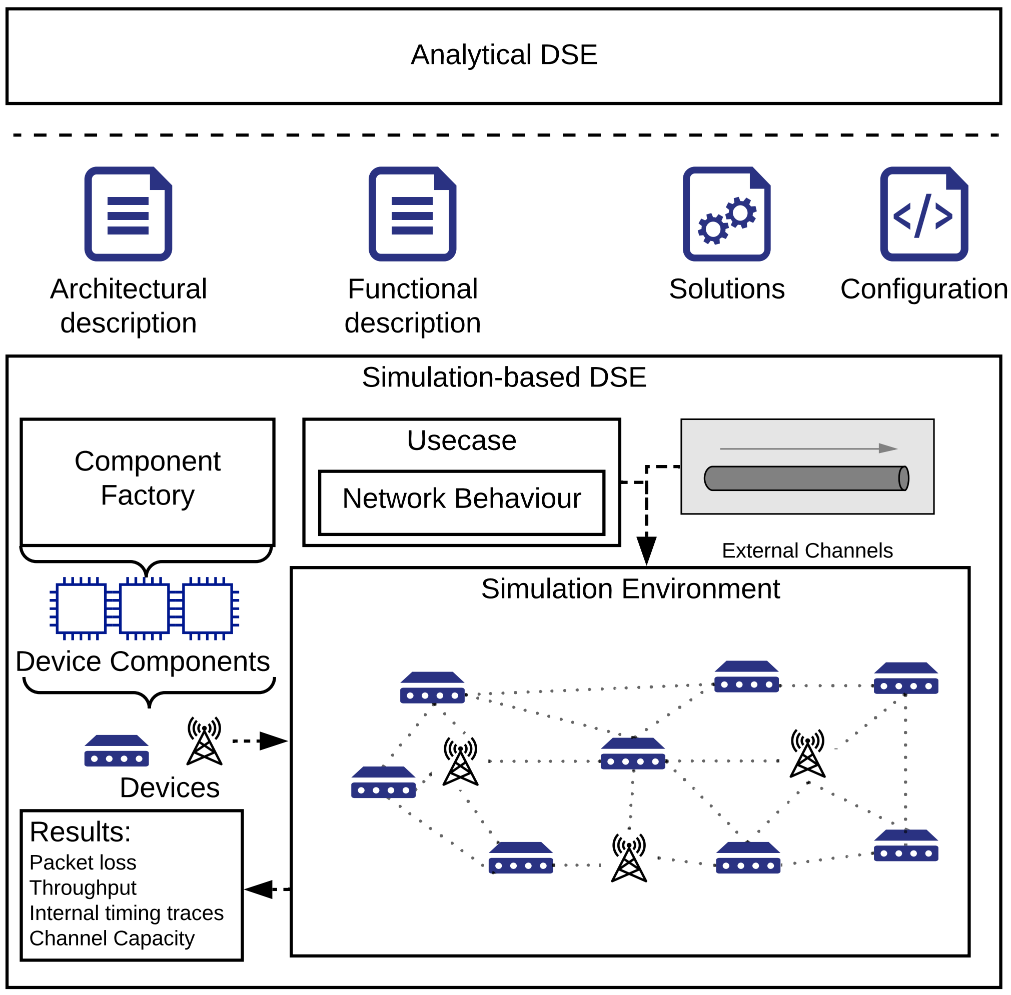


Figure 4.10: Simulation-based DSE part of the SaDSE tool.

4.4.1.1 Device Architecture

A device's architecture is composed of the hardware components. The analytical part of the SaDSE tool selects the appropriate components for the system design. To inform the component factory about the affiliation between the device type and the hardware components, the designers must add this information to the component's description. Also, the physical communication system used for the device's internal communication must be affiliated with the corresponding devices. The component factory generates the simulation models of the devices, their hardware components, and their internal communication system based on this description. Furthermore, it connects the device's internal components to the communication system according to the architectural description. The communication system consists of physical communication channels (`PCC`s). Each `PCC` is characterized by its transmission speed, as defined in the architectural perspective, and offers an interface (`PCC_IF`), to which other components can connect. Figure 4.11 shows an example system model. It consists of one device containing three hardware components (`HWC`s), linked with two physical communication channels. Furthermore, the device contains one external port used to connect it to an external channel.

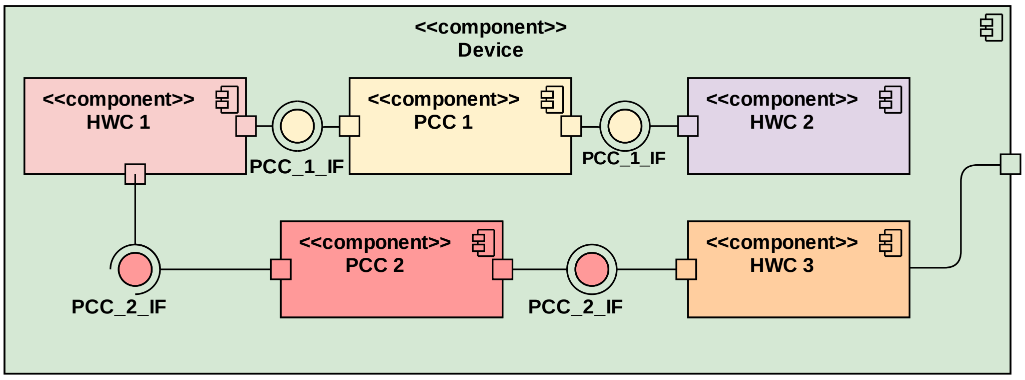


Figure 4.11: Example of an architectural simulation mode generated by the component factory. What `HWC`s are generated is defined by the solution found by the analytical DSE

The component factory also integrates the security mechanisms supported by the different hardware components. Depending on what tasks use which security mechanisms, this usage is also represented in the simulation. This information is taken from the solutions found by the analytical part of the SaDSE framework.

4.4.1.2 Functional Blocks

Similar to the generation of the architectural simulation components, the component factory also generates simulation components representing the system's behavior. The component factory uses the functional view on the system to generate the functional blocks representing the tasks of the task graph. It connects them according to the task graph's edges using communication channels. These communication channels are used to transport the data entities the tasks operate on. The simulation system further identifies what data entities are generated by what tasks inspecting the functional view's data flow.

Based on the solutions found by the analytical part, the SaDSE framework represents the task of hardware component mapping in the generated system simulation. The component factory must reconnect each task's ports depending on the mapping of the tasks it is connected with. Figure 4.12 shows the generated system simulation with the tasks being mapped to the `HWC`.

The mapping decides if a communicating task passes data entities internally within the same component, or via the `PCC` connecting the `HWC`s, the tasks are allocated on. The mapping further defines the worst-case execution time (WCET) of the tasks. This execution time must be reflected in the system simulation. Furthermore, the `PCC` calculates the transmission delay based on its transmission speed, and the size of the data entities exchanged via the distinct `PCC`. In addition to the WCET and the communication transmission delay, the simulation model also integrates the execution delays caused by the chosen security mechanisms. This overhead is added to the WCET of each task. The overhead depends on the task's usage of security mechanisms, as described in the solution found by the analytical part.

The simulation-based DSE also allows the definition of recurring task execution. For example, `Task 1` in Figure 4.12 could be triggered every $15\mathrm{ms}$ for ten consecutive times. This repetition

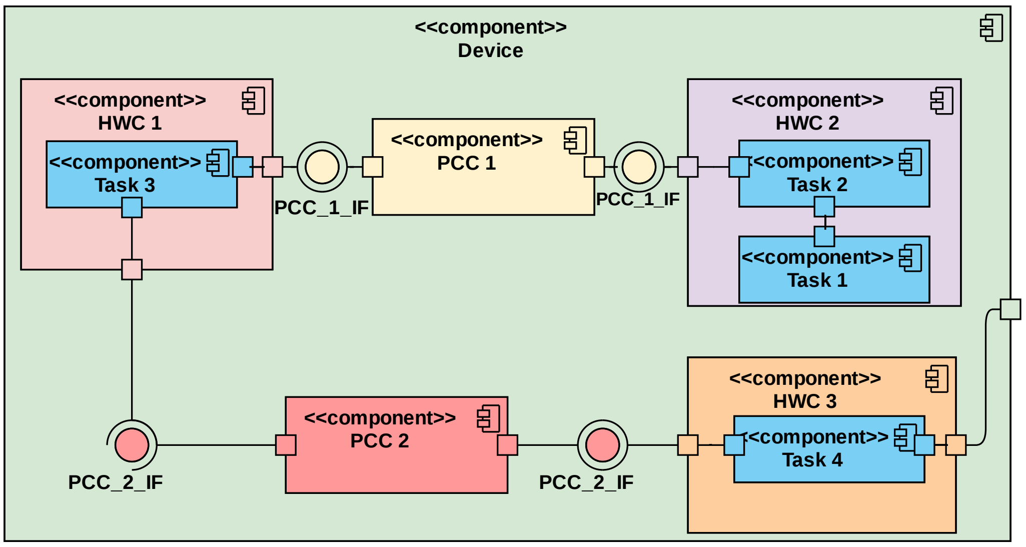


Figure 4.12: Example architectural simulation model with the tasks mapped to the distinct `HWC`s, generated by the component factory as defined by the solution found by the analytical DSE.

would lead to the execution of the whole system behavior described by tasks `Task 1` to `Task 4`, ten times in a row. This recurring execution can be defined by the designers in the system's task graph. These recurring executions might lead to concurrency issues considering multiple tasks being executed on the same hardware component. This concurreny mismatch can lead to a simulation behavior not reflecting the actual solution found in the analytical DSE, as tasks sharing the same hardware component could trigger their parallel execution. As each hardware component represents a single core, this behavior is prevented. The SaDSE framework's simulation part guarantees the subsequent execution by using an additional execution queue for each hardware component. This execution queue is used to check if the component already executes a task, and, if so, to queue up future task executions. Before a task is executed, it checks the execution queue of its host and adds itself to it. If no task is running on the component, the task is executed right away. Otherwise, the task waits until the component signals its execution. After a task has been successfully executed, its entry is removed from its host's execution queue.

4.4.1.3 Device Simulation

The component factory generates the devices with their selected hardware components and the mapped tasks and integrates them into the overall system simulation. Using this device simulation, their communication behavior can be evaluated.

Device Communication

To simulate the communication behavior of the overall system, its devices are connected using one or multiple external communication channel models. Depending on the system under design, an external communication channel could be, e.g., a BLE or an Ultra Wideband (UWB) channel. Figure 4.13 shows multiple devices connected to an external communication channel. The devices use the channel by interacting with the interface (`Ext_Chan_IF`) it provides. The external communication channel contains a routing table that stores information about the connection between the single devices and what messages are currently being transmitted via the channel. What device communicates with what device is configurable by the designer.

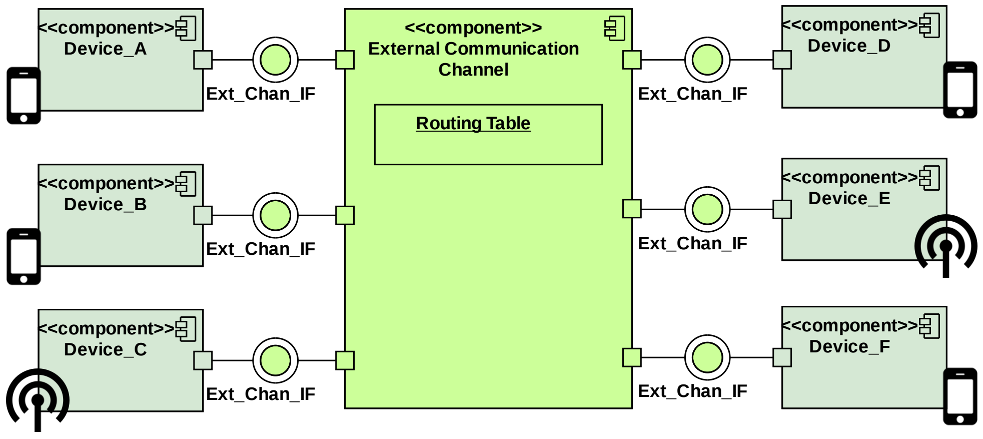


Figure 4.13: Example system simulation model consisting of multiple devices, connected to one another via an external communication channel. The devices are characterized as mobile or stationary devices. This information is used by the environmental simulation.

The internal logic of the channel handles the message exchange between the single devices, considering potential collisions, packet errors, etc. To simulate these effects, the external channel's implementation relies on simulation of the physical channel, as well as the system's environment.

Environment Simulation

The simulation of the physical environment consists of two parts which interact with each other: The simulation of the physical properties of the external communication channel and the dynamic changes of the environment. The simulation of the environment provides input to the external channel. This input consists of the device placement (stationary or mobile), the distance between them, and the obstacles in between. The external channel uses these inputs from the environment simulation for the calculation of the physical communication properties. The physical channel simulation output is then used to determine whether or not the receiving device correctly received a sent message. Considering the overall system simulation, this information allows the determination of channel throughput, information update rates, etc.

4.5 Simulation-based DSE Implementation

We realized the system-based DSE approach using the SystemC simulation environment. SystemC is a system simulation framework which comes as a C++ library. With SystemC, designers can describe a system from an abstract system level down to the register-transfer level. Especially the TLM-based system description is optimally suited for the generation of the simulation models for the simulation-based DSE part of the SaDSE framework. This modeling approach allows the integration of the performance characteristics taken from the provided solutions, denoting them as execution delays. Furthermore, the usage of the TLM models also allowed us to add certain details to distinct `PCCs` within the devices by modeling the used protocol stack and security protocols.

Based on the TLM model, we designed an improved protocol for the integration of an embedded system and a secure element [3].

Figure 4.14 gives a detailed view on the SystemC models generated by the component factory for the system simulation. It shows the composition of `Device`, the `HWC`, and the `Task` class. Furthermore, it shows that all these classes contain ports connecting them to different kinds of communication channels. These channels all derive from the same interface `CC_If`. The `CC_If` interface provides the routing table holding the information entries (`RT_Entry`) about the device connections. Based on the mapping of the `Tasks` to the `HWCs` and the association of the `HWCs` to the `Devices`, the component factory uses either an instance of the `Log_Channel`, the `PCC` or the `Ext_Chan` class to realize `CC_If`.

The `Task` class is simply generated as a skeleton container. To simulate its behavior, the designers must add a corresponding implementation to it. Supporting a generic approach, each task contains a function pointer with which the function implementing the behavior can be called. Furthermore, the system simulation provides a global lookup table that links the task name provided by the functional description and address of a function in the SystemC model. The `DataEntity` class serves as a container class for passing the information between the tasks and their behavioral implementation.

Furthermore, the `Task` class contains all used security mechanisms (`SM`), provided by the `HWC` to which it is mapped. The `SM` class contains the information of the `DataEntity` it is used on, and its performance overhead. Its overhead is calculated based on the entity's size. The generation of the `SM` instances and their assignment to the individual `Task` instances is performed based on the input solutions.

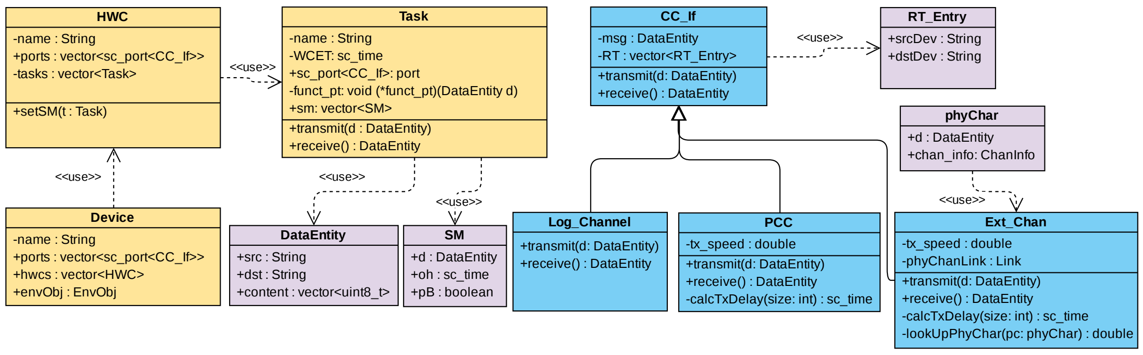


Figure 4.14: SystemC components generated from the component factory for the system simulation.

The components generated for the system simulation are linked with the environmental simulation and the physical channel model.

4.5.1 Environment Simulation Implementation

The environmental simulation focuses on indoor localization use cases. Its main purpose is to simulate indoor environments in which numerous different devices need to be localized. This environmental simulation is realized by a SystemC-based simulation in which mobile nodes are represented as moving objects, and anchor devices are represented as stationary objects. Each device in the system simulation is linked to either a mobile or a stationary object. This link depends on the given configuration. The localization is performed using UWB technology, passing ranging packets between the anchor devices and the mobile nodes and taking time stamps when transmitting and receiving them. The reception of these packets and the time stamp accuracy heavily depends on the line of sight (LOS) between the stationary anchor and the mobile device. This LOS is determined by performing a ray-tracing analysis from the transmitting to the receiving device. The LOS condition (LOS or no-LOS (NLOS)) is used as input for retrieving the packet bit errors from the physical channel simulation. A detailed explanation of the movement simulation is given in [144].

4.5.2 Connection to Physical Channel Simulation

The channel model for simulating the physical parameters of the UWB-based communication was provided by our partnering Institute of Microwave and Photonic Engineering and implemented by David Veit et al. [145]. Using their model, we precalculated channel models for numerous situations with different distances between the communicating devices, packet interference, LOS conditions, etc. Each simulation returned important information about the received packet's bit error rate. This information was stored in a database. During the system simulation, the SaDSE framework performs a lookup in this database for each UWB packet exchange, searching for the physical channel attributes, providing the characteristics on the LOS condition, the distance, and the message occupancy within the channel. The environment simulation provides the LOS condition and the distance between the communicating devices. The channel usage is provided by the SystemC model of the UWB channel, which tracks the ongoing packet transmission. The channel occupancy provides information about transmitted packets interfering with each other. The system simulation uses the information returned by the physical channel simulation to check whether the transmitted packet could have been received or not. Based on this information, the SaDSE framework calculates the channel throughput and the distance estimation rate of the designed localization system. To get the information from the physical channel simulation, a lookup into the database, storing the precalculated physical channel transmission must be performed. To speed up this process, the SaDSE framework stores already retrieved bit error rates according to the distance information, LOS condition, and packet interference into a lookup-table. All subsequent lookups with the same input parameters can retrieve the physical channel information from the lookup-table, thus, saving simulation execution time. The BLE channel's implementation in the SaDSE framework focuses on the transmission delay. This delay is calculated according to the average packet reception delay presented in the channel model by Spörk et al. [146].

5 Evaluation

We evaluated the SaDSE framework using two use cases. These use cases stem from our project partners NXP Semiconductors Austria GmbH and CISC Semiconductor. For both use cases, we modeled the systems' behavior and potential architectural realization with the SaDSE tool. The evaluation of the SaDSE framework checks the accuracy of its found solutions. Thereby we compare the solutions found by the SaDSE tool with the system designs independently chosen by the system experts. We extended this qualitative evaluation with the evaluation of the SaDSE framework's performance. The performance of the framework is compared to the original DeSyDe framework's performance, showing the overhead produced by the security constraint consideration during the DSE. We evaluated the framework's simulation-based capabilities by extending one of the use cases towards a system simulation with dynamic environments. Furthermore, we show how the system simulation can be used to design new communication protocols used within the found system designs.

The following sections describe the use cases from both the behavioral and architectural descriptions. We outline the security rating of both the potential attack scenarios and the security mechanisms' protection. The last section describes the performance overhead evaluation.

5.1 Use Case Evaluations

We used the SaDSE framework to model two use cases from their behavioral and architectural perspective. This description includes the modeling of potential attack scenarios and the systems' security protection mechanisms. The possible implementations and their performance and power consumption were estimated with the support of the project partners' system architects. The security mechanisms overheads were estimated using performance and power consumption measurements published in related research projects.

5.1.1 Security Rating

To evaluate the security of the possible solutions found by the SaDSE framework, the potential attack scenarios, and the protection capability of the security mechanisms must be rated. The rating of the attacks' potential to threaten distinct security assets within the system builds on the widely used CVSS. The rating of the security mechanisms' protection capabilities is based on the CC certifications and system expert knowledge.

5.1.1.1 Attack Rating

The attack rating for assessing the potential threat of the attack scenarios builds on the CVSS version 3.1 [99]. The CVSS score is composed of three subscores: the *base*, *temporal*, and the *environmental* subscore. These subscores are calculated using various metrics. The *base* metrics contain exploitability and impact metrics. Rating the probability of the attacker successfully conducting an attack step is based on a likelihood estimation. To obtain this estimation, we rely on the exploitability metric of the CVSS. These base metrics consist of the attack vector AV, the attack complexity AC, the privilege required PR, and the user interaction UI metrics. These metrics describe the difficulty of a performed attack.

In addition to the exploitability metrics, the probability of an attack step also captures the attacker's motivation for performing the attack. This motivation is largely determined by the impact of the successful attack execution on the targeted system. This impact is captured in the CVSS using the security requirement SR metric.

The probability of an attack step being successfully executed is captured in its attack success probability asp. This value is calculated according to Equation (5.1). The exploitability metrics are rated with values ranging from 0 to 1. The SR value ranges from 0 to 2.1. With 2.1 being the maximum, the asp cannot exceed 1.0, which is the allowed maximum value.

$$asp = AV * AC * PR * UI * SR \tag{5.1}$$

The CVSS supports security analysts by a textual description of the levels selectable for each metric. This description provides the analysts with a reference point for their rating. Each level is assigned a distinct rating value. Table 5.1 gives a short description of these levels and their rating values. To support the analysts with a more detailed rating of these metrics, the CVSS was extended with custom levels. These additional levels are described in [99].

5.1.1.2 Protection Rating

The protection rating offered by the hardware components is based on the description of the used security mechanisms. These descriptions are, e.g., based on existing CC certifications. Independent of the protection rating's basis, the security mechanisms increase the complexity of all attacks aiming at tasks mapped to the component providing this mechanism. This increase of the complexity is reflected in Equation (5.1) by exchanging AC with AC_{iv}. AC_{iv} is the attack complexity increased by the implementation vulnerability iv of the allocating hardware component. The AC_{iv} is calculated as $AC_{iv} = AC * iv$, with $0 < iv \leq 1$ and $AC_{iv} \leq 0.01$. Hence, the modified attack complexity cannot become less than infeasible. The range of iv and AC_{iv} represents the fact that a system is only secure as long as no security incident was reported. Hence, there is always a small possibility that an attacker can breach a system's security.

The CC defines, additionally to ToEs, STs, and PPs (see Section 2.3), categories of attack potential. Experts determine the attack potential during the vulnerability assessment of a ToE. The attack potentials are categorized as basic, enhanced-basic, moderate, high, and beyond-high attack potentials. The attack potentials result from the attacker's expertise, the knowledge of the ToE, used equipment, elapsed time until a vulnerability was identified, and the timeframe during which the attack is executable. The attack potential is used by the CC certification further to describe

Table 5.1: Ratings of AV, AC, PR, UI and SR based on the CVSS.

Level	Description	Rating
	Attack Vector	
Network (N)	Attack victim is remotely exploitable via the network stack - can be multiple hops away.	0.85
Adjacent (A)	Attack victim is bound to network stack. Attack is limited on protocol level. Attacker has access to same physical / logical network.	0.62
Local (L)	Victim is not bound to network stack. Attacker must access system locally.	0.55
Physical (P)	Attacker needs physical access to victim.	0.2
	Attack Complexity	
Low (L)	No special access conditions. Attack has repeatable success.	0.77
Medium (M)	Attacker needs domain knowledge of the attacked system.	0.5
High (H)	Attack needs measurable effort in preparation. Attacker gathers knowledge about system, prepares target environment, injects attack.	0.44
Very High (H+)	Attacker needs expert domain knowledge to launch an attack. Additionally, the attacker needs to scan the environment to as the attack is timing based.	0.2
Secured (S)	The attacker must bypass the system's protective mechanisms. A considerable effort must be spent to find security backdoors. Vulnerabilities are not improbable.	0.05
Infeasible (I)	A succesfull attack on the system would need vast computational effort for breaking the system's security. No vulnerability known for circumventing the system's protection.	0.01
	Privileges Required	
None (N)	No prior authentication needed by the attacker.	0.85
Low (L)	Privileges for basic user capabilities needed. Only access to non-sensitive resources needed.	0.62
High (H)	Attacker needs privilege to access significant control over victim.	0.27
	User Interaction	
None (N)	Exploit does not need any interaction from the user	0.85
Required (R)	Prior user actions are required for successfully performing the attack	0.62
	Security Requirements	
Not Defined (X)	Insufficient information about the impact of the successful attack	1
Very High (H+)	Successful attack is likely to have a catastrophic effect on the attacked system and its users	2.1
High (H)	Successful attack is likely to have a high impact on the attacked system and its users	1.5
Medium (M)	Successful attack is likely to have a serious effect on the attacked system and its users	1
Low (L)	Successful attack is likely to have only a limited effect on the attacked system and its users	0.5

the EALs of the certified ToEs. Each EAL (except EAL1) assures that the evaluated ToE is capable of protecting its security assets against the distinct categories of attack potentials. EAL2 and EAL3 is secure against basic attacks, EAL4 against enhanced basic attacks, EAL5 against moderate attacks, EAL6 and EAL7 against high potential attacks [147].

Based on these attack potential descriptions and the EALs' protection, the iv was chosen for the distinct hardware components: for EAL6 and EAL7 the iv is 0.01; for EAL5 the iv is 0.1; for EAL4 the iv is 0.2; for EAL3 and EAL2 the iv is 0.4; for EAL1 the iv is 0.6. For none CC certified hardware components we performed an assessment of their security mechanisms and estimated how likely it is for the potential attacker to find vulnerabilities.

5.1.2 Secure Sensor System

The secure sensor system use case was a result of the STIP project which was performed together with NXP Semiconductors Austria GmbH and CISC Semiconductor. The purpose of the secure sensor system is the continuous collection of sensor data using multiple embedded sensors. These sensors send the sensed data to a central gateway that accumulates it. This central gateway then sends the accumulated data to the analysis server. The server checks the data and, depending on the sensed data, takes control actions. The use case described here mainly builds on our work described in [36].

The secure sensor system design was used as a use case to show the usability of the SaDSE framework. Thereby, we used the SaDSE framework to model the system's functionality and describe the possible hardware components with which to realize the system's implementation. Furthermore, we described potential attack scenarios, as well as the system's security assets and security mechanisms. We were also interested in the influence of using different secret keys and placing them on different hardware components. The following sections give a detailed explanation of the modeling approach and the solutions found by our tool.

5.1.2.1 System Description

The system design's behavior and architecture are described using the SaDSE framework's functional and architectural perspective. To show the influence of the overall system's design on its security, two different functional perspectives were used as inputs for the SaDSE framework. This difference in the functional perspective results in two variants. In variant I, the gateway performs a prefiltering of the sensor data. In variant II, the gateway only forwards the accumulated sensor data directly to the analysis server.

Functional Perspective

We described the system's behavior using the functional perspective of the SaDSE framework. This description comprises the task graph and the data entities by the sensor system. Figure 5.1 shows the task graph, including the two variants for the use case. The task graph describes the sensor node getting configured by the gateway module. After the successful configuration, the gateway activates the sensor node, upon which the sensor starts collecting environment information. The sensor sends the sensed information to the gateways. At the gateway, the sensor data is either filtered or merely forwarded and sent to the analysis server. These different scenarios are described in Figure 5.1, marked as variant I and variant II. The analysis server receives the sensor data and stores and analyses it. Based on the analysis, the server takes suitable actions.

Table 5.2: Data entities used by the secure sensor system and their security properties: confidentiality (conf.), integrity (int.), authenticity (auth.).

	config	status	activation	sensor msg	sensor data	action cmd
conf.		x		x	x	x
int.	x	x			x	
auth.	x	x	x	x	x	x

Table 5.2 lists the data entities used by the designed system, including their security properties.

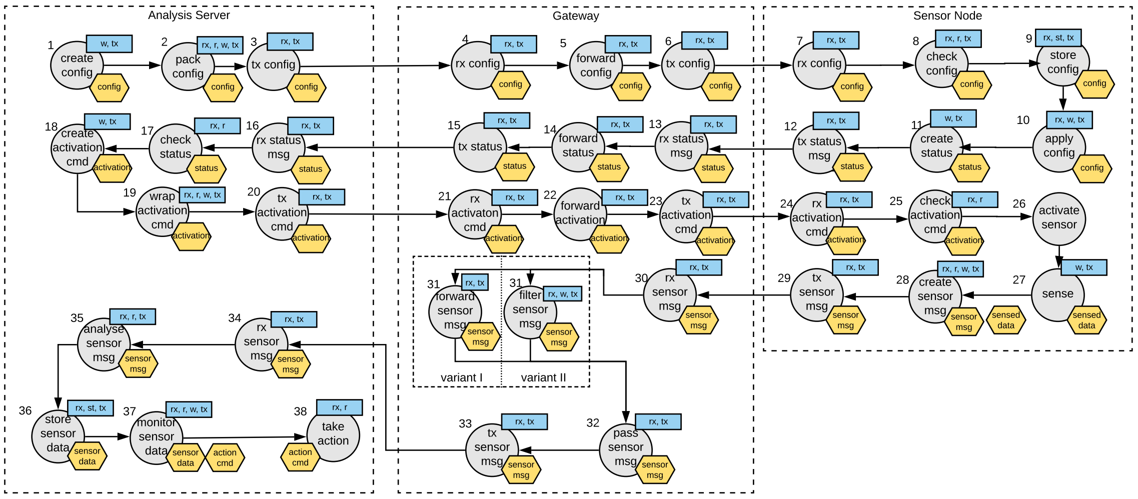


Figure 5.1: Secure sensor system task graph.

The operations comprise: transmit (`tx`), receive (`rx`), write (`w`), read (`r`), store (`st`). The security operations are deduced from the security properties and the operations performed on the data entities according to the rules described in Table 5.5. Figure 5.1 shows how the functionality of the overall system is deliberately split between the sensor node, the gateway, and the analysis server. Thus, the tasks are restricted to the devices. The SaDSE framework still must optimize the task mapping on the different hardware components within the distinct devices. These restrictions are described by the possible mappings of the tasks to the hardware components. This limitation is represented in the mappings described in table 5.4.

Architectural Perspective

The architectural perspective describes the hardware components potentially used by the system design. Table 5.3 lists the components and their different options regarding the security mechanisms. The performance and the power consumption was estimated for each security mechanism based on [148–150]. Both the gateway and the sensor device can consist of a microcontroller MCU and can use a secure element (SE) for performing security-relevant functions. The MCU and the SE are connected with an I2C bus. The sensor device is additionally equipped with a BLE radio. The gateway is equipped with a BLE and a WiFi radio. These radios are connected to the MCU of the sensor device and the gateway using an SPI bus. The sensor device additionally hosts a sensor module for accumulating data. This module is connected to the MCU of the sensor node via an SPI bus. The server is equipped with a WiFi radio. It is realized on a server platform with or without the support Hardware Secure Module (HSM), or directly on a HSM. The server platform can be connected to the HSM using an Ethernet connection. Both the BLE and the WiFi radio do not support any means of security mechanisms. Hence, they are not listed in Table 5.3.

The SE supports tamper-safe-storage (tss), task encapsulation (te) supporting a hardware firewall, symmetric (sym) cryptography, and asymmetric (asym) cryptography provided by a cryptographic co-processor. For the SE, the cryptographic algorithms are side-channel secured. The MCU supports in the security-enhanced variant te with a hardware firewall, hardware-based symmetric, and asymmetric cryptography. In its limited security variant, the MCU only provides software-based symmetric cryptography, which is not side-channel proof, and task encapsulation which is only software-based. The HSM provides tamper-safe-storage, firewall secured task encapsulation, and hardware-based secure symmetric and asymmetric cryptography. Without the HSM realization, the server only offers limited security. This variant only supports symmetric cryptography realized in software and without side-channel security, and task encapsulation only implemented in software without hardware firewall support.

Table 5.3: Hardware components selectable for the secure sensor use case. Each component comprises the implementation vulnerability risk (IVR), and the performance / power consumption of the security mechanisms.

HWC	Security Description	IVR	Security Performance	Security Power Consumption

SE	EAL 5+	0.1	tss(40Mb/s), te(10us)
			sym(500Mb/s), asym(120ms)
				tss(30mW), te(5mW)
				sym(60mW), asym(300mW)

	EAL 6+	0.01	tss(20Mb/s), te(10us)
			sym(400Mb/s), asym(150ms)
				tss(30mW), te(5mW)
				sym(70mW), asym(400mW)

MCU	security enhanced	0.25	te(10us), sym(600Mb/s)
			asym(150ms)
				te(5mW), sym(50mW)
				asym(250mW)

	limited security	0.6	sym(1.2Gb/s), te(5us)	te(5mW), sym(35mW)

Server	HSM	0.1	tss(20Mb/s), te(10us)
			sym(800Mb/s), asym(150ms)
				tss(50mW), te(20mW)
				sym(120mW), asym(600mW)

	limited security	0.6	sym(1.2Gb/s), te(5us)	te(5mW), sym(80mW)

Table 5.4 describes the possible mappings of the tasks depicted in Figure 5.1 to the hardware components described in Table 5.3. The tasks are abbreviated using their respective numbers. The WCET of a task executed on a distinct hardware component is added in brackets next to the task number. These WCETs are given in ms.

Table 5.4: Possible task to hardware component mappings for the secure sensor usecase, with additional WCETs given in ms.

HWC	Platform	Security Description	Task WCETs (in ms)
SE	Sensor Node	EAL 5+ EAL 6+	t8(50), t9(40), t25(50), t28(60) t8(50), t9(40), t25(50), t28(60)
SE	Gateway	EAL 5+ EAL 6+	t31-v1(30), t31-v2(70) t31-v1(30), t31-v2(70)
MCU	Sensor Node	security enhanced limited security	t8(20), t10(30), t11(40), t26(10), t28(30) t8(20), t10(30), t11(40), t26(10), t28(30)
MCU	Gateway	security enhanced limited security	t5(20), t14(20), t22(20), t31-v1(20), t31-v2(50), t32(20) t5(20), t14(20), t22(20), t31-v1(20), t31-v2(50), t32(20)

Server	Analysis server	HSM
		limited security
			t1(40), t2(20), t17(30), t18(50), t19(40)
			t35(60), t36(50), t37(60), t38(40)
			t1(20), t2(10), t17(10), t18(30), t19(20)
			t35(40), t36(20), t37(30), t38(30)

BLE Radio	Sensor Node
	Gateway
		no security
		no security
			t7(10), t12(10), t24(10), t29(10)
			t6(10), t13(10), t23(10), t30(10)

WiFi Radio	Gateway
	Analysis server
		no security
		no security
			t4(10), t15(10), t21(10), t33(10)
			t3(10), t16(10), t20(10), t34(10)

Sensor Module	Sensor Node	no security	t27(100)

5.1.2.2 Potential Attacker

To enforce the security constraints upon the system under design, we modeled, together with our project partners, potential security attack scenarios. To make reasonable assumptions about the security attacks aimed at the sensor system, an analysis of the potential attacker was performed. The assumed attacker is capable of sniffing the communication both between sensor nodes and gateway (BLE), and gateway and analysis server (WiFi). Furthermore, the attacker can physically access the sensor nodes and the gateway and tamper with its memory. The attacker can also inject malicious software code to the sensor nodes, the gateway, and the analysis server to retrieve data via externally accessible interfaces. With physical access to the gateway and the sensor nodes, the attacker can perform side-channel analysis to get information about the secret keys used during the cryptographic processes.

Based on this description of the potential attacker, we identified the following attack scenarios on the secure sensor system. The attack scenarios were modeled as BAGs, as depicted in Figure 5.2. The attacker aims at manipulating the sensor's configuration, trying to misconfigure the sensor and, hence, lead the overall system to unintended behavior. The attacker also aims at disclosing and manipulating the status of the sensor device. These attacks can also lead the sensor to malfunction. The attacker might also aim at faking the activation message and, hence, activate the sensor before being correctly configured. The attacker also aims to disclose and manipulate the sensor data, faking the sensor data message received by the analysis server. This attack can lead the analysis server to trigger wrong actions. Furthermore, the attacker aims at disclosing and faking the action command received by the sensor gateway. The disclosure of the action command can give the attacker unintended information about safety-critical measurement thresholds. Its faking can lead the system into unintended behavior.

The disclosure of the sensor data and its manipulation causing the analysis server to trigger unintended commands was rated as a severe incident. A misconfiguration or a faked status message was rated as a minor incident. This rating is represented in the thresholds assigned to the distinct attack goals in the BAG (Figure 5.2).

Figure 5.2 represents the overall BAG in a more concise form. Each attack in which more than one task is represented describes multiple attacks aiming at distinct tasks. Hence, also the CPTs of these concise representations are split into their respective sub-parts in the real BAG.

5.1.2.3 Security Mappings

The secure sensor system's potential designs can use various security functions and mechanisms to secure the used assets. These functions and mechanisms usable by the system are described by the security mapping provided to the SaDSE framework.

Rule Set

The relation between the operations (OP) and security operations ($SecOp$) performed by the tasks on the data entities, the chosen security functions (SF) and their realization by the designed system (SM), and what security properties (SP) are victim to what attacks (AT) are defined using specific rules. These rules are described by a rule set listed in Table 5.5.

The system performs the OP of writing (w), reading (r), storing (st), transmitting (tx), and re-

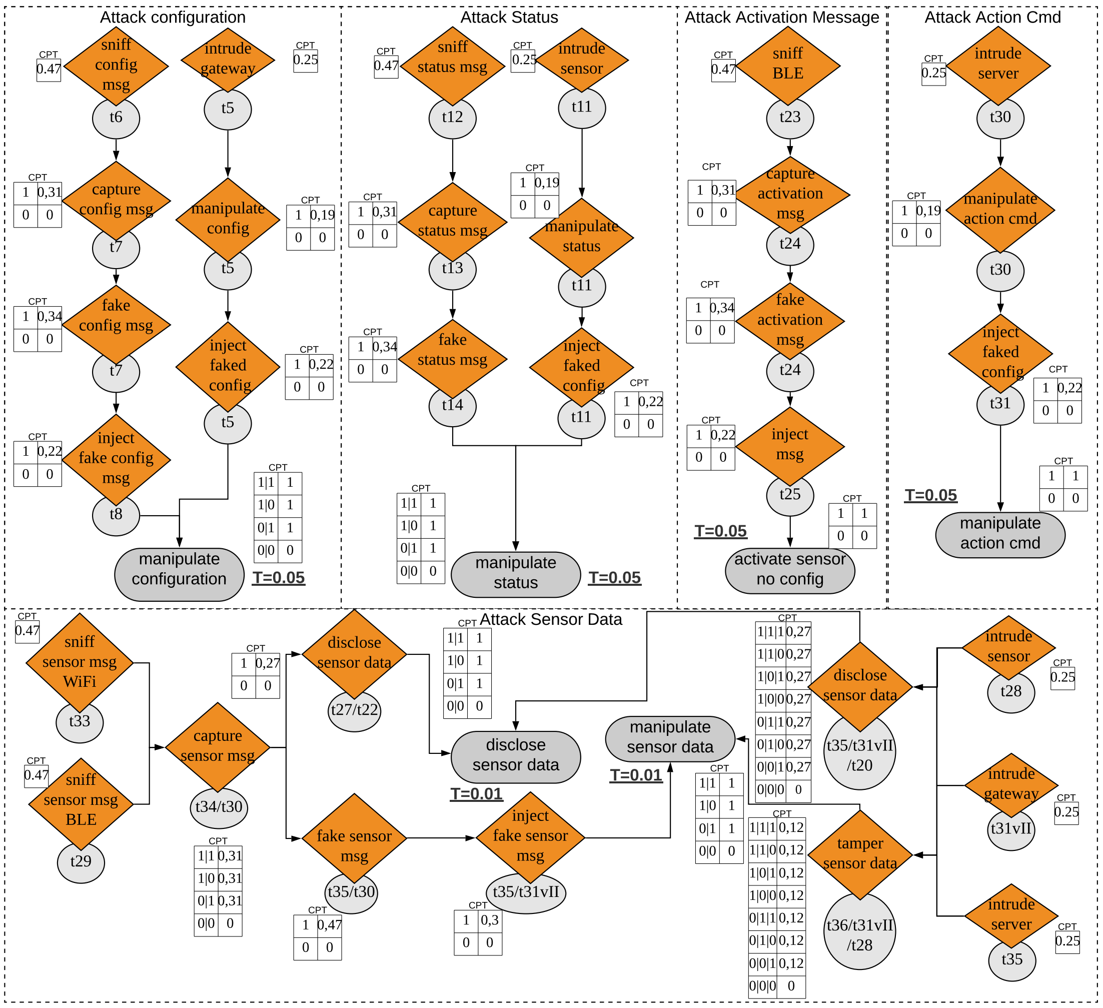


Figure 5.2: Attack graph describing the attack scenarios aimed at the secure sensor system.

ceiving (rx). The data entities need their confidentiality (sp_c), integrity (sp_i), or authenticity (sp_a) protected. Based on the OP and the SP, the system performs distinct $SecOp$ to protect the data entities' SP. The $SecOp$ comprise encryption so_e, authentication so_a, and tamper-proof storing so_t. The SP are aimed at by certain AT. The AT comprise information disclosure (at_i), spoofing (at_s), and tampering (at_t). The $SecOP$ use distinct SF, comprising cryptographic algorithms for encryption (sf_{crypt_e}) and authentication (sf_{crypt_a}), task encapsulation (sf_{te}), and tamper-safe storage (sf_{tss}). The SF are implemented by distinct SMC. These SM comprise symmetric encryption ($sm_{enc_{sym}}$) and authentication algorithms ($sm_{auth_{sym}}$), and asymmetric ones ($sm_{enc_{asym}}$ and $sm_{auth_{asym}}$). If a symmetric or asymmetric encryption or authentication is used depends on the chosen secret keys.

Security Functionality and Secret Keys

To secure the sensor system's assets, the framework was provided with a set of security functions to protect them. Certain security functions are only usable when provided with the correct secret keys. These security functions are cryptographic algorithms used for encryption (sf_{crypt_e}) and authentication (sf_{crypt_a}). These algorithms either use symmetric or asymmetric secret keys. Thereby, the algorithms can use the following secret keys. The symmetric master key (mk) and the asymmetric certificate (cert) are used to derive a symmetric session key (sk). This session key is used to secure the sensor nodes' communication to the gateway and the gateway to the analysis server. The session key is periodically updated using the master key/certificate. Furthermore, the gateway, sensor nodes, and the analysis gateway use a symmetric binding key (bk) used for the encryption and authentication of the internal communication.

Table 5.5: Security rules used in the secure sensor use case study

$SOR \mapsto SecOp$	$AT \mapsto SP$	$\kappa^{SecOp} \mapsto \kappa^{SF}$
$f((r \vee w) \wedge sp_c) = so_e$	$f(at_i) = sp_c$	$f(so_e \wedge ext) = sf_{crypt_e}$
$f((r \vee w) \wedge sp_a) = so_a$	$f(at_s) = sp_a$	$f(so_a \wedge ext) = sf_{crypt_a}$
$f(st \wedge (sp_c \vee sp_a \vee sp_i)) = so_t$	$f(at_t) = sp_i$	$f((so_e \vee so_a) \wedge int) = sf_{te}$
		$f(so_t) = sf_{ss}$

$\kappa^{SF} \mapsto \kappa^{SM}$
$f(sf_{crypt_e}) = (sm_{enc_{asym}} \vee sm_{enc_{sym}})$
$f(sf_{crypt_a}) = (sm_{auth_{asym}} \vee sm_{auth_{sym}})$
$f(sf_{te}) = sm_{te}$
$f(sf_{tss}) = sm_{tsm}$

Based on the lifetime of the used secret keys, the attack success probability was rated accordingly. Hence, the master key and the certificate are more interesting targets for the potential attacker than, e.g., the session key. The master key and certificate were rated with an attack success probability of 0.37, the binding key with 0.17, and the session key with 0.09. These attack success probabilities were calculated based on the CVSS. Based on their lifetime, the master key and the certificate is must be secured using sf_{tss}. The other security functions provided to the system were rated with 0.3 for sf_{te} and 0.2 for sf_{tss}.

5.1.2.4 Solution Space

The functional and architectural description, the attack scenarios, and the description of the security functionality and ruleset were fed to the SaDSE framework. We configured the framework to find the solutions with the optimal performance, the least power consumption, the most secure solution, the fastest yet secure solution, and the most power-efficient yet secure solution. The SaDSE provided us with the desired outputs. To put the overall solution space considered by the SaDSE framework into perspective, Figures 5.3 and 5.4 show the overall solution space as scatter plots for system variant I (forwarding of sensor data) and system variant II (prefiltering of sensor data at the gateway), respectively. Each point in the plots shows one solution calculated by the SaDSE. These solutions are ordered according to their performance and power consumption, normalized to the solution with the fastest performance and lowest power consumption, respectively.

Figures 5.5 and 5.6 show the solutions ordered according to their performance normalized to the fastest solution and the average attack success probability (AP_{avg}) of all attack goals, calculated as $AP_{avg} = \frac{\sum_{i=0}^{G} asp_i}{G}$, where asp_i is the attack success probability of goal i and G denotes the numbers of all attack goals. Considering the number of solutions found to be secure for both

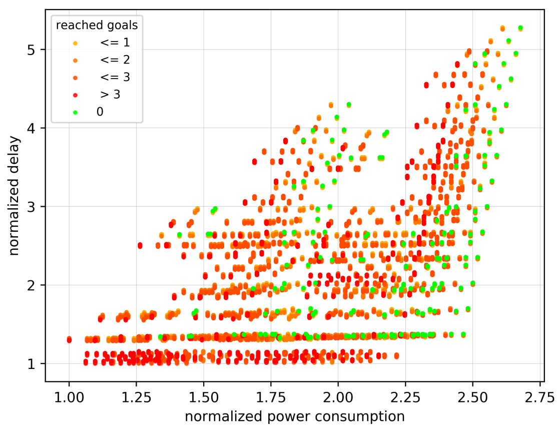


Figure 5.3: Solution space of variant I ordered according to normalized power consumption and performance

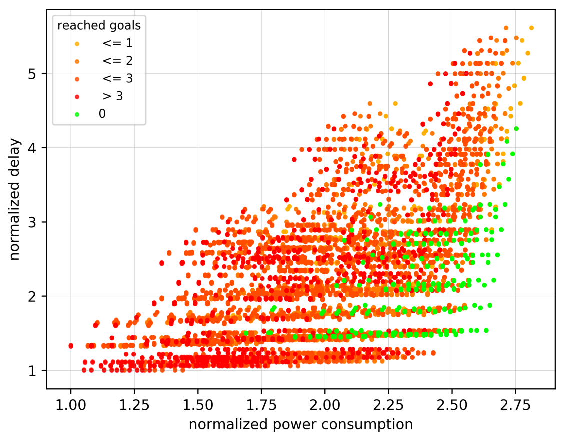


Figure 5.4: Solution space of variant II ordered according to normalized power consumption and performance

variants I and II, one can notice a significant difference. An additional attack vector is opened based on a gateway intrusion performed by the attacker due to the additional filtering process performed on the gateway. Hence, the gateway in variant II must provide a more thorough security hardening than the gateway used in variant I. In both variants, the overall solution space is vastly reduced by integrating the security constraints. For variant I, only $9,400$ solutions (7.5%) out of $125,304$ solutions were rated secure. For variant II, only $4,060$ solutions (3.24%) out of the $125,304$ solutions are considered secure.

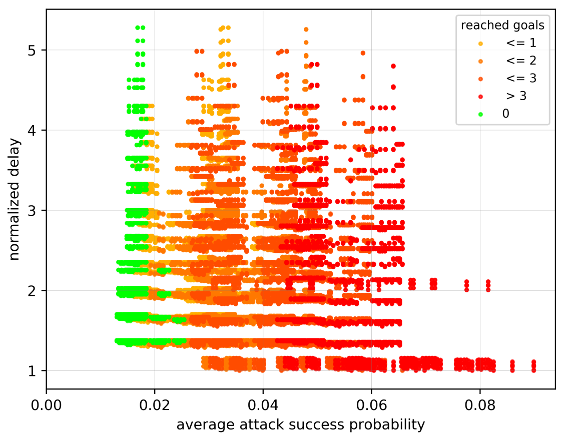


Figure 5.5: Solution space of variant I ordered according to average attack success probability and normalized performance

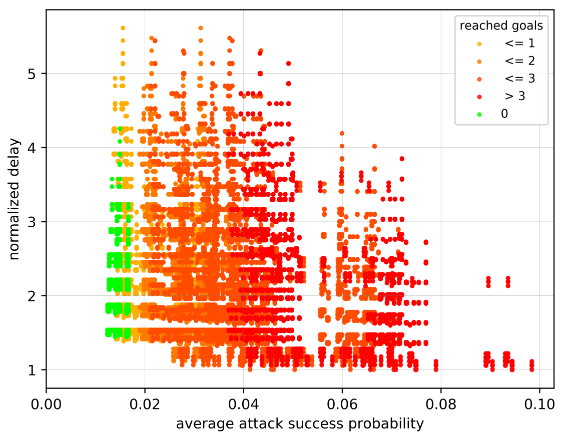


Figure 5.6: Solution space of variant II ordered according to average attack success probability and normalized performance

Considering the overall solution space, we were interested in the most secure, the fastest, the fastest secure, and the most power-efficient secure (MPE) solution. These solutions are described in Table 5.6. The table describes the system partitioning of the named solutions, their overall delay in seconds, their power consumption in mW, and their key placement. One can notice the

Table 5.6: Most secure, the fastest, the fastest secure, and the most power efficient secure (MPE) solutions found based on average attack probability (AP_{avg}), the execution time and the power consumption.

Device	HWC	Most secure	Fastest (var. I)	Fastest secure (var. I)	Fastest secure (var. II)	MPE secure (var. I)
Sensor	MCU	/	lim.-sec.	sec.enh.	sec.enh.	sec.enh.
	SE	EAL 6	EAL 5	EAL 6	EAL 6	EAL 6
Gateway	MCU	sec.enh.	lim.-sec.	sec.enh.	sec.enh.	sec.enh.
	SE	EAL 6	/	/	EAL 6	EAL 6
AS	Server	HSM	lim.-sec.	lim.-sec. + HSM	lim.-sec. + HSM	lim.-sec. + HSM
$AP_{avg} =$		0.012	0.13	0.015	0.016	0.014
delay (s)		2.98	1.95	2.54	2.8	2.93
power cons. ((mW)		940	430	465	490	375
Key Placement						
Sensor	MCU	/	ssk, sbk	ssk, sbk	ssk, sbk	ssk, sbk
	SE	ssk, smk, $cert$	sbk, smk, $cert$	sbk, smk	sbk, smk	sbk, smk
Gateway	MCU	ssk, sbk	/	/	ssk, sbk	/
	SE	sbk, smk, $cert$	/	/	sbk, smk	/
AS	Server	ssk, smk, $cert$	ssk, smk, $cert$	ssk, smk, $cert$	ssk, smk, $cert$	ssk, smk, $cert$

difference in the SE selection for the use case variants I and II. This difference fits the variation seen in the overall solution space found for the two variants. The SaDSE framework is able to find the real optimal solutions given the optimization criteria due to the used constraint programming approach.

Based on the found solutions, a prototype implementation was performed by the project partners. This prototype was based on the fastest secure solution found by the SaDSE framework for variant I.

System Realization

The solutions found by the SaDSE were used by our project partners to design and implement a prototype sensor system, consisting of one secure sensor node, a secure gateway, and a prototype server application. The secure sensor nodes were realized using a Raspberry Pi 3 Model B, integrating a BLE 4.2 chip. The node was equipped with an HTS221 temperature sensor and extended with an SE050 SE utilizing the chip's I2C interface. The gateway was realized using a Raspberry Pi 4 Model B controller, which offers both BLE and WiFi interfaces. The analysis server was realized as a cloud solution instantiated on Amazon EC2. The Amazon Web Services (AWS) CloudHSM is a cloud-based HSM. The AWS CloudHSM supports the creation and manages confidential information, such as cryptographic keys, and other information. The secure data exchange between the sensor node and the analysis server via the gateway is secured using an AES-based encryption scheme. The authentication between gateway and analysis server is based on an RSA-based authentication process. The prototype's performance was evaluated by calculating the average over 50 measurement cycles. When featuring the security mechanisms, the overall execution time of the prototype is $2,338\text{ms}$. Without any security mechanisms, the prototype's execution time is $1,852\text{ms}$ [151]. This execution time was measured from the accumulation of the sensor data until its reception on the analysis server. When comparing the execution times of the prototype with the fastest secure solution ($2,540\text{ms}$) and fastest solution ($1,950\text{ms}$) found for variant I, one can notice that the found solutions match the real execution times quite well.

5.1.3 Secure Indoor Localization System

The secure indoor localization system use case was performed in the UB-Smart project together with NXP Semiconductors Austria. We used the secure indoor localization system use case to verify the functionality of the SaDSE framework by revisiting the system's design. In this use case, we used the risk tree approach combined with the BAG-based attack description. Furthermore, we used the secure indoor localization system use case to show the SaDSE framework's capability of generating a system simulation based on the solutions found by the analytical DSE part. This integration between the simulation-based and analytical DSE is explained in Section 5.3. The use case described here largely builds on our work described in [2].

5.1.3.1 System Description

Again, the system is described using the SaDSE framework's functional and architectural description. Assessing the system's security, we described the potential attacker targeting the indoor localization system. The potential attack scenarios are depicted within a BAG, extended with risk information. Thus we compared the BAG-based and the risk-based approach. The security mapping describes the rules mapping the security properties to security mechanisms and other mappings.

Functional Perspective

The secure localization use case describes an indoor localization system capable of performing secure ranging consisting of node and anchor devices. The anchors are stationary placed in a room at known and fixed locations. The node devices' location is unknown to the localization system. The anchors and nodes exchange ranging packets via an Ultra Wideband (UWB) channel. The UWB technology allows the devices to measure the timestamps of incoming and outgoing packets precisely. Knowing the timestamps of both anchor and node allows the calculation of the time of flight. Thus, the distance between the two devices can be estimated. With the distance estimations between a node and three or more anchors, the node's location can be calculated using trilateration, with respect to the anchors' positions. In this use case, the anchor is connected to a keyless entry system. The node can be used as a key, and the anchor controls the lock.

The system representation fed to the SaDSE framework is a simplified representation of the overall system. The functionality of the use case is described by the task graph depicted in Figure 5.7. The system's functionality is split into three phases.

Authentication Phase: In the authentication phase, the node and the anchor device authenticate each other by exchanging challenges. These challenges are signed by the devices using their symmetric master keys. This authentication is only performed once and valid as long as the system localizes the node.

Session & Configuration Phase: In the session and configuration phase, the node and the anchor derive their session keys from the master key. This session key is used within a time-limited session to encrypt and authenticate the configuration sent from the anchor to the node, authenticate the status message passed from the node to the anchor, and derive the ranging key.

Ranging Phase: The ranging phase consists of the ranging packet exchange. These packets contain the node's timestamp information sent from the node to the anchor. The ranging packets are secured using the ranging key. A new ranging key is derived from the session key for each ranging cycle. A ranging cycle starts with the poll ranging packet sent by the node and ends with the final ranging packet. The anchor uses its timestamps and the node's timestamps to calculate the distance between the two devices. If the distance is below a configured threshold, the anchor opens the lock and sends the information about the unlock process to the node.

The secure localization system designed in this system can be used in diverse scenarios. In the scenario at hand, the system is designed to control access to a valuable facility. However, the designed anchor and node devices can also be used in general for a secure localization system. Using the solutions found for the device designs, we evaluated such a localization system in Section 5.3.

The data entities and their security properties are listed in Table 5.7. The data entities comprise the challenge exchanged between anchor and node, the session request (requ.) and response (resp.), the configuration (config.) of the node, the status message, the open command (cmd), and the ranging (rng.) packets (poll message, response message, final message.

Table 5.7: Data entities used by the secure localization system and their security properties: confidentiality (conf.), integrity (int.) and authenticity (auth.).

	config	challenge	session requ. & resp.	config.	status	rng packet	open cmd	mk	sk	rk
conf.		x	x		x	x	x	x	x	x
int.							x	x	x	x
auth.	x	x	x	x	x	x	x	x	x	x

Architectural Perspective

Both the anchor and the node device are composed of the same hardware components. They consist of an application processor (AP), an optional secure element (SE), an UWB radio (UR), and a BLE radio. The BLE radio does not offer any security mechanisms.

The hardware components come in different versions providing various security mechanisms. These are listed in Table 5.8. The AP, SE, and UR come with different security variants. The AP features hardware-based (HWC) or software-based cryptography (SWC). The SWC comes either with side-channel protection (scp) or only as a functional (f) implementation. Furthermore, the AP can feature a trusted execution environment (TEE). The SE comes with various EAL levels. The UR variants feature HWC, functional SWC, and either support firewall (FW) based task encapsulation, a TrustZone (TZ), or a microcontroller separation (MS).

The variants of the hardware components support different security mechanisms. These mechanisms feature encryption sm_{enc}, authentication sm_{auth}, tamper-safe storage sm_{tss}, and task encapsulation sm_{te}. The mapping of these mechanisms to the security operations performed by the tasks of the functional description is defined by the rule set given in Table 5.10.

The possible mappings of the tasks to the hardware component variants of both devices are described in Table 5.9. These mappings include their WCETs. The estimated WCET of each task is written in brackets next to the task's identifier. The performance values for the different implementation options of the tasks were taken from an initial system design, and are given in ms.

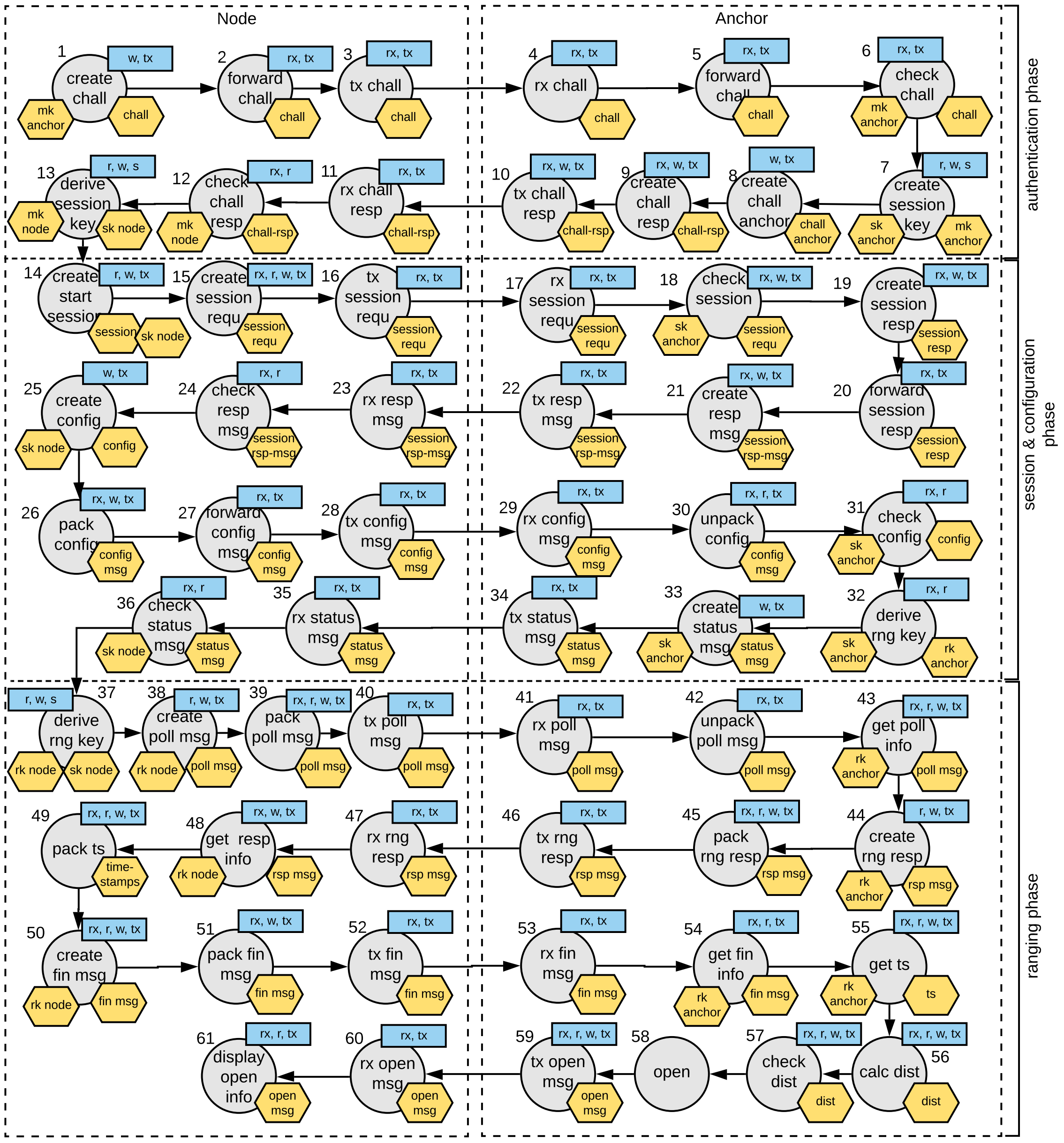

Node
Anchor
authentication phase
session & configuration phase
ranging phase
1 create chall w, tx mk anchor chall
2 forward chall rx, tx chall
3 tx chall rx, tx chall
4 rx chall rx, tx chall
5 forward chall rx, tx chall
6 check chall rx, tx mk anchor chall
7 create session key r, w, s sk anchor mk anchor
8 create chall anchor w, tx chall anchor
9 create chall resp rx, w, tx chall-rsp
10 tx chall resp rx, w, tx chall-rsp
11 rx chall resp rx, tx chall-rsp
12 check chall resp rx, r mk node chall-rsp
13 derive session key r, w, s mk node sk node
14 create start session r, w, tx session sk node
15 create session requ rx, r, w, tx session requ
16 tx session requ rx, tx session requ
17 rx session requ rx, tx session requ
18 check session rx, w, tx sk anchor session requ
19 create session resp rx, w, tx session resp
20 forward session resp rx, tx session resp
21 create resp msg rx, w, tx session rsp-msg
22 tx resp msg rx, tx session rsp-msg
23 rx resp msg rx, tx session rsp-msg
24 check resp msg rx, r session rsp-msg
25 create config w, tx sk node config
26 pack config rx, w, tx config msg
27 forward config msg rx, tx config msg
28 tx config msg rx, tx config msg
29 rx config msg rx, tx config msg
30 unpack config rx, r, tx config msg
31 check config rx, r sk anchor config
32 derive rng key rx, r sk anchor rk anchor
33 create status msg w, tx sk anchor status msg
34 tx status msg rx, tx status msg
35 rx status msg rx, tx status msg
36 check status msg rx, r sk node status msg
37 derive rng key r, w, s rk node sk node
38 create poll msg r, w, tx rk node poll msg
39 pack poll msg rx, r, w, tx poll msg
40 tx poll msg rx, tx poll msg
41 rx poll msg rx, tx poll msg
42 unpack poll msg rx, tx poll msg
43 get poll info rx, r, w, tx rk anchor poll msg
44 create rng resp r, w, tx rk anchor rsp msg
45 pack rng resp rx, r, w, tx rsp msg
46 tx rng resp rx, tx rsp msg
47 rx rng resp rx, tx rsp msg
48 get resp info rx, w, tx rk node rsp msg
49 pack ts rx, r, w, tx time-stamps
50 create fin msg rx, r, w, tx rk node fin msg
51 pack fin msg rx, w, tx fin msg
52 tx fin msg rx, tx fin msg
53 rx fin msg rx, tx fin msg
54 get fin info rx, r, tx rk anchor fin msg
55 get ts rx, r, w, tx rk anchor ts
56 calc dist rx, r, w, tx dist
57 check dist rx, r, w, tx dist
58 open
59 tx open msg rx, r, w, tx open msg
60 rx open msg rx, tx open msg
61 display open info rx, r, tx open msg

Figure 5.7: Secure localization system task graph.

5.1.3.2 Potential Attacker

Indoor localization systems pose a juicy target for security attacks, especially when utilized in a way described by this use case, in which they actively control the access to a restricted area. However, in other use cases, e.g., indoor navigation or indoor equipment tracking, the system poses the threat of security attackers obtaining private localization data of users and equipment, and even manipulating it. When controlling the access to restricted areas, a successful attack has a direct impact on the secured valuable objects.

Table 5.8: Hardware components with security features (Sec. feat.). Implementation vulnerability risk (IVR), performance (Perf.) given in µs, and power consumption (PWC) in mW

HWC	Sec. feat.	IVR	Perf. (µs)				PWC (mW)			
			sm_{enc}	sm_{auth}	sm_{tss}	sm_{te}	sm_{enc}	sm_{auth}	sm_{tss}	sm_{te}
AP	HWC, TEE	0.25	40	30	-	10	20	20	-	10
	SWC scp, TEE	0.4	70	50	-	10	60	40	-	10
	SWC scp	0.5	70	50	-	-	60	40	-	-
	SWC f	0.8	45	30	-	-	40	30	-	-
SE	EAL 6+	0.01	110	80	50	20	70	50	20	15
	EAL 5+	0.1	100	80	30	15	60	50	10	10
	EAL 4+	0.2	100	80	20	10	50	40	10	10
UR	HWC, FW	0.4	50	40	-	15	30	20	-	10
	HWC, TZ	0.5	50	40	-	10	30	20	-	10
	HWC, MS	0.3	60	45	-	20	40	30	-	20
	SWC f, TZ	0.6	80	60	-	10	40	30	-	10
	SWC f	0.8	50	40	-	-	35	20	-	-

Table 5.9: Possible task to hardware component mappings for the secure localization system, with additional WCETs given in µs. The hardware components are characterized with their security descriptions (Sec. Descr.)

HWC	Platform	Sec. Descr.	Task WCETs (in µs)
SE	Node	EAL 6+	t1(20), t12(10), t13(20), t14(10), t25(15), t26(20), t36(15), t37(25), t38(10), t49(10), t50(15)
		EAL 5+	t1(20), t12(10), t13(20), t14(10), t25(15), t26(20), t36(15), t37(25), t38(10), t49(10), t50(15)
		EAL 4+	t1(20), t12(10), t13(20), t14(10), t25(15), t26(20), t36(15), t37(25), t38(10), t49(10), t50(15)
	Anchor	EAL 6+	t6(15)t7(15),t8(10), t19(20), t20(15), t31(15), t32(20), t43(10), t44(15), t55(10), t56(15), t57(10)
		EAL 5+	t6(15)t7(15),t8(10), t19(20), t20(15), t31(15), t32(20), t43(10), t44(15), t55(10), t56(15), t57(10)
		EAL 4+	t6(15)t7(15),t8(10), t19(20), t20(15), t31(15), t32(20), t43(10), t44(15), t55(10), t56(15), t57(10)
UR	Node	HWC, TEE	t37(15), t38(5), t39(5), t40(2), t47(2), t48(5), t49(5), t50(5), t51(5), t52(2)
		SWC scp, TEE	t37(15), t38(5), t39(5), t40(2), t47(2), t48(5), t49(5), t50(5), t51(5), t52(2)
		SWC scp	t37(15), t38(5), t39(5), t40(2), t47(2), t48(5), t49(5), t50(5), t51(5), t52(2)
		SWC	t37(15), t38(5), t39(5), t40(2), t47(2), t48(5), t49(5), t50(5), t51(5), t52(2)
	Anchor	HWC, TEE	t41(2), t42(5), t43(5), t44(5), t45(5), t46(2), t53(2), t54(5), t55(5), t56(5), t57(5)
		SWC scp, TEE	t41(2), t42(5), t43(5), t44(5), t45(5), t46(2), t53(2), t54(5), t55(5), t56(5), t57(5)
		SWC scp	t41(2), t42(5), t43(5), t44(5), t45(5), t46(2), t53(2), t54(5), t55(5), t56(5), t57(5)
		SWC	t41(2), t42(5), t43(5), t44(5), t45(5), t46(2), t53(2), t54(5), t55(5), t56(5), t57(5)
AP	Node	HWC, FW	t2(5), t12(5), t13(15), t14(10), t15(5), t24(10), t25(10), t26(10), t27(5), t35(10), t36(10), t61(5)
		HWC, TZ	t2(5), t12(5), t13(15), t14(10), t15(5), t24(10), t25(10), t26(10), t27(5), t35(10), t36(10), t61(5)
		HWC, MS	t2(5), t12(5), t13(15), t14(10), t15(5), t24(10), t25(10), t26(10), t27(5), t35(10), t36(10), t61(5)
		SWC f, TZ	t2(5), t12(5), t13(15), t14(10), t15(5), t24(10), t25(10), t26(10), t27(5), t35(10), t36(10), t61(5)
		SWC f	t2(5), t12(5), t13(15), t14(10), t15(5), t24(10), t25(10), t26(10), t27(5), t35(10), t36(10), t61(5)
	Anchor	HWC, FW	t5(5),t6(10),t7(10),t8(5),t9(5), t18(5), t19(10), t20(5), t21(5), t30(5), t31(10), t32(10), t33(10), t58(5)
		HWC, TZ	t5(5),t6(10),t7(10),t8(5),t9(5), t18(5), t19(10), t20(5), t21(5), t30(5), t31(10), t32(10), t33(10), t58(5)
		HWC, MS	t5(5),t6(10),t7(10),t8(5),t9(5), t18(5), t19(10), t20(5), t21(5), t30(5), t31(10), t32(10), t33(10), t58(5)
		SWC f, TZ	t5(5),t6(10),t7(10),t8(5),t9(5), t18(5), t19(10), t20(5), t21(5), t30(5), t31(10), t32(10), t33(10), t58(5)
		SWC f	t5(5),t6(10),t7(10),t8(5),t9(5), t18(5), t19(10), t20(5), t21(5), t30(5), t31(10), t32(10), t33(10), t58(5)
BLE	Node	no security	t3(5),t11(5), t16(5), t23(5), t28(5), t35(5), t60(5)
	Anchor	no security	t4(5),t10(5), t17(5), t22(5), t29(5), t34(5), t59(5)

To assess the attack vectors on the indoor localization system, we first assessed the potential attackers. We assumed that the attacker aiming at the system can sniff both the BLE and UWB traffic. Furthermore, the attacker can fake messages and inject them into both wireless channels. The attacker can intrude on the software stack of both the anchor and the node device and disclose valuable security assets. The attacker can physically tamper and disclose the memory of the devices. The attacker can also mount side-channel attacks aiming at the cryptographic keys used to encrypt and authenticate the security assets.

Based on the attacker's capability, we derived the attack scenarios depicted in Figure 5.8. The attack graph describes the attack scenarios directed towards the three phases of the secure indoor localization system. The attacks aimed at the authentication phase comprise the disclosure and the manipulation of the challenge exchange between anchor and node device. The attacks are either launched by intruding the devices' software stacks or sniffing the BLE communication and injecting a faked challenge. Furthermore, after accessing a device's software, the attacker can attempt to disclose or even tamper the master key used at the challenge creation. We rated the tampering/disclosure of the master key as the attack with the highest impact that must be mitigated as much as possible.

The attacks on the session establishment and configuration phase consist of manipulating the session information and the configuration data. In this phase, also the session key constitutes a high-value target for the attacker. The session information and the configuration data can be attacked either by sniffing, faking, and injecting manipulated messages into the BLE communication, or by the intrusion into the devices' software. For disclosing and manipulating the session key, the attacker needs logical access to the devices. Because of the timely restriction, the session key's disclosure and manipulation were rated less sever than the attack on the master key.

The attacks on the ranging phase comprise the faking of timestamps, the manipulation of the distance calculation, and the ranging keys' disclosure and manipulation. The faking of the timestamps leads to manipulating the ranging process and to inaccurate distance measurements. The attacks on the timestamps can be mounted by interfering the UWB communication or by logically intruding the devices. The manipulation of the distance calculation and the ranging key's manipulation and disclosure require a prior intrusion of the devices' logic. As the ranging key is renewed after each ranging cycle, its disclosure is less severe than the disclosure of the session or master key.

We assessed the security of the secure indoor localization use case using both the BAG-based representation of the attack scenarios and the risk tree-based description. In contrast to the BAG-based security calculation, which uses single point estimations, the risk tree-based approach relies on probability distributions. These probability distributions are provided to the risk tree as vulnerabilities. Furthermore, the risk tree supports the integration of monetary impact values (*imp*), and frequency (*frequ*) values. The frequency values describe how often an attack is attempted per year. The vulnerabilities of the risk tree map the conditional probabilities of the CPTs of the BAG. The risk tree does not allow the dependency of an attack from multiple preceding attacks. Therefore, the SaDSE framework supports an automatic unrolling of the BAG into the corresponding risk trees. This transformation from the BAG to the risk tree must only be performed once, before starting the risk-based security constraint calculation. The frequencies and impacts of successfully reaching the distinct attack goals are also described in Figure 5.8. The attack goals leading to the system's malfunction and the tampering of the ranging key are rated with an impact of $10,000$\$. The tampering of the master key was rated with an impact of $1,000,000$\$, the tampering of the session key with $200,000$\$. These impact values account for the potential damage the successful attack can have on the overall system. The impact values were chosen arbitrarily and in such a way that a heavy effect on the risk-based calculation can be seen.

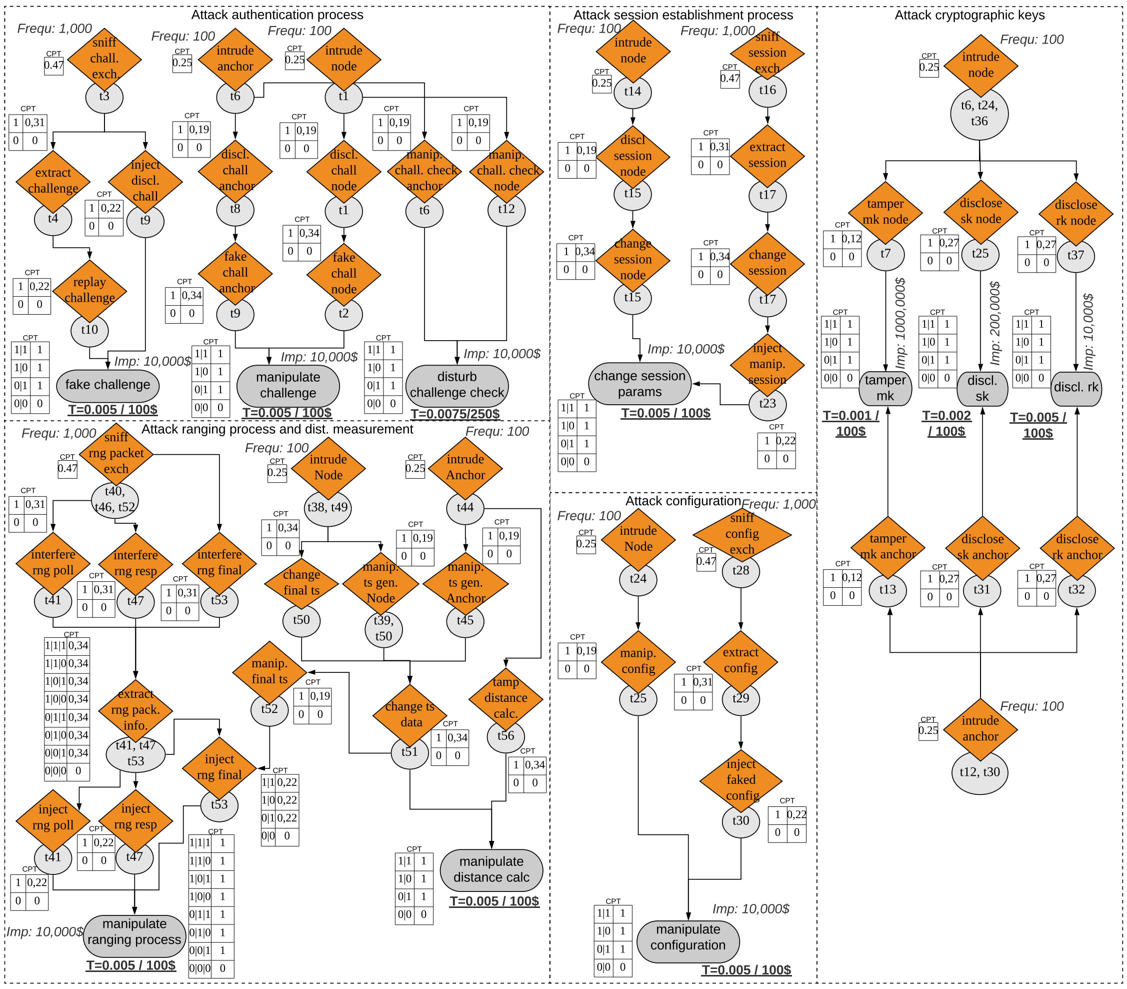


Figure 5.8: Attack graph describing the attack scenarios aimed at the secure indoor localization system.

5.1.3.3 Security Mappings

Similar to the secure sensor use case (Section 5.1.2), the system design knows the operations of writing w, reading r, and storing st, the attack type of information disclosure at_i, spoofing at_s, and tampering at_t. It knows the security properties of confidentiality sp_c, authenticity sp_a, and integrity sp_i, and the security operations / mechanisms of encryption so_{enc} / sm_{enc}, authentication so_{auth} / sm_{auth}, secure storage so_{ss} / m_{tss}, and the mechanism of task encapsulation sm_{te}. Contrasting the secure sensor use case, in this use case we did not define security functions the SaDSE framework has to choose from for protecting the system's assets. Without the declaration of security functions, the security operations $SecOp$ are directly linked to the security mechanisms SM provided by the system's hardware components. The mappings of operations OP and security properties SP to $SecOp$, the mappings of attack types AT to SP, and the mappings of $SecOp$ to SM are given in Table 5.10. The attributes int and ext denote, if the corresponding $SecOp$ is performed internally (within the same hardware component) or externally.

Table 5.10: Security rules used in the secure localization use case study

$SOR \mapsto SecOp$	$AT \mapsto SP$	$\kappa^{SecOp} \mapsto \kappa^{SM}$
$f((r \vee w) \wedge sp_c) = so_e$	$f(at_i) = sp_c$	$f(so_{enc} \wedge ext) = sm_{enc}$
$f((r \vee w) \wedge sp_a) = so_a$	$f(at_s) \mapsto sp_a$	$f(so_{auth} \wedge ext) = sm_{auth}$
$f(st \wedge (sp_c \vee sp_a \vee sp_i)) = so_t$	$f(at_t) = sp_i$	$f((so_{enc} \vee so_{auth}) \wedge int) = sm_{te}$
		$f(so_{ss}) = sm_{tss}$

Table 5.11: Most secure, fastest, and fastest secure solution found based on AP_{avg}, and MRV_{avg}, with the delay normalized to system with lowest delay. Solutions are given for node (N) and anchor (A).

HWC	Most secure	Fastest	Fastest sec. (BAG)	Fastest sec. (RISKEE)
AP (N & A)	HWC, TEE	SWC f.	HWC, TEE	HWC, TEE
SE (N & A)	EAL 6+	EAL 4+	EAL 4+	EAL 6+
UR (N & A)	HWC, MS	SWC f.	HWC, TZ	HWC, FW
AP_{avg} / MRV_{avg}	0.0007 / 117.45$	0.005 / 3905.37$	0.0017	126.74$
norm delay	~1.73	1.0	~1.074	~1.16

5.1.3.4 Solution Space

The functional description, the description of the potential architectural system realizations, the potential attack scenarios, and the mapping rules were fed to the SaDSE framework. We configured the framework to find secure solutions, using the BAG-based and risk tree-based approach. We configured the SaDSE framework to find the most secure, the fastest, and the fastest secure solution. The framework calculated the solutions' security as average attack success probability $AP_{avg} = \frac{\sum_{i=0}^{G} asp_i}{G}$ and as average mean risk value $MRV_{avg} = \frac{\sum_{i=0}^{G} mrv_i}{G}$, with G being the number of all attack goals, and asp_i and mrv_i being the attack success probability and the mean risk value of goal i, respectively. The system partitioning for the desired solutions are described in Table 5.11. The selected hardware components are described for both the anchor (A) and the node (N) device. For each solution, the normalized (norm) delay is noted.

In addition to the described solutions, we configured the SaDSE framework to present the whole explored solution space. Figures 5.9 - 5.11 depict the overall solution space [2]. In these figures, each solution is depicted as a distinct point in the scatter plot. Figure 5.9 shows all BAG-based solutions, Figure 5.10 shows all risk-based solutions, and Figure 5.11 shows all solutions for which the security was calculated using a combined BAG- and risk-based approach. One can notice, that the BAG-based calculation approach considered more solutions to be secure (green points), than the risk-based calculation approach.

This difference is caused by the additional impact and frequency attributes provided by the risk trees. Especially, the additional information on the monetary loss induced by a successful security attack tightens the constraints the solutions must fulfill. Thus the number of secure solutions found for the risk-based security constraints is further reduced. This is also reflected in the fastest secure solution found for the BAG-based and risk-based calculation (Table 5.11). The solution found for the BAG-based solution only uses SEs with an EAL of 4+ and uses TrustZone-based separation for the task encapsulation on the UR. The solution found by the risk-based calculation enforces SEs with EAL 6+ to be used for both anchor and node devices. Furthermore, the solution utilizes a hardware-based task encapsulation, based on a firewall integrated into the UR. The

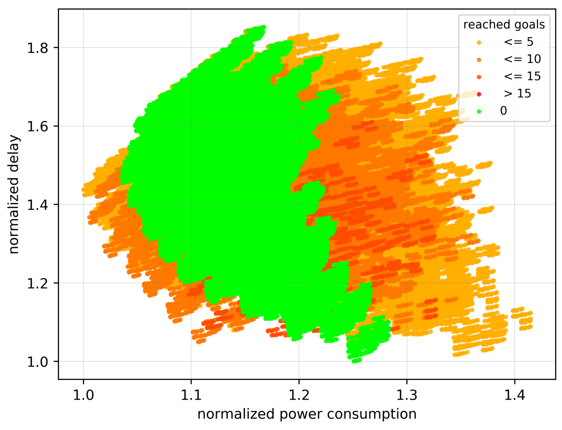


Figure 5.9: Solution space with security calculated using BAGs.

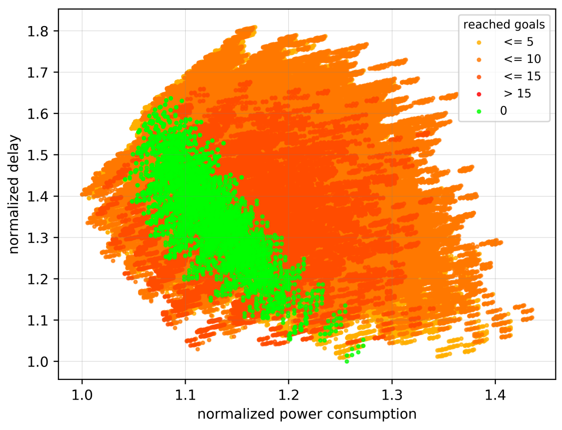


Figure 5.10: Solution space with security calculated using risk trees.

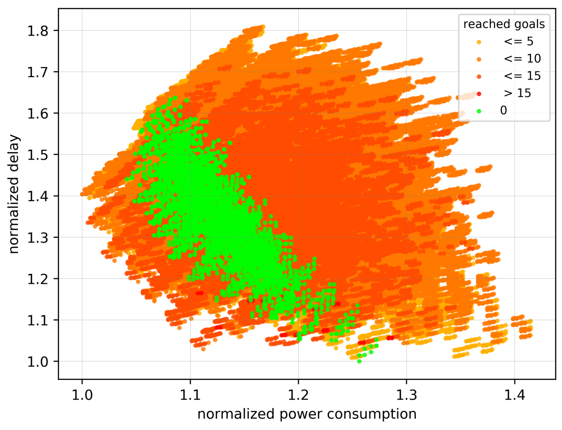


Figure 5.11: Solution space with combined BAG and risk tree based security calculation.

usage of the higher-rated SE is due to the risk induced by disclosing the master key and session key. The hardware-based firewall mitigates the risks of manipulating the distance calculation and timestamps stored by the devices. The performance of the fastest secure solution was compared to the prototype implementation, measured from creating the challenge (task 1) to creating the polling message (task 38). The found solution performance was 12.8% slower than the prototype's performance. One must notice that these values heavily depend on the estimation accuracy of the WCETs provided as input to the SaDSE framework.

The combined BAG- and risk-based calculation approach performs in a first step the calculation of the asp of the attack goal using the BAG-based approach. Only if the asp of the goal does not exceed its threshold, the SaDSE framework also calculates the mrv for the goal. Thus, the more time-consuming calculation of the goal's risk is only performed if the faster BAG-based computation yields the attack goal not reached. With the combined approach, the SaDSE framework finds the same secure solutions as when using the risk-based approach, but with better performance. This can be seen when comparing Figures 5.10 and 5.11. However, comparing the insecure solutions, one can notice that the combined approach might find solutions that have even more reached attack goals than their risk-based pendants. This difference can occur for potential system designs where the asp of a goal does not exceed its threshold, but the risk-based calculation renders the goal reached. The performance comparison between the single approaches is shown in Section 5.2.

The analysis of the secure indoor localization use case produced us with some key insights towards the usability of the analytical part of the SaDSE framework. When comparing the solutions found for the BAG-based and risk-based approach to the system design produced by our project partner, we noticed that the BAG-based solution for the fastest secure system design would lack two essential security features. First, a high-secure SE, and second a hardware-based security context protection. Contrasting the solution found by BAG-based approach, the risk-based approach was capable of providing the fastest secure solution, which was in line with the system design chosen by the project partner's security experts. Considering this outcome, we can see that the SaDSE framework is capable of providing meaningful and reasonable solutions for secure system designs. However, the accuracy of the found solutions depends on the provided inputs' accuracy. Therefore, the solutions produced by the SaDSE framework depend on the experience and the knowledge of the domain experts.

5.2 Performance Measurements

The DSE performed by the SaDSE framework comes with considerable computational overhead. Depending on the modes used for the SaDSE framework, this performance overhead varies. This section describes the performance measurements for the secure sensor system and secure indoor localization use cases and shows the additional overhead caused by the security constraint calculation. For both use cases, the SaDSE framework was run on a system providing 16GB of RAM and an Intel® Core™ i7-4600U CPU running at $2.10\,\text{GHz}$.

Secure sensor system use case

The performance of the SaDSE framework for calculating the solution space for the secure sensor use case was measured with and without the task mapping limitation described in Section 4.3.1.3. Without the task mapping limitation, the overall DSE took $\sim$2h56min for the use case variant II. This calculation time was reduced by using the task mapping limitation to $\sim$1h32min. Hence, the task mapping limitation reduced the computation time by $\sim 47\%$. For use case variant I, the computation time without the task mapping limitation took $\sim$2h43min. With the task mapping limitation, the DSE took $\sim$1h40min. The task mapping limitation reduced the computation time by $\sim 35\%$. Without considering the security constraints, the calculation of the overall solution space took $\sim$56min. Thus, the integration of the security constraints leads to a performance overhead of $\sim 68\%$ when not using the task mapping limitation.

Secure Indoor Localization System Use Case

For the secure indoor localization system use case, we measured the framework's performance using the three different security constraint calculation approaches: the BAG-based, the risk-based, and the combined approach. The performance of these approaches was measured using different variants of the use case. Each variant describes a subset of the overall design space. Variant 1 describes the whole use case. Variant 2 describes only the authentication phase, and the session establishment and configuration phase. Variant 3 describes only the authentication phase. Variant 3 only produces 580 possible solutions, variant 2 produces $19,728$ solutions, and variant 1 comes with the full $947,072$ possible solutions. Figures 5.12 and 5.13 show the performance measurements given in seconds for the three variants.

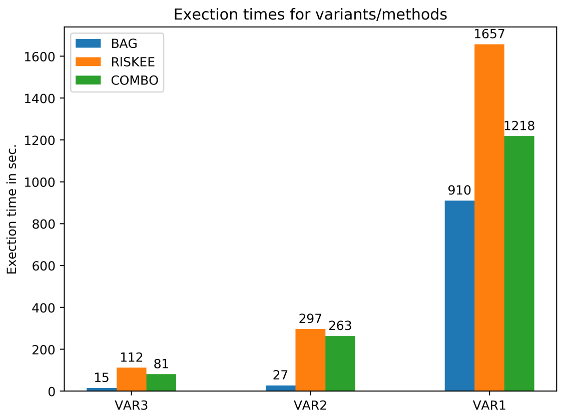


Figure 5.12: SaDSE framework's performance without task mapping limitation [2].

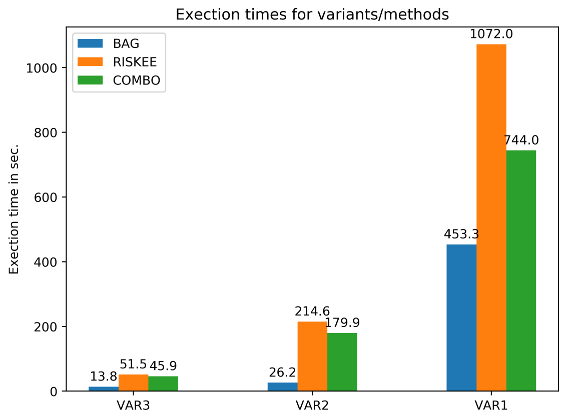


Figure 5.13: SaDSE framework's performance with task mapping limitation.

Figure 5.12 shows the computation time without task mapping limitation, Figure 5.13 with the task mapping limitation. One can see that the increasing number of possible solutions vastly increases the computational overhead of the SaDSE framework. The task mapping limitation's impact on the performance increases with the design space's size. Considering variant 1, the risk tree-based calculation is 1.82 times more time consuming than the BAG-based calculation without task mapping limitation. With task mapping limitation, the increase of the computation time between BAG and risk tree-based is 2.36 times. Considering the risk tree-based computations with and without task mapping limitation for the individual variants, the task mapping limitation reduces the computation time by 54.57%.

Assessing the additional computation time caused by the introduction of the security constraints, we compared both BAG and risk tree-based variant 1 without task mapping limitation to the DSE without using security constraints. Without the introduction of the security constraints, the DSE of variant 1 takes $10\text{min}34\text{s}$. Hence, the BAG-based calculation comes with an overhead of 43.53%, the risk tree-based calculation with an overhead of 163.35%.

5.3 Simulation-based DSE

The simulation-based DSE part of the SaDSE framework was evaluated using the secure indoor localization use case. We selected certain solutions from this use case and let the framework generate a simulation model from those solutions according to preset configurations. The following sections describe the simulation setups chosen for further assessing the usability of the solutions found by the analytical part of the SaDSE framework, and how the generated system simulation can be further used to improve distinct parts of the system under design.

5.3.1 System Simulation Evaluation

The solutions found for the secure indoor localization use case were used to perform a simulation of multiple anchors and nodes within an indoor environment. The indoor environment is represented as a room that bounds the single nodes' movement and the placement of the anchors, and as a distinct channel model in the physical channel simulation provided by Veit et al. [145]. To show the impact of the task mapping and system partitioning performed by the analytical part of the SaDSE on the overall system behavior, we selected a range of solutions to be used in the simulation-based DSE part. The fastest, the most secure, the least secure, and the three fastest and secure solutions were used in the simulation scenario. Each solution was used for generating the device models in each distinct scenario.

The SaDSE framework generated the components for the system simulation using the devices described in the secure indoor localization use case. The anchor and node devices are connected via the BLE and the UWB communication channels. The environment of the simulation was configured to represent a room of $20\mathrm{m}$ length, $15\mathrm{m}$ width and $4\mathrm{m}$ height. We placed 5 anchor devices in this simulated room in all four corners ([$5\mathrm{m}/3\mathrm{m}$], [$5\mathrm{m}/12\mathrm{m}$], [$16\mathrm{m}/12\mathrm{m}$], [$16\mathrm{m}/3\mathrm{m}$]), and in the center ([$10\mathrm{m}/7\mathrm{m}$]) of the room. We simulated several different scenarios. Each scenario differs in the number of nodes to be localized in the room. The simulation was run with 5, 10, 20, 50, 70, and 100 nodes. The nodes are placed at random locations within the room and move into different directions. Whenever they hit an obstacle, a wall, or another node, they randomly change direction and proceed. The anchor and node devices are generated using the system partitioning and task mapping according to the selected solution.

The devices perform the functionality given by the task graph. Each scenario is run with the selected solutions given by the analytical DSE part. Figures 5.14 and 5.15 show the UWB channel throughput and localization update rate for each solution in every scenario. Each scenario was run with the node and anchor devices generated from the different solutions found by the analytical DSE. In the figures, *Sol1* describes the fastest solution, *Sol2* describes the most secure solution, *Sol3* describes the least secure solution, *Sol4* the fastest secure solution, *Sol5* the second fastest secure solution, and *Sol6* the third fastest secure solution. The solutions were run in every scenario. These scenarios are abbreviated with *S1 - S6*. *S1* simulates 5 node devices, *S2* 10 nodes, *S3* 20 nodes, *S4* 50 nodes, *S5* 70 nodes, and *S6* 100 nodes.

The channel throughput was calculated as the ratio between transmitted and received packets. The localization update rate describes how often the anchors can estimate the distance to the node, therefore, tracking how often the end task of the task graph is reached. To trigger multiple ranging rounds, the task *create poll message* (t38) was configured as recurring task with an offset between $5\mathrm{ms}$ and $10\mathrm{ms}$, and 500 iterations. Each scenario was simulated for one minute.

The physical channel simulation calculated bit errors for packet transmissions with various distances (from $1\mathrm{m}$ up to $60\mathrm{m}$), with LOS and NLOS, and packet overlap of 100%, 80%, and 50%. We do not assume that any error-correcting codes are transmitted with the exchanged packets. Hence, a single bit error renders a packet unusable [152].

One can notice that for both the channel throughput and the distance estimations, the fastest solution outperforms all other solutions. Especially the most secure solution comes with a noticeably lower channel throughput and less distance estimation per minute than the other solutions. When considering the three fastest solutions, one can notice that their channel throughput and distance

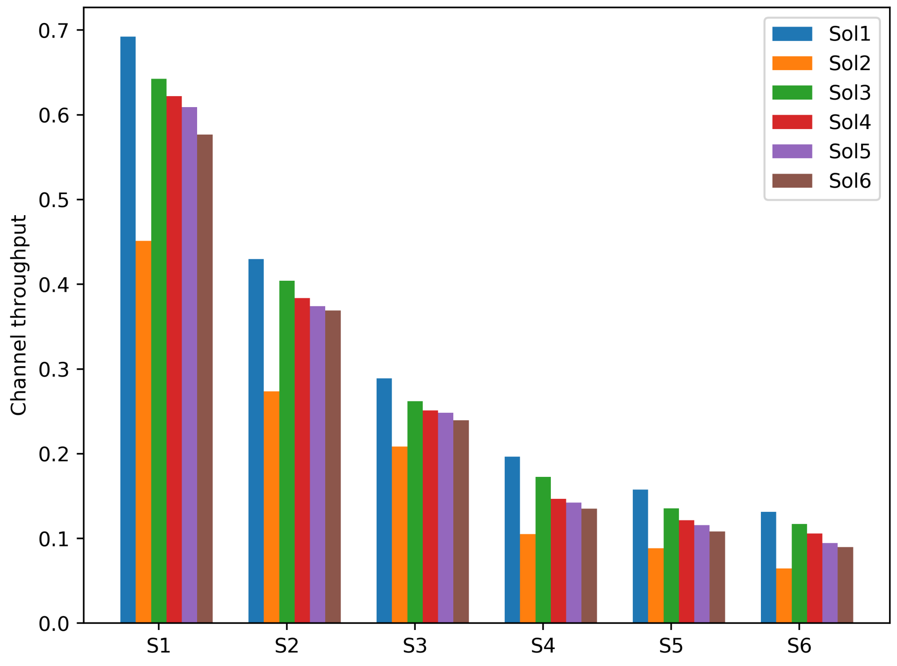


Figure 5.14: Channel throughput of the single scenarios running the devices generated from the respective solutions found during the analytical DSE.

estimations per minute are close to the fastest solution's performance. This behavior is similar in all scenarios, except the channel throughput in *S1*. Furthermore, the impact of the different solutions on the channel throughput and the distance estimations per minute becomes less noticeable the more the system is tracking nodes.

With an increasing number of tracked nodes, the influence of the chosen networking and ranging protocol increases, as was shown in [144]. The ranging protocol described by the secure indoor localization use case and the randomly chosen starting offset leads to collisions on the wireless communication channel that increase with the number of communication participants. For scenarios with many communication participants, a supervised networking protocol, such as the slotted ALOHA system, would yield a much better channel throughput. The influence of the chosen solutions on the number of distance estimations is best seen in *S4*, in which the selection of a solution other than the fastest or fastest secure solution would yield less than one distance estimation per second. Based on the use case, this could influence the use of experience with the localization service.

With the simulations performed using the secure indoor localization use case, we show how the simulation system generated by the SaDSE framework can be used to assess the system's performance under design in various scenarios. Especially at the early stage of the system design, this approach can give a valuable insight into the overall system and help designers to understand the impact of the chosen system components on its performance.

The accuracy of the simulation is hard to assess, as we were not able to realize the usecase using a real setup. However, the used physical channel model [145] was qualitatively compared to real measurements presented in [153]. Hence, we argue that the physical channel model is accurate.

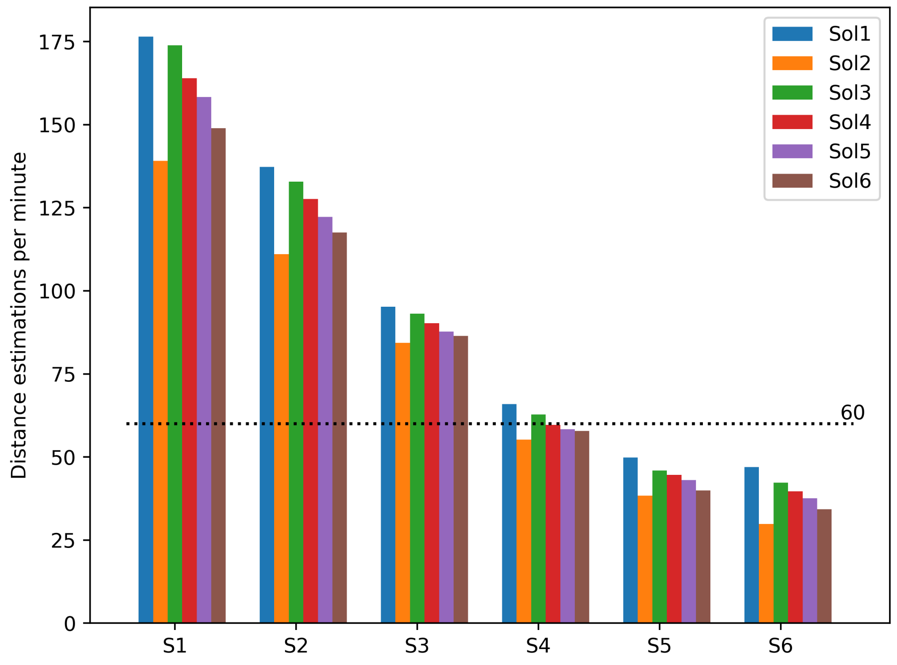


Figure 5.15: Number of distance estimations performed by the anchors in the single scenarios. For each scenario, the devices were generated from the respective solutions found during the analytical DSE. The threshold of 1 distance estimation per second is marked.

The device simulations are generated from the solution found in the analytical DSE part of the SaDSE framework. Hence, they come with the same performance accuracy (12.8% slower) as the solution they were generated from.

5.3.2 Simulation-based Improvements

In addition to the environment simulation, the system simulation generated from the solutions found by the analytical DSE can be used to extend and detail specific parts of the system design, such as used network protocols, communication bus systems, algorithmic behavior, or additional hardware components. Based on the system generated by the SaDSE framework, we performed such an extension on the protocol level. We extended the protocol used for the communication between the SE and the microcontroller (MCU) connected to it. SEs are usually connected to other system components using an I2C or SPI communication bus where the SE usually acts as a slave.

Considering the communication logic implemented between the SE and the connected MCU, the used protocol stack comprises a transport layer protocol and an application layer protocol. On the transport layer, the *T=1* protocol is widely used [154]. On the application layer, the Application Protocol Data Unit (APDU-protocol) is a standardized protocol used for communicating with smart cards. [155]. The APDU-protocol is based on a master-slave concept, where the master sends commands to the slave, and the slave responds to them. In this scheme, the SE only acts as a slave to the connected MCU, processing the received commands and responding to them.

However, in certain situations, it can be beneficial that the SE is also capable of sending messages to the MCU out of its initiative. Examples for such situations could be passing information to the MCU about possible secret key compromise, or passing a new session key after session expiring.

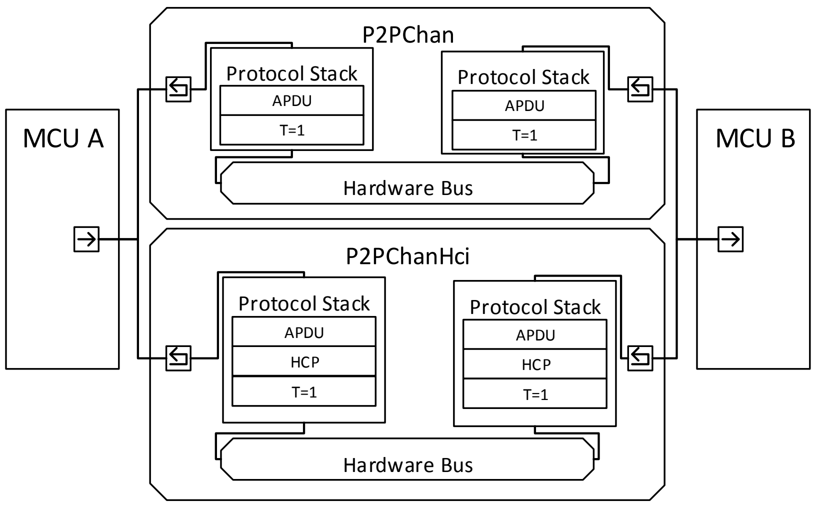


Figure 5.16: Diagram showing the basic blocks for implementing the peer-to-peer APDU approach implemented in the system simulation model [3].

To accomplish this change in the communication direction, a change in the protocol stack is necessary. We used the system model generated by the SaDSE framework to design a peer-to-peer protocol stack and simulated it. We designed two approaches. The first approach only uses the APDU protocol and wraps commands from the SE into responses, and responses from the MCU into commands. Thereby, we introduce a new command demanding a message from the SE and a command wrapping the MCU's response. The second approach integrates the Host-Controller Protocol (HCP) as an additional layer into the protocol stack [156]. The HCP allows the passing of APDU commands and responses as event messages. These HCP events are used in the second approach to wrap the APDU messages. Additionally, a logic for changing the communication direction was proposed. Both approaches were integrated into the system simulation, as depicted in Figure 5.16. `MCU A` and `MCU B` represent the hardware components generated from the SE and the UWB radio (UR). The `P2PChan` and the `P2PChanHci` implement the physical communication channel (`PCC`) connecting SE and UR in the system simulation. For both approaches, the hardware must support an additional signal triggering the UR to activate the bus communication.

The performance overhead of the proposed approaches was evaluated. We used the standard communication with the SE acting as a slave and the UR acting as master as a performance reference. Using the HCP based peer-to-peer solution, the communication overhead was increased 5% in both directions. This increase is caused by the additional header information induced by the HCP. The APDU wrapping approach only induces an additional overhead of 98% when sending commands from the SE to the UR. The original command-response direction from the UR to the SE stays untouched and, hence, comes without any performance overhead. The purpose of this design study was to show the feasibility of using the generated system simulation to further detail distinct aspects of the designed system, including potential optimizations.

5.3.3 Usability of the Framework

The usability of the framework was assessed by measuring the time it took to describe the two usecases presented here. Based on time-writing and time stamps from the versioning control system, it took us $9\text{h}30\text{min}$ to model the secure indoor localization use case and 12h to model the secure sensor system. This does not take into account the modeling of the usecases themselves, but only measures the time it took us to provide them as inputs to the SaDSE framework. This long modeling times were mostly caused by the input format (XML) supported by the SaDSE framework. A graphical interface would improve the modeling. Furthermore, we argue that the used specification language is very flexible, as it allows the designer to formalize additional security operations, mechanisms, and many other attributes due to the extendable rule set. The SaDSE framework also allows the splitting of use cases into sub-modules to better support modeling large and complex systems. With this option, each sub-module is optimized on its own, limiting the possible optimization of the overall usecase. The optimized sub-modules can, however, be reintegrated into solutions covering the overall usecase with a merge-program provided by the SaDSE framework.

6 Summary and Conclusion

In this chapter, we conclude the thesis by summarizing its contributions, outlining the limitations, and discussing future work.

6.1 Conclusion

In this thesis, we presented the SaDSE framework. This framework considers security aspects during an automatic DSE for the early phase of designing secure embedded systems. Contrasting other works focusing on security constraints during early system design, the framework offers the designers the direct integration of attack scenarios from the attacker's perspective modeled as either BAGs or risk trees. Additionally, the framework integrates the security aspect of the secret keys used for the cryptographic operation. Thus, it takes into account the risks induced by a potential leakage of the keys used to secure the system's assets. These aspects are integrated into the analytical DSE part of the SaDSE framework. Based on the inputs, the framework derives a set of security constraints to which the feasible solutions must adhere. This derivation is defined by a rule set extendable by the designers. With the SaDSE framework's additional capability of generating a system simulation from the solutions found by the analytical DSE, it also allows the simulation of the found solutions within dynamically changing environments. The generated simulation model allows the designers to ensure the usability of the proposed solution further. The generated simulations can further be used to optimize distinct parts of the system under design. With the integration of the BAG and risk tree-based attack scenario descriptions into the analytical DSE in combination with the consideration of the secret keys' security, the rule-based security constraint calculation, and the combination with the system simulation, the SaDSE framework offers a holistic approach for the automated DSE of secure systems.

We showed the applicability of the SaDSE framework by integrating it into the design process of a secure sensor system and revisiting the early system design phase of a secure indoor localization system. These use cases, provided by our project partners, gave us valuable insight into our system's usability. In both use cases, the SaDSE framework was capable of yielding design proposals similar to the actually implemented designs. As security experts from our project partners supported these designs, we argue that the solutions found by the SaDSE framework can cope with attack scenarios anticipated for the use cases. Hence, when provided the realistic estimation for the input parameters, the SaDSE framework can support the design of secure embedded systems early on. Based on the secure indoor localization system, we further showed the influence of the selected solutions found in the analytical DSE on the overall system behavior. The generated system simulation was further used to design a protocol stack allowing bi-directional communication between a SE and a MCU.

6.2 Limitations

The here presented approach comes with some limitations. The results with the SaDSE framework are heavily dependent on the quality of its inputs. The better the estimation of the power consumption, worst-case execution times, attack potential, and security of the components, the more accurate the found solutions. Especially the estimation of the security aspects of systems poses a major challenge, which has been investigated by various researchers. Considering these aspects, the impact of successful attacks is the easiest aspect to quantify. The probability of a security attack and the capability of a system to mitigate them are measures whose quality heavily depends on the security experts. In general, this estimation is a challenge in security risk assessment [106].

In addition to the constraints covered by the SaDSE framework, the design and especially the successful implementation of a new system or product depends on further aspects not yet covered. One such aspect is the anticipated time to market and the implementation risk induced by the pressure applied by the various internal and external stakeholders. The integration of these aspects could massively benefit the early design of systems and products.

6.3 Future Work

The SaDSE framework offers multiple directions in which the security-aware DSE of embedded systems can be further extended. The usage of the BAGs and the risk trees for modeling the attack scenarios offer the designer a means to describe potential attacks from the attacker's perspective. Currently, these scenarios are described by security experts. However, BAGs can also be automatically built from historical data as proposed by Feng et al. [157]. Therefore, the SaDSE's attack perspective could be extended with the option to have the attack scenarios generated from documented attacks.

As explained in Section 6.2, the SaDSE framework should be extended by further constraints, such as implementation risk and time to market. These constraints would add a distinct benefit to the early system design phase. In addition to these constraints, also the system simulation can be extended in future work. As the SaDSE framework generates a system simulation in SystemC, external network simulators can be integrated, such as Omnet++ as described by Zhao et al. [158].

7 Publications

In the course of the author's Ph. D. studies the following publications were published in various conferences. These publications constitute the basis of this thesis.

A) Lukas Gressl, Ulrich Neffe, Christian Steger, "Design and Implementation of an HCI based Peer to Peer APDU Protocol", in *Proceedings of the 21st Euromicro Conference on Digital System Design (DSD 2018).*

B) Lukas Gressl, Christian Steger, Ulrich Neffe, "Message Encapsulation Pattern", in *Proceedings of the 23rd European Conference on Pattern Languages of Programs*.

C) Lukas Gressl, Christian Steger, Ulrich Neffe, "A Security Aware Design Space Exploration Framework", in *Proceedings of the Fourteenth International Conference on Systems (ICONS 2019).*

D) Lukas Gressl, Christian Steger, Ulrich Neffe, "Consideration of Security Attacks in the Design Space Exploration of Embedded Systems", in *Proceedings of the 22nd Euromicro Conference on Digital System Design (DSD 2019).*

E) Lukas Gressl, Christian Steger, Ulrich Neffe, "Security Driven Design Space Exploration for Embedded Systems", in *Proceedings of the Forum for Specification and Design Languages (FDL 2019).*

F) Lukas Gressl, Alexander Rech, Christian Steger, Andreas Sinnhofer, Ralph Weissnegger, "Security Based Design Space Exploration for CPS", in *Proceedings of the 35th Annual ACM Symposium on Applied Computing*.

G) Lukas Gressl, Michael Krisper, Christian Steger, Ulrich Neffe, "Towards an Automated Exploration of Secure IoT/CPS Design-Variants", in *Proceedings of the International Conference on Computer Safety, Reliability, and Security*.

H) Lukas Gressl, Alexander Rech, Christian Steger, Andreas Sinnhofer, Ralph Weissnegger, "A Design Exploration Framework for Secure IoT-Systems", in *Proceedings of the International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA 2020)*

I) Lukas Gressl, Michael Krisper, Christian Steger, Ulrich Neffe, "Towards an Automated Exploration of Secure IoT/CPS Design-Variants", in *Proceedings of the International Conference on Cyber Security and Protection of Digital Services (Cyber Security 2020).*

J) Lukas Gressl, Christian Steger, Ulrich Neffe, "Design Space Exploration for Secure IoT Devices and Cyber-Physical Systems", accepted for publication in the *ACM Transactions on Embedded Computing Systems*.

K) Alexander Rech, Lukas Gressl, Fikret Basic, Christian Seifert, Christian Steger, ”Multi-Layered IoT System Design Towards Secure End-to-End Communication”, presented at the *46th Annual Conference of the IEEE Industrial Electronics Society (IES)*.

2018 21st Euromicro Conference on Digital System Design

Design and Implementation of an HCI based Peer to Peer APDU Protocol

Lukas Gressl
Institute of Technical Informatics
Graz University of Technology
Graz, Austria
gressl@tugraz.at

Ulrich Neffe
NXP Semiconductors Austria GmbH
Gratkorn, Austria
ulrich.neffe@nxp.com

Christian Steger
Institute of Technical Informatics
Graz University of Technology
Graz, Austria
steger@tugraz.at

***Abstract*—An ever increasing number of System on Chips need secure storage of key material or confidential data, therefore relying on the usage of Secure Elements (SEs). In traditional systems, the SE is a passive device, communicating with the other system's components via a master-slave topology. As applications running on SEs tend to become more involved in the interaction with other components by actively sending out data, the present communication setup poses a hindrance. In this paper we propose a method, which allows the bidirectional exchange of command-response messages of the Application Protocol Data Unit (APDU) protocol, by encapsulating the APDU messages in packets defined by the Host Controller Interface (HCI). Thus, the master-slave based APDU protocol can be used in a peer to peer communication, without changing the APDU protocol, and minimally extending the HCI. In this paper, the HCI extensions of the new approach are explained. The HCI based approach is compared to a method, which only uses the APDU protocol, by evaluating a simulation based implementation, and comparing the expected performance of both approaches.**

***Keywords*-Embedded Secure Elements, System on Chip, Network Protocol, P2P Communication, Embedded Applications**

I. Introduction

In today's System on Chips (SoCs), an increasing number of applications rely on secure storage of important data, such as key material, or confidential data. Secure Elements (SE), which offer such a secure storage, play an increasingly important role in SoCs. The SE usually is a passive entity within the system, which responds to commands sent by other components. Therefore, the communication system, on which the SE communicates with other communication partners, is based on a master-slave topology, restricting the SE's actions within an SoC. With systems becoming more complex, the SEs future role might not only be to handle secure information and make it available to other components. They could also handle secure applications, which actively send commands to other system components. To enable this future behavior, a future communication protocol must grant all communication participants the same privileges. One such architecture, in which the usage of a bidirectional communication protocol would be beneficial, is depicted in Figure 1. In this architecture, the SE is connected to both the radio and the micro-controller (MCU), acting as a slave. In a setup, in which the radio sends confidential data to the SE and the MCU interacts with the user, the SE might host an application, which actively transmits data via the radio, based on the information it prior got from it. Furthermore, the SE's application might send some information to the MCU, which is used to instruct the user. In such a system, a SE, which only acts as a slave, could not fulfill the required functionality.

The paper is split into the following sections: Section II outlines other fields of application for peer to peer (P2P) protocols; Section III describes the approach for creating a bidirectional communication protocol; Section IV outlines the communication model, and gives an evaluation of the model-based timing results. Section V gives our conclusion and an outline on future extensions.

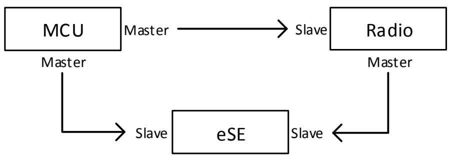


Figure 1: Abstract SoC consisting of an SE (acting as slave), connected to an MCU and a Radio.

II. Related Work

P2P communication systems are well known for forming the basis of distributed computing systems, allowing the sharing of computer resources without the need of an intermediary centralized server. They can be used for resource sharing [1], instant messaging [2], or secure storage systems [3]. The usage of P2P systems is not restricted to web-based applications, such as described in the preceding paragraph, but is increasingly used in the context of Internet of Things [4], even in the field of vehicle to vehicle communication [5]. Further applications relying on P2P communication can be found in the field of machine to machine communication [6]. Looking at a smaller granularity, P2P communication is also used for the interaction of devices as small as smart cards. The physical link is usually established by Near Field Communication (NFC) [7], via which data is transfered using the NFC Logical Link Control Protocol (LLCP) [8]. Although the LLCP knows a P2P protocol, it was not considered in the proposed design, as the goal was to achieve an P2P communication with as little changes to the already existing protocol stack, as possible.

III. Designing an HCI based P2P APDU protocol

The goal of the P2P APDU protocol is to break up the master-slave topology between two components of an SoC without changing the application layer protocol. By sticking to the already well established APDU protocol, a high acceptance of the proposed design should be achieved. As already existing applications, running on the individual components, would not need to change the message parsing at the Application Layer. Therefore, the Application Layer Protocol should stay unchanged and the bidirectional communication should come with minimal overhead. Considering the defined requirements, two different approaches were evaluated, which allow a bidirectional APDU based command-response exchange. The first approach is straight forward. The SE is the slave, the MCU the master in the communication. Case A shows the usual command-response exchange between the two communication parties. Case B shows a method to allow the MCU to send a command to the SE. The MCU starts with sending an empty command-APDU (C-APDU). Depending on the underlying hardware bus, the SE might need to trigger the communication with an external interrupt signal. In response to this C-APDU, the MCU responds by sending an response-APDU (R-APDU), which holds the actual command in its data field. The SE receives the R-APDU, interprets the enclosed command, and sends back its response, encoded into the data field of a C-APDU. The MCU extracts the response and sends back an empty R-APDU, thus completing the command response exchange. With this method, a P2P APDU exchange can be accomplished. The drawback of this method is, that each command-response exchange from the SE to the MCU needs four messages, causing additional delay.

To overcome this empty message exchange, the second approach uses a bidirectional message type of an underlying protocol to encapsulate the command and the response APDU. For this purpose, the Host Controller Interface (HCI) standard is used [9]. The HCI standard defines the Host Controller Protocol (HCP), which is situated in the network layer of the protocol stack. Its main purposes are to allow the setup of a network, consisting of multiple logical hosts. The HCP knows command and response pairs, but also uses an asynchronous message type, called event. This event type is used to allow the C-APDU - R-APDU exchange in a bidirectional way. Figure 2 shows the difference in the message exchange between the two design approaches.

It can be noticed that the second design approach encapsulates the command and the response APDU using the event message type of the HCP. Two different types of events, called `EVT_C-APDU` and `EVT_R-APDU`, are used. The event types are explained in more detail in Section III-A. The drawback of the second design approach is that by

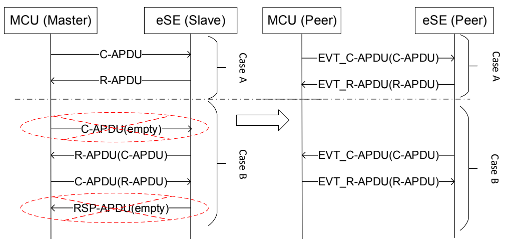


Figure 2: Bidirectional APDU protocol approaches.

encapsulating each message, additional header information is transmitted, increasing the communication delay. For Case A, this means that the overall exchange time is expected to increase. Using the HCP, an additional communication setup is necessary, prior to exchanging the HCP packets, which adds to the overall communication time. For Case B, it can be seen that by using the asynchronous bidirectional event message type of the HCI standard for encapsulation, the empty messages of the first design approach can be omitted. However, the transmission and processing overhead must be taken into consideration for Case B as well.

A. Extension to the HCI standard and Adaptation of the Protocol Stack

The latest HCI/HCP Protocol standard contains a clause defining a mechanism enabling the exchange of APDU messages [9]. Two different types of hosts, the server APDU host (slave) and the client APDU host (master) are defined. The client host provides a so-called *APDU application gate*, which accepts the reception of `EVT_R-APDU` messages. The server host provides the *APDU gate*, which accepts the reception of `EVT_C-APDU` messages. As this setup does not allow the exchange of `EVT_C-APDU` and `EVT_R-APDU` messages in both directions, the interface is extended with an additional *P2P APDU gate*, which allows the reception of both event message type. Thus, a bidirectional APDU command-response exchange can be achieved. Additionally to the `EVT_C-APDU` and the `EVT_R-APDU` message, the *P2P APDU gate* must also support the other events of the *APDU application* and the *APDU gate*, as defined in the HCI/HCP Protocol standard, respectively [9]. Offering both peers the same set of events supports the APDU message exchange from both directions. The type encodings of the `EVT_C-APDU` and the `EVT_R-APDU` message must be adapted for the *P2P APDU gate*, as both the *APDU application gate* and the *APDU gate* use the same instruction code for both events.

As the *P2P APDU gate* is able to send and receive both `EVT_C-APDU` and `EVT_R-APDU` messages, its state diagram needs additional states to differentiate between sending and receiving mode. As shown in Figure 3, the *P2P APDU gate* goes into the `INIT` state after network setup. From this state, it can either go into the `IDLE` or the `WAIT CMD`, depending on the reception of `EVT_ATR`, which is used by the sender to indicate its availability for processing APDU commands. If the gate transits into the WAIT CMD state, it receives commands and transmits responses, until reception of a EVT_END_TRANS, which sends it back to the IDLE state. If the gate gets into the IDLE state, it may either send or request a command. If it requests a command, it sends a EVT_REQ_CMD message and goes into the WAIT CMD state. From this state it might only return to IDLE upon reception of a EVT_REQ_CMD message. If the gate transmits a command while residing in the IDLE state, it transits into WAIT RSP. From this state, the gate may send commands and receive responses, until a change in the direction of the command-response exchange is requested. To initiate this change, it sends a EVT_END_TRANS message and goes into the state WAIT CMD. Using the state diagram shown in Figure 3, changing the direction of communication between two *P2P APDU gates* is only possible, if the *P2P APDU gate* in charge indicates an end to the APDU exchange. Upon this event, both *P2P APDU gates* can agree upon the direction of the following command-response exchange. Next to the *P2P APDU gate*, the host must also provide other

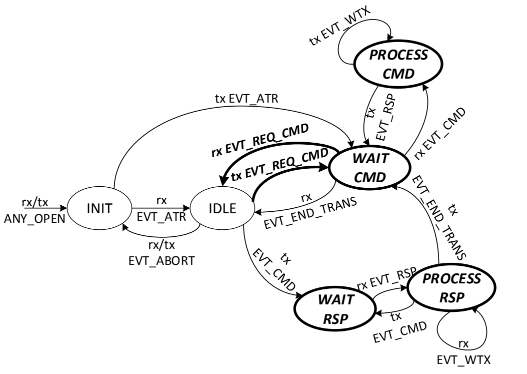


Figure 3: State diagram of *P2P APDU gate* with transmitted (tx) and received (rx) event messages.

necessary gates, as described in [9]. For using the HCP to exchange APDU messages, an additional driver is needed in the communicating devices' firmware, thus extending the protocol stack by one layer. The driver's purpose is to check the HCP packet's source and destination. It hands over the packet's data (received APDU message), together with the information about the data's size, and the information, whether an command or response APDU was received, to the Application Layer. The HCP expects that the packets are transmitted via a Data Link Layer (DLL). The DLL must support an error free transmission, respect the order of sent and received packets, provide its own data flow control, and specify a maximum packet size. Furthermore, it must report the size of each received packet, transmitted via the physical layer. With this data, the HCP driver is provided all the necessary information to support the upper layer, which further processes the received APDU message. The HCP driver also encapsulates a given message into an HCP Packet before handing it over to the underlying DLL. For this purpose, the driver needs, next to the APDU message to encapsulate, information regarding the type of the APDU and the address of the pipe. Concluding, the proposed design offers bidirectional communication with minimal overhead, without changing the Application Layer Protocol, and the additional possibility to address non physically connected components. Therefore, the design needs an additional driver to construct and parse the HCP packets and messages, and the network must be setup.

IV. Model Implementation and Evaluation

To evaluate the effect of the P2P APDU based approach on the communication performance, both approaches introduced in Section III were implemented in a system model and compared to each other. The system implements a communication model, describing the delay caused by the exchange of an arbitrary number of APDU messages between two MCUs. The communication is implemented in a SystemC Transaction Layer Model (TLM) [10]. In this model, two MCUs are communicating with each other via a hierarchical channel. This channel is declared as an interface, offering the connected modules to send and receive both C-APDUs and R-APDUs. Two channels implement this interface, P2PChan and P2PChanHci. P2PChan implements the first approach presented in Section III, P2PChanHci the second one, utilizing the HCI. For communication, the MCU module contains a port connecting it to the interface. Figure 4 shows the TLM's structure, as well as the layouts of P2PChan and P2PChanHci.

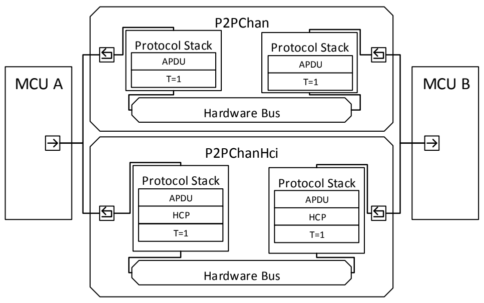


Figure 4: TLM based system implementation.

It can be noticed in Figure 4 that the protocol stack is the main difference between the two implementations. Considering the protocol stack, P2PChanHci adds an additional HCP layer. To make both approaches comparable, the used protocols in the single layers of the communication stack stay unchanged. Both approaches exchange the message packets via the same underlying hardware bus system.

For correct transmission, HCP builds on an underlying DLL, according to the requirements defined in [9]. As T=1 protocol [11] fulfills these requirements, it is used to transport the HCP packets. By using the T=1 protocol as DLL in both channel implementations, it has no influence on the difference in the communication delay. To evaluate the performance difference between `P2PChan` and `P2PChanHci`, the model was extended with timing annotation. The timing delay of the hardware bus was modeled by dividing the transmitted bytes through the configured data rate. The processing time of the messages parsed through the single layers of the protocol stack was measured on a real SE implementing the APDU and T=1 protocol. This measurement starts at receiving a C-APDU and ends with sending the R-APDU via the connected bus. The additional parsing delay of the HCP layer was estimated by a domain expert. Thus, a parsing delay for both `P2PChan` and `P2PChanHci` was arrived at, which was used for annotating the model implementations respectively. The setup delay for creating the HCI network was estimated considering the necessary HCP messages, the parsing delay of the HCP and T=1 protocol layers, and the data rate of the hardware bus model.

The implementations of the two approaches were evaluated by investigating their overall communication delay. To make both designs comparable to each other, the configuration for both approaches were kept unchanged. Thus, the difference in the overall communication time only depends on the additional header information per packet, the number of exchanged messages, and the processing time of the added HCP layer in the protocol stack. The communication delays were put into relation using the standard APDU command-response exchange as reference value. For `P2PChan` MCU A acts as the original master, MCU B as the original slave. As it can be seen in Table I, sending APDU commands from MCU B to A takes approximately twice as long as sending them the other way around, when using `P2PChan`. This is caused by the additional time spent for sending and parsing the empty APDU command and response in the beginning and at the end of each transaction. Exchanging APDU command-response pairs using the HCI based P2P approach, implemented in `P2PChanHci`, both directions need roughly the same amount of time.

Table I: Normalized performance estimation

P2P[A to B]	P2P[B to A]	P2P-HCI[A to B]	P2P-HCI[B to A]
100%	194%	105%	105%

V. Conclusion and Future Work

In this paper, a new method for achieving a P2P APDU exchange based on HCI is presented, allowing a bidirectional APDU command-response exchange. Neither the Application Layer protocol, nor the underlying DLL protocol is changed in our approach. Merely, the HCI architecture is extended by an additional gate definition. Comparing the presented protocol with an APDU only approach, a P2P APDU command-response exchange is performed with almost the same communication delay in both directions. The method proposed in this paper can be extended to other command-response based protocols. Interesting aspects for future works would be to design and evaluate a security extension for the HCI based P2P APDU protocol. The standard APDU message exchange is usually secured by using the Secure Channel Protocol '03' (SCP03) [12]. The cryptographic key exchange is either managed by using the SCP03 or the Secure Channel Protocol '11' (SCP11) specification [13]. The SCP03 was not developed for supporting a P2P APDU exchange. Therefore, the usage of them must be evaluated, next to other possible solutions, as described in [14] and [15].

Acknowledgment

Project partners are NXP Semiconductor Austria GmbH and the Technical University of Graz. This work was supported by the Austrian Research Promotion Agency (FFG) within the project UBSmart (project number: 859475).

References

[1] S. Androutsellis-Theotokis and D. Spinellis. A survey of peer-to-peer content distribution technologies. *ACM Computing Surveys*, (4), 2004.
[2] R. Smith. Instant Messaging as a Scale-Free Network. 2002.
[3] M. Waldman, A. D. Rubin, and L. F. Cranor. Publius: A robust, tamper-evident, censorship-resistant web publishing system. *In Proc. 9th USENIX Security Symposium*, (August), 2000.
[4] D. Bandyopadhyay and J. Sen. Internet of things: Applications and challenges in technology and standardization. *Wireless Personal Communications*, 58, 2011.
[5] W. Chen and S. Cai. Ad hoc peer-to-peer network architecture for vehicle safety communications. *IEEE Communications Magazine*, (4), 2005.
[6] M. J. Booysen, J. S. Gilmore, S. Zeadally, and G. J. van Rooyen. Machine-to-machine (M2M) communications in vehicular networks. *KSII Transactions on Internet and Information Systems*, 6(2):529–546, 2012.
[7] C. Vedat, O. Kerem, and B. Ozdenizci. *Near Field Communication: From Theory to Practice*. Wiley Publishing, 2012.
[8] NFC Forum. Logical Link Control Protocol Technical Specification NFC Forum. (LLCP 1.0), 2014.
[9] ETSI. TS 102 622 - V12.1.0 - Smart Cards; UICC - Contactless Front-end (CLF) Interface; Host Controller Interface (HCI), 2014.
[10] D. C. Black and J. Donovan. *SystemC: From the Ground up*. Kluwer Academic Publishers, 2004.
[11] ISO IEC. ISOIEC 7816-3. (4):27, 2006.
[12] GlobalPlatform. *GlobalPlatform Card Technology Secure Channel Protocol ' 03 '*. 2014.
[13] GlobalPlatform. *Secure Channel Protocol ' 11 '*. 2014.
[14] P. Urien. LLCPS: A new secure model for Internet of Things services based on the NFC P2P model. *IEEE 9th International Conference on Intelligent Sensors, Sensor Networks and Information Processing, Conference Proceedings*, (April), 2014.
[15] S. Turner. Transport layer security. *IEEE Internet Computing*, 18, 2014.

Message Encapsulation Pattern

Lukas Gressl
Graz University of Technology -
Institute for Technical Informatics
Graz, Austria
gressl@tugraz.at

Christian Steger
Graz University of Technology -
Institute for Technical Informatics
Graz, Austria
steger@tugraz.at

Ulrich Neffe
NXP Semiconductors Austria
Gratkorn, Austria
ulrich.neffe@nxp.com

ABSTRACT

How to change the communication behavior of devices participating in a network with an already defined topology? This question describes the fundamental problem, which the Message Encapsulation Pattern aims to solve. Network participants usually follow a client - server, or peer to peer based communication model. This model uses a dedicated application layer protocol, which defines the message types exchanged by the communicating devices. In client - server based models, the client sends requests to and receives responses from the server, thus constituting a one - way message exchange. This setup can be changed by using a different application layer protocol, or by tunneling the messages of the application layer protocol through an underlying bidirectional protocol layer. With both approaches, a peer to peer communication can be achieved. However, tunneling the existing application layer protocol has the advantage that the application does not need to change its message parsing. This tunneling approach can be found in various domains. It is not only used in the context of Internet services, but can also be found in the communication between the single components of System on Chips (SoCs), or the smart card to terminal interaction. This paper describes, how the Message Encapsulation Pattern works, its advantages and disadvantages, and how it can be used.

CCS CONCEPTS

• **Networks** → **Peer-to-peer protocols**; *Routing protocols*; *Transport protocols*; • **Computer systems organization** → System on a chip; Embedded software;

KEYWORDS

Bidirectional Communication Protocols, System on Chip, Embedded S oftware

ACM Reference Format:
Lukas Gressl, Christian Steger, and Ulrich Neffe. 2018. Message Encapsulation Pattern. In *23rd European Conference on Pattern Languages of Programs (EuroPLoP '18), July 4–8, 2018, Irsee, Germany.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3282308.3282326

1 INTRODUCTION

The way in which devices within a network communicate with each other is defined by their respective roles. Usually the direction in which the information flows from one device to another depends on whether it is a peer, a client, or a server. Clients and servers communicate with a certain network model, in which the client demands some data from the server by sending a request, on which the server sends back the requested data via a response. Contrasting this type of one way communication, peer devices are not restricted to one message type. Rather than sending only requests or responses, they communicate with each other on an equal level, sending and accepting both message types. Figure 1 depicts a system consisting of three separated devices, which are configured using both peer to peer (P2P) and client - server network model.

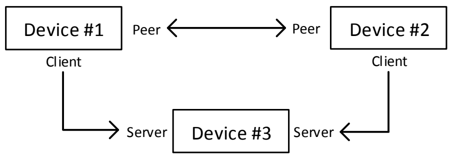


Figure 1: System architecture, in which the MEP can be used to allow a bidirectional communication between *Device#1* and *Device#3*, as well as between *Device#2* and *Device#3*

In this setup, the role of *Device#3* is restricted on responding to requests coming from *Device#1* or *Device#2*. If the role of *Device#3* should be changed from a passive communication participant to an active one, the restrictions of the client - server configuration must be faced. Such a situation might occur, if e.g. the behavior of *Device#3* is altered in a way that it receives a request from *Device#1* and, based on this request, sends a message to *Device#2*. With the current communication setup, this would not be possible. As the application layer protocol used by the single devices defines, whether the device acts as a client or a server, a valid solution for the problem of *Device#3* would be to change this protocol. Such a protocol change has a severe impact on the overall system, which might be infeasible. In such a systems, a P2P communication should be enabled, without changing the used application layer protocol. This would allow existing applications to stay unchanged and new ones to take advantage of the bidirectional message exchange, still using the well known request - response messages.

To overcome the protocol's restrictions on the communication direction without altering the application layer protocol, the pattern described in this work can be used. It describes a method to allow the transition of an application layer protocol used in a client - server to a P2P model, without changing the top layer protocol itself.

The paper uses the canonical way of pattern description. Therefore, the following sections explain the context of the proposed pattern (Section 2.1), the problem and its forces (Section 2.3), the solution and its accompanying consequences (Section 2.4), as well as known uses of the MEP, and other patterns related to it (Section 2.5).

2 MESSAGE ENCAPSULATION PATTERN

2.1 Context

Changing the direction of communication between two devices is of special interest for allowing more flexible usability of Secure Elements (SEs). SEs are usually used for storing confidential and sensitive data and, therefore, predestined for hosting banking or ticketing applications [14]. Usually, SEs are integrated in larger systems, which often comprise some kind of radio, as well as an additional application processor (AP) [1]. In such setups, the SE is usually a passive communication party, receiving requests from the AP or radio.

Considering the MEP, three patterns, which are used by the MEP and can be seen as prerequisites, must be mentioned. The communication participants are either clients, servers or peers. These roles are described by the *Peer-to-peer Architectural Pattern* and the *Client-Server Pattern* respectively [17] [18]. How data is encapsulated by the single layers of the network stack is described the *Protocol Layer Design Pattern* [4]. These patterns can be seen as prerequisites for the pattern presented in this paper.

2.2 Motivating Example

Consider an SE, connected to a radio and an AP. The AP interacts with some output device, allowing user interaction. The SE runs an application at a time, which, based on the information it gets from the radio, commits a secure transaction with a wireless reader. To enhance the user experience, the SE's application wants to send some state information to the AP, which it can process and present to the user. In a traditional setup, in which the SE acts as a server to both AP and radio, such a behavior could not be implemented, as the SE lacks a mechanism to actively inform the AP about its state because of its limitation due to the used application layer protocol. Figure 2 depicts such a system.

A possible solution would be changing to some other application layer protocol. The SE can host several applications, which can be selected and deselected. These applications parse incoming commands according to the specific message format of the used application layer protocol. Therefore, they are tightly coupled with the used application layer protocol and a change to the top layer protocol would constitute a severe impact, which might be infeasible. Furthermore, the overall communication speed between the two components should be fast. This is especially important for systems interacting with a user. Additional overhead, either by unnecessary message exchange, or by excessive header information

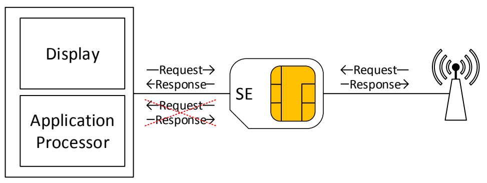


Figure 2: System consisting of a SE connected to a radio and an AP. The SE acts as server to both radio and AP. Requests can only be sent from the client to the server, which hinders the SE to actively inform the AP on a new state.

should therefore be avoided. The request - response pairs must not be broken, thus, the change of the communication direction should be performed in an organized way. This is especially important, as many applications communicating via such a message exchange expect receiving a response after sending a request to the communication partner.

One prerequisite of the pattern is, that the components are connected to each other via a physical link, which allows bidirectional communication. The main users of this pattern are designers, which are faced with a system already communicating in a client - server network model and want to change it to a P2P one, without changing the application layer protocol.

Considering the example with a system comprising an SE, the used application layer protocol comes with the additional restriction of not informing the receiver about the type of the message. The protocol lacks the message type information because for the server - client model it is simply unnecessary, as the server only expects responses, the client only requests. Thus, without sending information about the message type, neither client nor server would be able to distinguish a request from a response.

A possible method for enabling a bidirectional communication is depicted in Figure 3. SE encapsulates its request to the AP within the data field of its response. The AP encapsulates its response to the SE in the data field of a request. To initiate the protocol, the server must have some physical connection to the client trigger the sending of the first request, which is used to initiate the message exchange. In case both components are placed on an SoC, this physical connection can be a simple wire, used for triggering interrupts. To end the communication, the client needs to send a final response to the server. This response is needed to satisfy the protocol's rules. This method enables a bidirectional request - response exchange, but comes with major overhead on the overall transaction. The MEP shows an approach, which mitigates the additional communication cost.

This approach comes with several drawbacks. The initializing request and ending response adds to the overall communication delay. It influences not only the transmission delay, needed for exchanging the messages via a physical channel, but also the processing delay, needed for parsing and interpreting the message at both the sender and receiver. This communication initialization and closure induces additional latency when sending a request from the server to the client device. Therefore, this approach makes the request - response exchange from the server to the client less performant than the exchange in the other direction. As one communication direction should not be preferred over the other because of performance measures, this approach is not preferable. The dependency of the server to wait for the empty packet of the client to start its own request exchange, shows another problem with this approach. As it does not handle the switch in the communication direction, the server cannot be sure if it sending a request will be possible.

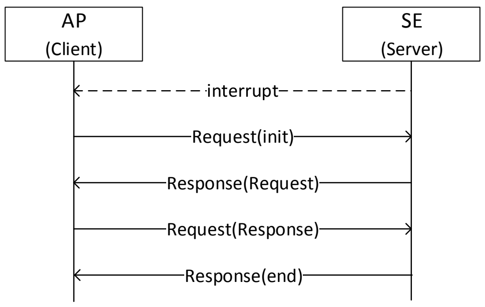


Figure 3: Message flow between Server and Client. Sending a request from the client to the server and a response in the other direction (as intended) is no problem. Using the same protocol for exchanging the request - response pair the other way around needs additional messages.

2.3 Problem

The problem, which must be solved, can be phrased as: *"How to change the communication behavior within an existing network model between two participants, without changing the application layer protocol and keeping the communication overhead as low as possible?"*

Typically, devices communicating with each other are configured either in a client - server or P2P model, defining the message types and the direction of the communication. The message types are defined in the application layer of the used protocol. Once the application layer protocol is defined, it is difficult to switch the network model without changing the this top layer protocol. Especially in a client - server network the direction of the message exchange is inflexible, because the direction, in which certain message types are transmitted, is limited. This limitation comes from the used application layer protocol, which defines two types of messages: requests and responses. The client asks for information from the server by sending a request, and the server answers this request by sending back a response. If a designer wishes to allow a bidirectional communication between the client and the server, a change towards a P2P networking model must be performed. Such a change can be performed by using a different application layer protocol, which might cause major changes for already existing applications, and might therefore not be desired by the designer. Considering also the solution approach from the *Motivating Example*, the problem comes with the following forces.

- **F1** The message encoding in the application layer protocol must not change. Completely changing the top layer protocol would mean that all applications hosted by the device must change their message parser implementation.
- **F2** The performance of the overall communication must not be increased drastically. This is especially relevant for scenarios in which user experience is important, or which have to react to real time events.
- **F3** No unnecessary messages shall be sent only for complying with the protocol rules. Thus, no additional transmission or processing delay is produced.
- **F4** Both server and client should be able to switch the communication's direction, either by requesting the active part, or by indicating that no more requests will be sent. Preferably, switching the direction should not break up an already ongoing request - response pair exchange, and should offer both participants to equally negotiate the new direction.

2.4 Solution

The MEP solves the problem by adding some additional information to the transmitted messages, wrapping the actual requests and responses. Figure 4 shows this approach. The MEP encodes the actual request and response using some message type of an underlying protocol. This underlying protocol must offer bidirectional communication. To ensure standard compatibility, the encoding message types should be taken from an already well established protocol. Wrapping the request and response by using a message format of some other protocol, additional header information of the encapsulating message type must be transmitted, causing communication overhead. Therefore, this method is only feasible, if the additional header information causes less communication delay than the exchange of an extra request - response pair. From a protocol stack perspective, the MEP adds another layer to the communication stack.

With this message encapsulation, the MEP enables the transformation of a client - server model to a P2P one by decorating the exchanged requests and responses with some additional information. Thus, the exchanged packets become bigger, as additional information must be transmitted per packet. It also means, that a further layer is added to the communication stack and that additional parsing of the meta data, surrounding the request and response, must be performed by a new driver. The benefit is, that the channel traffic caused by the additional exchange of the empty messages is evaded. The new parts in the protocol stack are shown in Figure 5.

The parser, which decorates the messages with additional information, implemented the *Decorator* design pattern [12]. By adding to and parsing the type information from the messages, the application can be informed, if the received message is a request or a response.

Considering the approach presented in the context of the SE to AP communication, Figure 6 shows a comparison between the MEP's solution and the request - response encapsulation approach described in the context and depicted by Figure 3. It can be seen that,

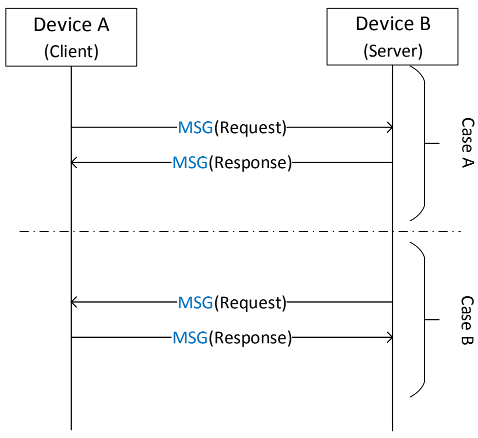


Figure 4: Solution to the problem stated in Section 2.3. The request - response exchange is encapsulated by a bidirectional message format of an underlying protocol in the communication stack. This allows a bidirectional message exchange.

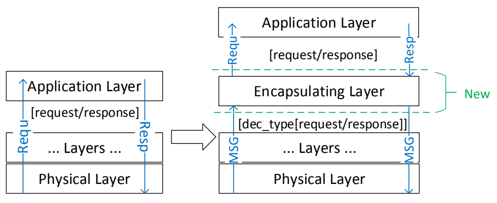


Figure 5: The protocol stack is extended by an additional layer marked with the "'New"' bracket. The request / response messages are decorated with additional meta information visualized by the change from request / response to MSG in the right hand protocol stack representation

by using the bidirectional message type of the underlying protocol for wrapping the request - response messages, the MEP approach saves the initializing request and the ending response in *Case B*. This results in halving the amount of exchanged messages in this *Case*. However, the additional header information of the wrapping message type must be taken into consideration, when looking at the additional transmission delay in *Case A*. The consequences of the MEP's solution approach are explained in more detail in Section 2.4.1

2.4.1 Consequences. Considering the solution and the forces they must meet, one may notice that the MEP's solution meets first three forces (**F1**, **F2** and **F3**) by encapsulating the request and response messages by using a message type of an underlying bidirectional protocol. As can be noticed in Figure 4, the encapsulation is performed in both *Cases A* and *B*. Force **F4** must be complied to by using a mechanism, which enables the active communication party to inform the passive one that no more requests remain outstanding. Thereafter, the passive party might inform the active one about a change in the communication direction, or remain in its passive role, waiting for further requests. Such a mechanism should be provided by the protocol the requests and responses of the application layer are decorated with. Concluding, the benefits and liabilities of the MEP are listed below.

The benefits of the Message Encapsulation Pattern are:

- No empty requests or responses are needed to fulfill the application layer protocol of the server - client topology. This solves force **F3**.
- By evading the unnecessary transmission of empty packets, the overall performance is increased. This solves force **F2**.
- No change in the application protocol layer is necessary, as the on top application still works with requests and responses. The encapsulation is abstracted by the intermediate parser. This solves force **F1**.
- Using the bidirectional message type of the encapsulating protocol, designing a special notification for switching the communication direction is can be achieved using the new intermediary layered protocol. This solves force **F4**

The liabilities of the Message Encapsulation Pattern are:

- Additional meta information is needed to allow the bidirectional exchange of the request - response pairs. This increases the packet size transmitted on the physical layer and mitigates the performance increase. This affects force **F2**.
- For parsing the additional meta information, with which the request and response pairs are decorated, and unpacking the encapsulated request or response, a driver, situated below the application layer, is necessary. This also affects force **F3**.
- In general, adding an additional layer in the protocol stack needs a parser to be implemented for constructing and parsing the new header information. New code is always a further source for software bugs. Adding this protocol layer to already existing devices means that unit and system testing must be extended to ensure proper functionality.

2.5 Known Uses

The following sections describe known uses of the MEP. Section 2.5.1 describes the use of the MEP within the communication between multiple components of an SoC. Section 2.5.2 describes the usage of the connection less User Datagram Protocol (UDP) to decorate all other transmission protocols, such as the Transmission Control Protocol, to allow P2P communication. Section 2.5.3 shows another way of implementing the MEP by combining the functionality of an NFC Push Protocol server and client into one device and using the P2P mode of the Logical Link Control Protocol, tunneling the respective request and response messages.

2.5.1 P2P APDU exchange. In the smart card domain, a terminal communicates with a corresponding chip card by exchanging Application Protocol Data Units (APDUs) via an either contact based

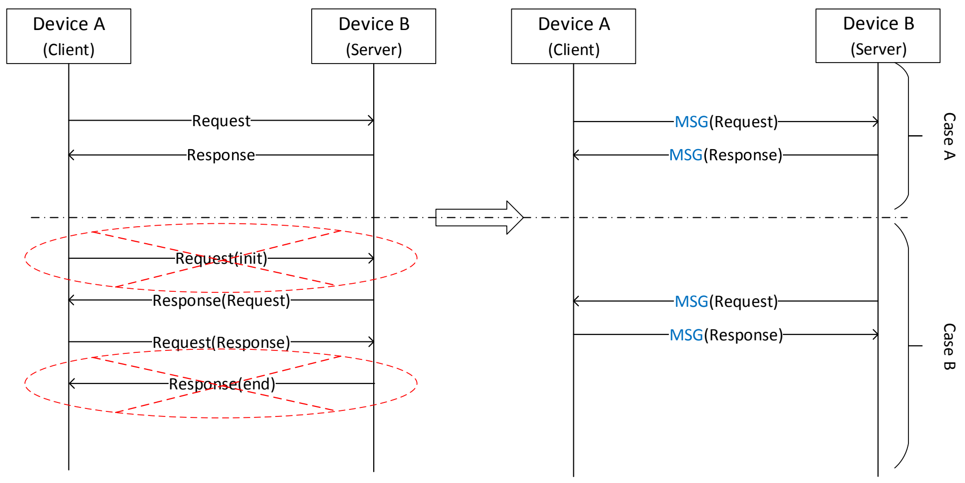


Figure 6: The MEP's solution in comparison with the approach described in the context. The request - response exchange initialization, which is crossed out in red, can be replaced by encapsulating the actual request - response pair using the message format of an underlying protocol in the communication stack, which allows a bidirectional message exchange.

or contact-less physical connection. This APDU format, which is defined by ISO/IEC, only knows commands (C-APDUs) and responses (R-APDUs). Furthermore, the APDU protocol divides the communicating parties into a master and a slave. The master only sends C-APDUs and interprets R-APDUs, the slave only interprets C-APDUs and responds with corresponding R-APDUs. In between the APDUs, which are located at the application layer, and the physical link, either the T=0 or T=1 data link layer protocol is used to guarantee correct transmission of the exchanged command - response pairs. Both T=0 and the T=1 protocols are by ISO/IEC as well [7] [8].

The Message Encapsulation Pattern can be used to transform the master - slave topology, predetermined by the used APDU application layer protocol into a P2P network. The C-APDUs and R-APDUs are decorated by using a bidirectional asynchronous message type of a communication layer in between the APDU and the T=1 or T=0 protocol. The used message type is taken from the Host Controller Interface (HCI) standard. The HCI standard defines a protocol, which can be localized in the network layer of the protocol stack. Its main purpose is to allow the setup of a network, consisting of multiple embedded devices, and allow the routing of packets through a network of so called logical pipes. The HCI protocol knows command and response pairs, but also uses an asynchronous message type, called event. This event type is used to allow a the C-APDU - R-APDU exchange via the logical pipes of the network by decorating the commands and responses with the event type message [3].

The HCI standard furthermore introduces gates, which are used to encapsulate the access to certain services. For exchanging APDU command - response pairs, the latest HCI standard defines the APDU gate, used by the server, and the APDU application gate, used by the client. The gate controls, which kind of events, either C-APDU event or R-APDU event, is allowed to pass the respective gate. The APDU gate only interprets C-APDU events, the APDU application gate only R-APDU events. With a small modification of the HCI standard, a bidirectional exchange of the C-APDU and R-APDU events is possible, thereby modifying the client - server to a P2P topology. This modification consists of the additional definition of a dedicated P2P-APDU gate, allowing the interpretation of both the C-APDU and the R-APDU event. This approach is described in [6]. Figure 7 shows this setup.

2.5.2 OverUDP. In [9], Janbeglou and Brownlee present a method for tunneling transport layer protocols in the UDP [2] for P2P Applications. The authors describe their method, called *OverUDP*, in which they use UDP to tunnel all transport layer protocols, such as the Transmission Control Protocol, to be used in P2P communication. The UDP was chosen by the authors, because it is an connection less protocol, and, therefore, suitable for allowing home Internet users to communicate with each other directly. Furthermore, UDP is very common in P2P communications. Using the *OverUDP* each participant in the P2P communication encapsulates all network packets, regardless of the transport layer protocol, into UDP datagrams and sends them via the network. The receiving party receives the UDP datagrams, decapsulates the network packets from the datagrams' payload and injects them into the device's network data flow.

To setup the UDP connection, *OverUDP* uses the Session Traversal Utilities for Network Address Translation (STUN), as, according

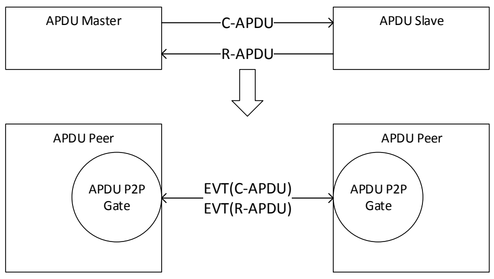


Figure 7: Block diagram showing the difference before and after applying the Message Encapsulation Pattern to the APDU communication. The master - slave topology is transfered to a P2P one. The C-APDUs and R-APDUs are decorated using the events of the HCI protocol

to [16], works best for UDP, because it is a pure P2P solution. STUN, which is not only used by *OverUDP*, but also with Transmission Control Protocol (TCP) [5], or Web Real-Time-Communication (WebRTC) [13], offers the establishment to a server connection. This STUN server is used by both communicating devices to get their respective endpoint information, containing the public IP address and the port. This information is exchanged between the two parties. In the following communication, the port and IP address information is used to transmit the messages via the UDP channel, established between the two endpoints. Figure 8 depicts this method.

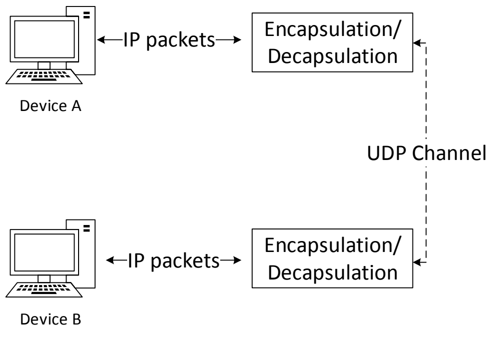


Figure 8: *OverUDP* used to tunnel messages between two communication partner (Device A and B), thereby achieving P2P communication, adapted from [9]

2.5.3 NFC Push Protocol P2P Communication. The NFC Push Protocol (NPP), which is specified by Android, is used for pushing NFC Data Exchange Format (NDEF) messages from one device to another. This communication is a one way transmission of NDEF messages from an NPP client to an NPP server. The NDEF messages are transmitted via the Logical Link Control Protocol (LLCP), which itself offers an P2P mode [15]. Any device which uses the NPP must run an NPP server, to which another device using an NPP client can connect [10]. In this setup the roles of the devices are fixed to an active or passive part, depending on whether they utilize an NPP server or client. To allow a bidirectional message exchange between two NFC devices, each device must run an NPP server, as well as an NPP client. Thus an bidirectional message exchange between the two devices can be achieved [11].

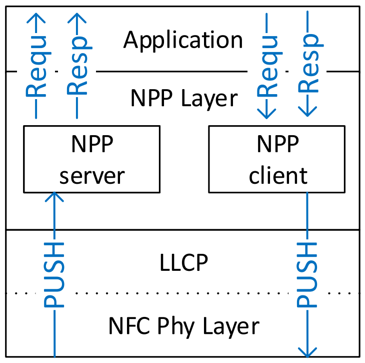


Figure 9: P2P communication enabled by switching between an NPP client and server running on the same device, adapted from [11]

As the two communicating devices switch between the active and passive role within the communication, the top layer application uses the NPP server and client to encapsulate the exchanged messages. This utilization of the NPP layer follows the principal of the MEP. Figure 9 visualizes the protocol architecture.

ACKNOWLEDGMENTS

I would like to thank my shepherd Victor Sauermann, who did an excellent job in helping me get my paper ready for the workshop. I also want to thank my workshop group, especially Christopher Preschern, who gave me feedback and advice not only during, but also after the conference. They both greatly helped improving the overall paper.

REFERENCES

[1] Vincent Alimi, Marc Pasquet, and Laboratoire Greyc. 2009. Post-distribution provisioning and personalization of a payment application on a UICC-based Secure Element. (2009), 701–705. https://doi.org/10.1109/ARES.2009.98
[2] Lars Eggert and Gorry Fairhurst. 2008. Unicast UDP Usage Guidelines for Application Designers. RFC 5405. https://doi.org/10.17487/RFC5405
[3] ETSI. 2014. TS 102 622 - V12.1.0 - Smart Cards; UICC - Contactless Front-end (CLF) Interface; Host Controller Interface (HCI).

[4] Eventhelix. 2018. Protocol Layer Design Pattern. https://www.eventhelix.com/RealtimeMantra/PatternCatalog/protocol{_}layer.htm{#}.W0dQIMKxVEY

[5] B. Ford, P. Srisuresh, and D. Kegel. 2005. Peer-to-peer communication across network address translators. *Usenix.Org* (2005), 1–31. https://doi.org/10.1109/CEC.2007.4424785 arXiv:cs/0603074

[6] Lukas Gressl, Ulrich Neffe, and Christian Steger. 2018. Design and Implementation of an HCI based Peer to Peer APDU Protocol. *Proceedings - 21st Euromicro Conference on Digital System Design DSD 2018* (2018).

[7] ISO IEC. 2006. ISOIEC 7816-3. 4 (2006), 27.

[8] ISO IEC. 2014. ISOIEC FDIS 7816-4. (2014), 162. www.iso.org

[9] Maziar Janbeglou and Nevil Brownlee. 2016. OverUDP: Tunneling transport layer protocols in UDP for P2P application of IPv4. *Proceedings - IEEE 30th International Conference on Advanced Information Networking and Applications Workshops, WAINA 2016* (2016), 325–330. https://doi.org/10.1109/WAINA.2016.41

[10] A. Lotito and D. Mazzocchi. 2012. OPEN-NPP: An open source library to enable P2P over NFC. *Proceedings - 4th International Workshop on Near Field Communication, NFC 2012* (2012), 57–62. https://doi.org/10.1109/NFC.2012.16

[11] R. Mainetti, L. and Patrono, L. and Vergallo. 2012. IDA-Pay: An innovative micro-payment system based on NFC technology for Android mobile devices. *Software, Telecommunications and Computer Networks (SoftCOM), 2012 20th International Conference on* 8, 4 (2012), 1–6.

[12] James E McDonough. 2017. *Decorator Design Pattern.* Apress, Berkeley, CA, 207–224. https://doi.org/10.1007/978-1-4842-2838-8_16

[13] Dr. David A. McGrew and Eric Rescorla. 2010. Datagram Transport Layer Security (DTLS) Extension to Establish Keys for the Secure Real-time Transport Protocol (SRTP). RFC 5764. https://doi.org/10.17487/RFC5764

[14] David M Monteiro, Joel J P C Rodrigues, and Jaime Lloret. 2012. A Secure NFC Application for Credit Transfer Among Mobile Phones. (2012).

[15] NFC Forum. 2014. Logical Link Control Protocol Technical Specification NFC Forum. LLCP 1.0 (2014).

[16] J. Rosenberg, J. Weinberger, C. Huitema, and R. Mahy. 2003. STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs).

[17] Unified-am. 2018. Reusable Asset: Client-Server Pattern. http://www.unified-am.com/uam/UAM/guidances/reusableassets/uam{_}pattern{_}client-server{_}2E388191.html

[18] Unified-am. 2018. Reusable Asset: Peer-to-peer Architectural Pattern. http://www.unified-am.com/uam/UAM/guidances/reusableassets/uam{_}pattern{_}p2p{_}144410EC.html

ICONS 2019 : The Fourteenth International Conference on Systems

A Security Aware Design Space Exploration Framework

Lukas Gressl
Institute of Technical Informatics
Graz University of Technology
Graz, Austria 8010
Email: gressl@tugraz.at

Christian Steger
Institute of Technical Informatics
Graz University of Technology
Graz, Austria 8010
Email: steger@tugraz.at

Ulrich Neffe
NXP Semiconductors Austria GmbH
Graz University of Technology
Email: ulrich.neffe@nxp.com

***Abstract*—System designers are often faced with a huge variety of alternative hardware platforms and architectures, when designing new products. Especially the various options for allocating a set of tasks to processing units greatly influences the overall system performance and power consumption. As the possible design space is too complex for manual evaluation, automatic Design Space Exploration (DSE) tools are used for selecting first system designs. These tools assess the various mappings between tasks and processing units. They target the best allocation, optimizing the system's performance and power consumption, while considering other predefined design constraints. Traditionally, security requirements do not belong to the set of design constraints these tools deal with. Thus, security requirements must be introduced manually, which might induce additional costs to the overall project. To enable security-by-design using DSE, the Security Aware Design Space Exploration (SADSE) Framework was developed. This framework allows the integration of attack scenarios and security requirements, as well as platform security features into the DSE, at a level of detail not yet considered by other tools. SADSE allows an optimal allocation of tasks onto hardware platforms, while satisfying predefined security constraints. This paper shows how security requirements and attack vectors are modeled in SADSE, followed by the evaluation of a keyless entry system use case, where the tool finds a secure mapping of tasks to processing units.**

***Keywords*–Security; Design Space Exploration; Embedded Systems.**

I. Introduction

Designing a new product means making a lot of decisions, ranging from which hardware components to take to what system functionality and on which component to place them on. This variety opens up a huge space of alternative designs which must be considered by designers, system architects and product owners. The resulting design influences the power and performance characteristic. This design choice is an issue, especially in the domain of embedded systems and stretches from selecting hardware components to mapping of system functionality. The optimal allocation of system functionality to dedicated hardware blocks, such as special hardware or general purpose processors, poses a complex problem. This allocation cannot be solved manually regarding more than one characteristic. To tackle this problem and to shorten the design process, automatic Design Space Exploration (DSE) tools are used. These tools scan a space of alternative designs and allocation options, and compute an optimized solution.

Especially for devices in the domain of the Internet of Things and Cyber Physical Systems (CPS), the information security plays a vital role. CPS sense data and handle confidential or even personal information, which imposes security requirements to these devices. The security requirements are usually defined by an expert, and depend on the project setup. These requirements are considered right at the beginning or integrated later. Later on integration of security requirements increases the project's costs significantly more than introducing security at the beginning of the design flow. Therefore, most companies, producing secure products, introduce security requirements initially at the design phase. In this phase, DSE tools can be used to support designers in their choices. As traditional DSE tools lack the ability of considering information security, their usability for designing secure products is limited.

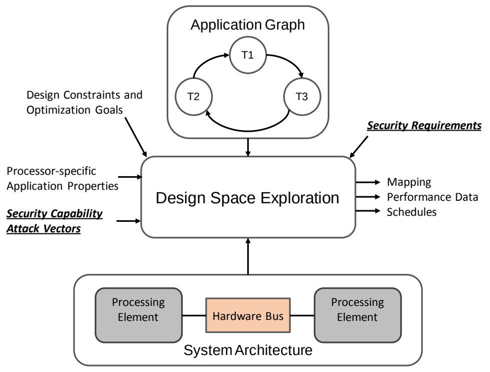


Figure 1. DSE process, based on [1] extended by security requirements, security capabilities, and attack vectors.

To bridge this gap in the design flow, we present a Security Aware Design Space Exploration framework (SADSE). The SADSE framework allows an automatic DSE under consideration of security requirements and threat scenarios. The framework offers the designer to define security requirements for single data entities the system tasks operate on, and attack scenarios for the individual function blocks. Given these security requirements and attack scenarios together with the defined hardware platform, the framework performs an optimized allocation of functionality to hardware blocks. Thereby, it considers the security requirements and the hardware components' security capabilities. Figure 1 shows an overview of the traditional DSE process extended with the additional security assets introduced in this paper.

The basis of the SADSE framework implementation is the

Constraint Programming (CP) based Design Space Exploration for System Design tool, as described in [2]. The extension of considering security constraints in the DSE is presented in this paper and the SADSE framework's functionality is evaluated using an embedded access control device as a use case. The rest of the paper is structured into the following sections: in Section II previous work considering DSE and security requirements is presented; in Section III the methodology introduced by this paper is explained in detail; in Section IV the SADSE framework is used to evaluate a secure task mapping of a keyless entry system and the framework's performance is evaluated; in Section V a conclusion is drawn and future work is discussed.

II. Related Work

The optimal allocation of tasks to hardware components considering the overall system execution time, power consumption, scheduling, etc. is a well described problem for embedded devices, multiprocessor- and multicore-systems.

Other works already proposed frameworks performing automatic DSE for embedded systems under consideration of hardware software codesign. The optimal allocation of streaming applications onto a heterogeneous multi-processor system is investigated in the works of Khalilzad et al. [3], and Rosvall et al. [1] [2]. In the framework proposed by these authors, streaming applications are represented as synchronous data flow graphs, and their tasks are mapped to distinct heterogeneous processors. The framework describes the problem of the optimal mapping of tasks to processors as a constraint satisfaction problem, which is solved by using CP. Finding the optimal hardware-software split for embedded devices using heuristic algorithms in DSE was investigated by Knerr [4]. In his work, Knerr considers the problem of finding the best partitioning of functionality implemented in software and hardware components, considering a predefined hardware platform. His approach considers various optimization criteria, such as chip area size, power consumption or performance.

Security requirements in DSE are described in a range of modeling and analysis techniques. In this area, the work of Kang [5], Stierand et al. [6], and Hasan et al. [7] are prominent. In their work, Hasan et al. consider an already existing task schedule of an real time operating system on a predefined multicore-system. The authors present a framework allowing to insert security tasks into this schedule without changing it and without breaking the system's real time constraints. Kang describes a tool which supports system designers in their decisions considering the correct use of security features.

Stierand et al. [6] present a framework in which security parameters are introduced into automatic DSE, putting it into the context of the automotive domain. They focus on the communication part between tasks, assessing the attack vulnerability of the channels connecting them. This vulnerability is determined by the capability of the attacker. Thus, they add security requirements to the exploration. To mitigate these attacks, Stierand et al. propose to map these vulnerable tasks onto architecture modules providing hardware security extensions, ensuring that such attacks cannot be performed. As the task model of their approach combines functionality and data as one, the correct mapping of the single tasks to hardware secured electronic control units depends on the definition of what operations a task executes on some piece of information.

From the described DSE tools, only Stierand's framework focuses on the correct allocation of security vulnerable tasks on dedicated hardware components during an automatic DSE. In comparison to their work, the framework presented in this paper regards data and control flow separately. With this separation, multiple interpretation variants of a task functionality are overcome. This allows a more detailed attribution of security requirements to the respective information blocks. Furthermore, we do not regard these security requirements exclusive to the communication channels between tasks. Our approach pursues a more holistic way of introducing security into automatic DSE. We consider the attack scenarios not only on the communication but also on the tasks, and the architectural blocks and assign security attributes to the data used by the tasks. Furthermore, by assigning security levels to the distinct hardware elements, we do not simply solve a mapping problem, but are also able to find a suitable platform configuration. Section III discusses the proposed approach in detail.

III. Proposed Methodology

Performing an optimal allocation of functionality to a predefined system architecture under consideration of security constraints needs a way of accurately defining tasks, architectural blocks, and security constraints. This section introduces the necessary components and describes the underlying constraint solving problem of the SADSE framework.

A. Representation of the System Functionality

According to [4], the functionality of a system is defined by a directed process or task graph in which the nodes represent functional elements, and the edges represent data transfers between those elements. This combined representation of data and functionality in one task leads to ambiguous results when attempting to define security requirements on it. Therefore, we split task functionality and data in our approach. The SADSE framework allows the definition of distinct security requirements on data entity without mingling it with the task's functionality. Each task is linked to a data entity by a set of operations. This more precise modeling of the control and data flow of the system enables the framework to perform a more comprehensive mapping. This explicits control and data modeling leaving less space for interpretation what a task is actually doing with its associated data. This is important when it comes to the decision of where to map tasks that handle secure data.

By splitting functionality and data each single data block can be attributed with security requirements, determining how the data must be secured. This assignment must be performed by a security expert based on a *Confidentiality-Integrity-Availability* (CIA) triad [8].Our approach focuses on the confidentiality, the integrity and the authenticity of the data entity. Therefore, the security requirement for a data unit is denoted as the tuple $sr = (conf, int, auth)$, where $conf$, int, and $auth$ can either be 0 or 1. Combining the definition of the security requirements (sr) with the operations, a data entity basically defines a set of operations and security requirements. For better readability, a task performing a set of operations on a set of data entities is defined as a process. The security requirements for each data entity must be determined by the designer and serves as an input to the SADSE framework.

Figure 2 shows the separation of task functionality and security attributed data connected by a range of operations. The system's functionality is represented as a directed process graph. The data flow in this graph consists of the set of all tasks operating on the same data entity. As each process operating on the same data entity is connected via an edge to its parent process, the data flow can be found by traversing all parent processes with the same data entity attribution.

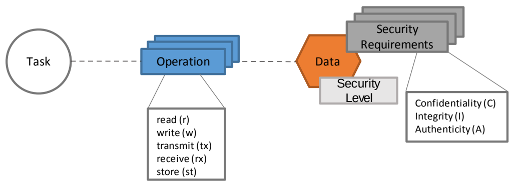


Figure 2. Representation of a process, consisting of a task, its associated data attributed with security requirements, and the task's operation performed on the data.

Considering the set of operations, the basic operations on the data entities, such as read (r), write (w), transmit (tx), receive (rx), and persistently store (st) are directly defined by the designer. This set of operations is represented by the tuple $op = (r, w, tx, rx, st)$, where each element can either be 0 or 1. The security related operations are derived from the basic operations and the data entity's security requirement. Additionally to the security requirement, each data entity is assigned a security assurance level, which must be evaluated by a domain expert. For simplification, these assurance levels are abstracted as an integer ranging from 0 to 3, with 3 representing the highest security level and 0 no security. Any task reading or writing confidential data must decrypt the data before processing it and encrypt it before passing it to another task. The same principle applies to the authenticity of data, which must be ensured by applying a signature or authentication code after writing and verified before reading. Transmitting and receiving of secured data does not enforce any security operation. These security operations $op_{sec} = (enc, sign, st_{sec})$ are derived according to (1). These operations and security assurance levels must be mapped to the security capabilities of the individual Processing Elements (PEs) which are explained in detail in the next section.

$$op_{sec}(op, sr) = \begin{pmatrix} (r \vee w) \wedge conf \\ (r \vee w) \wedge auth \\ st \wedge (auth \vee conf \vee int) \end{pmatrix} \quad (1)$$

B. Representation of the System Architecture

The hardware platform is represented by PEs, which are connected to each other via Hardware Bus Systems (HWBs). PEs can represent general purpose processors or application specific integrated circuits. PEs and HWBs are assigned distinct characteristics and attributes. The set of attributes for PEs are chip area, memory size and power consumption, whereas the attributes for HWBs are power consumption and transmission speed. PEs are further characterized by their security capabilities. They describe the PEs capability on cryptography ($crypt$), verification ($verify$), and tamper resistant storage (trs), which is described by the tuple $sec_{cap} = (crypt, verify, trs)$, where $crypt$, $verify$, and trs are abstracted by a security capability level ranging from 0 to 3, 3 being the highest security capability level and 0 meaning no security capability. These capabilities are implemented by additional hardware or software modules. The distinction of a software or a hardware implementation is performed by the attribution of the PE. A hardware implementation may increase the chip area, whereas a software implementation might shrink the available size of memory. Thus, a PE can be formalized as a set of modes, in which each mode defines $sec_{cap}s$ and the corresponding attributes. An HWB can be defined as a set of characteristics and modes. Furthermore, not all PEs are directly connected to one another. Any two PEs are connected via a hardware bus. With these definitions, an architectural platform can be described. Figure 3 depicts two PEs connected to one another by a hardware bus with their respective attributions.

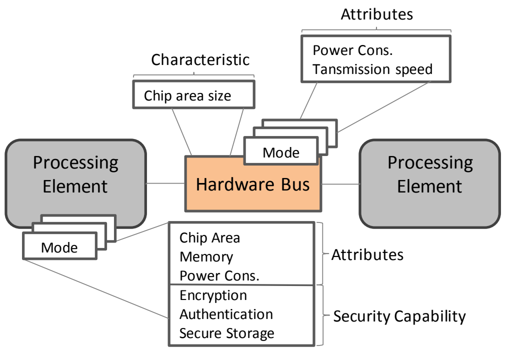


Figure 3. Hardware platform representation, consisting of PEs, connected by a hardware bus.

C. Attack Vectors

For determining the attack vectors on the system entities, we use the STRIDE analysis [9]. We focus on the attacker capabilities of spoofing (S), tampering (T), and information disclosure (ID), which can be either 0 or 1. These attack vectors, described by the tuple $av = (ID, S, T)$ can be directly mapped to sr, as spoofing affects the authenticity, tampering the integrity, and information disclosure compromises the confidentiality of the data. From the assets that can be attacked, we focus on processes, data stores, and data flows. In our approach, processes from the STRIDE analysis are simply processes p, data stores are represented by PEs, and data flows are the set of processes operating on the same entity of the data. Therefore, the av on a data flow is the combination of av of the involved processes. The susceptibility of the physical connections between the PEs is integrated into the attack vectors of the respective PEs. The attack vectors are defined by the designer.

The combination of the security requirements of the single data entities, the operations performed on the data entities by the processes, and the attack vectors form the basis on which the SADSE framework performs the mapping of the processes to PEs. Thereby, the SADSE framework considers the PEs' security capabilities. The mapping, and its influences on the overall system performance is explained in the next section.

D. Mapping Functionality to Architecture: A Constraint Satisfaction and Optimization Problem

Mapping the functionality of the system described by the process graph to the system architecture and selecting the optimal PE-modes is a typical application of combinatorial optimization. These classes of problems can be solved using CP. At the core of the CP method, a set of decision variables describes the problem at hand. Each of these decision variables has a certain domain of possible values. The variables depend on one another, described by the constraints. These constraints determine which combinations of values within the domain of the single variables are allowed. A constraint solver is used for finding an optimal mapping of the processes on PEs, satisfying all constraints [10]. The framework basically distinguishes between two types of design constraints - constraints to satisfy and to optimize.

Constraints to be satisfied are the security requirements and communication feasibility cf. The communication between two processes is only realizable, if either both processes are allocated on the same PE, or, if allocated on two different PEs, there is an HWB connecting them. The security constraints are a combination of av, sr, and op_{sec}. For each process mapped to a PE, the security constraints $sc = (sc_{enc}, sc_{auth}, sc_{store})$ are calculated according to (3). The attack vectors of the process and of the PE, on which the process is mapped, are denoted av_p and av_{PE}, respectively. OP_{sec}, defined in (2) is the result of all security operations performed by process p, and n is the number of all data entities p operates on. Furthermore, $asslvl_{sec} = (enc_{lvl}, sign_{lvl}, st_{lvl})$ stores the maximum security assurance level of the data entities these security operations are performed on.

$$OP_{sec} = \begin{pmatrix} enc_1 \vee \cdots \vee enc_n \\ auth_1 \vee \cdots \vee auth_n \\ st_{sec1} \vee \cdots \vee st_{secn} \end{pmatrix} \tag{2}$$

$$sc(av_{PE}, av_p, OP_{sec}) = \begin{pmatrix} (ID_{PE} \vee ID_p) \wedge enc \\ (S_{PE} \vee S_p) \wedge auth \\ (T_{PE} \vee T_p) \wedge st_{sec} \end{pmatrix} \tag{3}$$

$$sc_{lvl}(asslvl_{sec}, sc) = \begin{cases} enc_{lvl} & sc_{enc} > 0, \\ sign_{lvl} & sc_{auth} > 0, \\ st_{lvl} & sc_{store} > 0 \end{cases} \tag{4}$$

Equation (4) is use to calculate the security constraint levels for each process. For each level, there exists a sec_{cap} provided by the PE, which ensures the data's security requirement and mitigates the attack vector, satisfying the data's security levels. This mapping function map^p_{PE} is denoted by (5), and must be performed for all possible mappings of processes to PEs. Only if all mappings return 1, the security constraints are satisfied.

$$\begin{aligned} map^p_{PE}(sec_{cap}, sc_{lvl}) = (crypt \geq enc_{lvl}) \wedge \\ (verify \geq sign_{lvl}) \wedge (trs \geq st_{lvl}) \end{aligned} \tag{5}$$

Constraints to be optimized can be the power consumption of the overall system, the chip area size, as well as the system's performance. The performance is calculated considering the tasks' Worst Case Execution Times (WCETs). The WCETs reflect the processing delays of a process executed on a PE, for which an implementation exists or which can be estimated by the designer. More specifically, a process' WCET must be estimated or known for a PE's mode to be considered for the mapping by the SADSE framework. The security capabilities induce additionally computational overhead, which influences the overall execution time, depending on the process mapping the SADSE framework performs. Furthermore, the designer can specify different modes for each PE, which is also explored by the tool. Additionally to an optimal mapping of processes to PEs an optimal selection of the PE modes is done.

Depending on the situation and its requirements on execution time, power consumption, etc., one implementation would be preferred over the other. The framework performs an automatic and optimal mapping of the required functionality to the respective implementation alternatives, considering their performance, power consumption, needed memory, and gate size, and ensuring that the security hardness characterization fulfills the needed attack mitigation as defined by the designer.

E. SADSE Framework Implementation

As basis of the DSE tool, we used the work of Rosvall et al. [1]. The data blocks, operations, attack vectors, security requirements, security capabilities, and security levels were added to the platform and function graph representations. The network system was extended by a configurable bus system. The restrictions imposed by the bus system, and the security features were implemented as additional constraints and included into the CP model. The additional delay caused by the individual security features is added to the calculation of the overall execution time.

IV. Use-case Evaluation and Results

The SADSE framework was evaluated by performing a performance optimized mapping of a keyless entry system. The system's functionality was derived from the systems described in [11] and [12]. The system consists of a lock and a device. The device's functionality and architecture is described in here. The device builds up a connection with the lock by receiving a request $requ^{lock}_{chall}$ from the lock. The device creates a challenge $resp^{dev}_{chall}$ using its master key key_{master} and sends it to the lock. The lock sends its own challenge $resp^{lock}_{chall}$ which is checked by the device, again using key_{master}. The device derives a long time key key_{lt} from key_{master} and sends an ready request $requ^{dev}_{ready}$ to the lock. It receives a response from the lock $resp^{lock}_{ready}$ stating that it is ready to open a session. The device informs the user, requesting an action $action_{user}$. It then derives a session key $key_{session}$ from key_{lt} and creates an open request $requ^{dev}_{open}$ using $key_{session}$.

The hardware platform considered for the analysis is represented by a device, consisting of an application processor, a secure element, a micro controller, and a Bluetooth Low Energy (BLE) radio. All components are connected with each other by a bus system. The functionality mapped to the device establishes an authenticated and secure connection between itself and an external lock. Therefore, it uses key_{lt} and $key_{session}$. The key_{lt} is negotiated between lock and device. It is used as long as lock and device are paired. The $key_{session}$, which is derived from key_{lt}, is used for the authentication between lock and device and is updated frequently. Hence, disclosure of the $key_{session}$ poses a less severe security impact to the access system. Figure 4 shows the system's task graph and hardware architecture. The mapping is performed based on the task's WCETs when running on the individual PEs, and their security constraints. The goal of the SADSE framework is now to find the optimal mapping of the tasks to the PEs, running in a specific mode, as well as an optimal selection of the PEs. This selection is optimal if the overall system's execution time is minimal and the security constraints are satisfied.

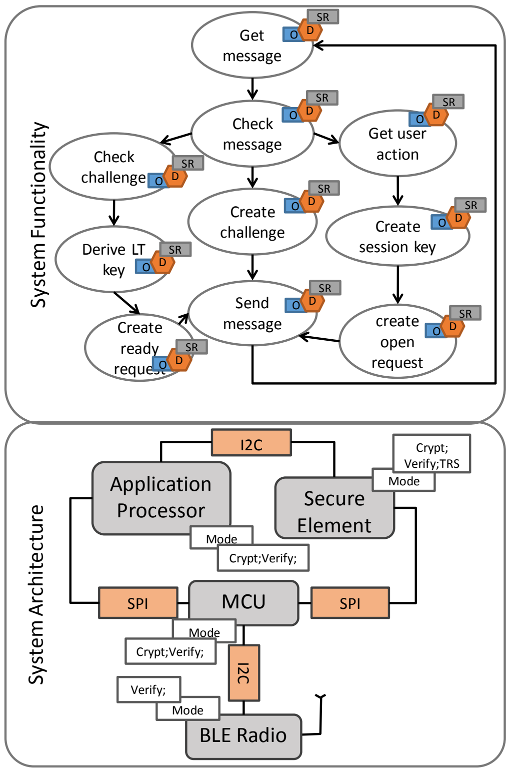


Figure 4. Evaluation example. Keyless entry system's simplified functionality which is to be mapped to a hardware platform. The PEs are connected via bus systems

The tasks of the system's functionality were attributed with the data blocks they are operating on. The attack vectors were attributed to the PEs of the system architecture. The SADSE framework was configured in such a way that the security constraints should be satisfied and the overall system' execution time should be minimized. Two runs with changing data operations on the session key, and a third run without any security constraints were performed. The security capabilities, the attack vectors, and the capability levels of the single hardware blocks are listed in Table I. The security capabilities of the components are based on existing hardware components. The Application Processor's (AP's) capabilities are derived from a *Snapdragon 410E* [13], the Secure Element's (SE's) capabilities from an *P6021* [14], the Micro Controller's (MCU's) capabilities from an *ARM A57* [15], and the BLE Radio's capabilitie's from an *HZX-51822-16N03* [16]. The components security capabilities define the security levels in each mode. E.g. the SE's encryption mechanism AES-256 is assigned a security level of 3, whereas an AES-128 gets a security level 2. A DES-112 is only assigned a security level of 1. For authentication functions, such as MAC and HMAC a similar classification is performed. Table II shows the attributions of the single data entities with security requirements, and their security assurance levels. Table III shows the attributions of the single tasks with data entities and operations.

TABLE I. ATTACK VECTORS AND SECURITY CAPABILITIES

HW	av	Sec_{cap}	m0	m1
AP	(S, T, I_D)	$(enc, verify)$	(1,1)	(2,2)
MCU	(T, I_D)	$(enc, verify)$	(2,2)	(3,3)
SE	(S, T, I_D)	$(enc, verify, sec_{store})$	(2,2,3)	(3,3,3)
BLE	(S, T, I_D)	$(verify)$	(1)	-

TABLE II. DATA BLOCK SECURITY REQUIREMENTS AND ASSURANCE LEVELS

Data Block	sr	$asslvl_{sec}$
$requ^{lock}_{chall}, action_{user}$	-	-
$key_{lt}, key_{master}, key_{session}$	$(Conf, Int)$	$(3, 3, 2)$
$resp^{dev}_{chall}, requ^{dev}_{open}, requ^{dev}_{ready}$	$(Conf, Auth)$	$(2, 2, 2)$
$resp^{lock}_{chall}, resp^{lock}_{ready}$	$(Conf, Auth)$	$(2, 2, 2)$

TABLE III. TASKS AND USED DATA BLOCKS

Task Name	Data Block	Operations
Get message	$resp^{lock}_{chall}, requ^{lock}_{chall}, resp^{lock}_{ready}$	rx, tx
Check message	$resp^{lock}_{chall}, requ^{lock}_{chall}, resp^{lock}_{open}$	rx, r
Create challenge	$resp^{dev}_{chall}$	w, tx
	key_{master}	r
Check challenge	$resp^{lock}_{chall}$	rx, r
	key_{master}	r
Derive LT key	key_{lt}	w, st
	key_{master}	r
Create ready request	key_{lt}	r
	$requ^{dev}_{ready}$	w, tx
Send message	$requ^{dev}_{open}, requ^{dev}_{ready}, resp^{dev}_{chall}$	rx, tx
Get User Action	$action_{user}$	r
Create session key	$key_{session}$	w, st
	key_{lt}	r
Create open request	$key_{session}$	r
	$requ_{open}$	w

The system's functionality and the hardware platform are presented in Figure 4. Table IV shows the full mappings of tasks to hardware components for the distinct runs. The tasks *Get message* and *Send message* have a fixed mapping to *BLE Radio*. As shown in Table IV, the framework was able to correctly map the security critical tasks to the respective hardware components and select the optimal modes regarding the overall performance of the system. To introduce the overhead of the respective security mechanisms of each mode, their computational overhead was derived using the work of [17]. For simplification, the WCETs of each process to PE mapping stays unchanged for the single PE's modes. Thus, the change in the system's performance is only induced by the selection of the security mechanisms.

To demonstrate the effect of the security requirements on the mapping, two runs. In the first execution, the configuration as described in the tables was chosen. In the second run, no security requirements were used. Table IV shows the optimal mappings of tasks to PEs in the respective runs. The PEs are numbered from 0 to 3: AP (0), MCU (1), SE (2), and BLE Radio (3). Each PE offers to possible modes, 0 or 1. The PE and mode mapping is abbreviated with [PE](mode). It can be seen, that the tool was able to correctly allocate *Derive LT key* and *Create session key* to the secure element in run *#1*. In run *#2*, the allocation changes completely, as no security constraints are to be solved. In run *#1*, 45 solutions were found in less than 400ms. In run *#2*, the SADSE framework found 5576 solutions in 27 seconds.

TABLE IV. MAPPING TASKS TO PROCESSING ELEMENTS

Task Name	mapping #1	mapping #2
Get message	`[3](0)`	`[3](0)`
Check message	`[1](0)`	`[1](0)`
Create challenge	`[1](0)`	`[1](0)`
Check challenge	`[1](0)`	`[1](0)`
Derive LT key	`[2](1)`	`[1](0)`
Create ready request	`[1](0)`	`[0](0)`
Get User Action	`[0](1)`	`[0](0)`
Create session key	`[2](1)`	`[2](0)`
Create open request	`[1](0)`	`[3](0)`
Send message	`[3](0)`	`[3](0)`

The keyless entry system example shows the correct functionality of the SADSE framework. It is able to find a valid solution which satisfies both the security constraints and has the fastest execution time. Considering the security constraints leads to a reduced number of found solution, which also speeds up the finding of the optimal solution for the keyless entry example.

V. Conclusion and Future Work

The SADSE framework allows to define security attack vectors and security requirements for system functionalities defined by designers. These security requirements and attack vectors are defined by security experts, following widely used approaches, such as STRIDE analysis or the CIA triad. Based on these requirements and the information about the assumed performance, the security levels, and power consumption of the single tasks executed on distinct hardware platforms, the SADSE framework finds an optimal mapping, under consideration of the security constraints. With this tool, security requirements can be regarded right at the beginning of the design phase. Thus, a greater awareness of security constraints is introduced into the early stages of product design.

Currently, the SADSE framework only regards abstract security levels, considering the capability of the components and the needed security levels of the data entities. These levels are mere placeholders and are to be replaced by real cost factors. To acquire these security costs, a novel method will be developed, helping designers to assess the right level of protection. Furthermore, we want to include distinct security communication protocols, as well as add key distribution mechanisms to the SADSE framework.

Acknowledgment

Project partners are NXP Semiconductor Austria GmbH and the Technical University of Graz. This work was supported by the Austrian Research Promotion Agency (FFG) within the project UBSmart (project number: 859475).

References

[1] K. Rosvall and I. Sander, “A constraint-based design space exploration framework for real-time applications on mpsocs,” in Proceedings of the Conference on Design, Automation & Test in Europe, ser. DATE ’14. 3001 Leuven, Belgium, Belgium: European Design and Automation Association, 2014, pp. 1–6.

[2] K. Rosvall, N. Khalilzad, G. Ungureanu, and I. Sander, “Throughput Propagation in Constraint-Based Design Space Exploration for Mixed-Criticality Systems,” Proceedings of the 9th Workshop on Rapid Simulation and Performance Evaluation: Methods and Tools - RAPIDO ’17, 2017, pp. 1–8.

[3] N. Khalilzad, K. Rosvall, and I. Sander, “A Modular Design Space Exploration Framework for Embedded Systems,” IEEE Proc. Computers & Digital Techniques, vol. 152, 2005, pp. 183–192.

[4] B. Knerr, “Heuristic Optimisation Methods for System Partitioning in HW / SW Co-Design,” Ph.D. dissertation, Vienna University of Technology, 2008.

[5] E. Kang, “Design Space Exploration for Security,” no. April 2008, 2016, pp. 1–4.

[6] I. Stierand, S. Malipatlolla, S. Froschle, A. Stuhring, and S. Henkler, “Integrating the security aspect into design space exploration of embedded systems,” Proceedings - IEEE 25th International Symposium on Software Reliability Engineering Workshops, ISSREW 2014, 2014, pp. 371–376.

[7] M. Hasan, S. Mohan, R. Pellizzoni, and R. B. Bobba, “A design-space exploration for allocating security tasks in multicore real-Time systems,” Proceedings of the 2018 Design, Automation and Test in Europe Conference and Exhibition, DATE 2018, vol. 2018-Janua, 2018, pp. 225–230.

[8] M. Farooq, M. Waseem, A. Khairi, and S. Mazhar, “A Critical Analysis on the Security Concerns of Internet of Things (IoT),” International Journal of Computer Applications, vol. 111, no. 7, 2015, pp. 1–6.

[9] S. Hernan, S. Lambert, T. Ostwald, and A. Shostack, “Threat modeling-uncover security design flaws using the stride approach,” MSDN Magazine-Louisville, 2006, pp. 68–75.

[10] P. Baptiste, C. Le Pape, and W. Nuijten, Constraint-based scheduling: applying constraint programming to scheduling problems. Springer Science & Business Media, 2012, vol. 39.

[11] J. Xu and et al., “Pairing and authentication security technologies in low-power bluetooth,” Proceedings - 2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing, GreenCom-iThings-CPSCom 2013, 2013, pp. 1081–1085.

[12] H. Oguma, N. Nobata, K. Nawa, T. Mizota, and M. Shinagawa, “Passive keyless entry system for long term operation,” 2011 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, WoWMoM 2011 - Digital Proceedings, 2011, pp. 1–3.

[13] “ARM Cortex®-A57 MPCore ProcessorCryptography Extension Technical Reference Manual,” ARM Limited, Tech. Rep.

[14] “BSI-DSZ-CC-1072-2018 for NXP Secure Smart Card Controller P6021y VB *,” 2018.

[15] “Qualcomm Snapdragon 410E Processor(APQ8016E) Technical Reference Manual,” Qualcomm Technologies, Inc., Tech. Rep.

[16] Shen Zhen Huazhixin Technology Ltd, “HZX-51822-16N03 Bluetooth 4.0 Low Energy Module Datasheet,” Tech. Rep., 2017.

[17] A.-K. Al Tamimi, “Performance Analysis of Data Encryption Algorithms.”

2019 22nd Euromicro Conference on Digital System Design (DSD)

Consideration of Security Attacks in the Design Space Exploration of Embedded Systems

Lukas Gressl
Institute of Technical Informatics
Graz University of Technology
Graz, Austria
email: gressl@tugraz.at

Christian Steger
Institute of Technical Informatics
Graz University of Technology
Graz, Austria
email: steger@tugraz.at

Ulrich Neffe
NXP Semiconductors Austria GmbH
Gratkorn, Austria
email: ulrich.neffe@nxp.com

***Abstract*—Designing secure systems is a complex task, particularly for designers who are no security experts. Cyber security plays a key role in embedded systems, especially for the domain of the Internet of Things (IoT). IoT systems of this kind are becoming increasingly important in daily life as they simplify various tasks. They are usually small, either embedded into bigger systems or battery driven, and perform monitoring or one shot tasks. Thus, they are subject to manifold constraints in terms of performance, power consumption, chip area, etc. As they are continuously connected to the internet and utilize our private data to perform their tasks, they are interesting for potential attackers. Cyber security thus plays an important role for the design of an IoT system. As the usage of security measures usually increases both computation time, as well as power consumption, a conflict between these constraints must be solved. For the designers of such systems, balancing these constraints constitutes a highly complex task. In this paper we propose a novel approach for considering possible security attacks on embedded systems, simplifying the consideration of security requirements immediately at the start of the design process. We introduce a security aware design space exploration framework which based on an architectural, behavioral and security attack description, finds the optimal design for IoT systems. We also demonstrate the feasibility and the benefits of our framework based on a door access system use case.**

***Index Terms*—Cyber Security; Embedded System Design; Secure IoT Systems; Mixed Criticality Design Space Exploration;**

I. Introduction

Information security for networked systems has been an important issue for both industry and research throughout the last decades. In recent years, large scale security attacks on governmental, military, and industrial targets were reported to have been successfully carried out [1]. These publicly reported incidents can be assumed to form the tip of the iceberg.

Internet of Things (IoT) systems in the private sector are attack targets on a much more frequent basis. Their requirements of high performance, low power consumption and a small form factor make the integration of security features a challenging problem for system designers [2]. Recent incidents with the access system built into cars emphasize the need for security aware system design [1]. Finding an optimal solution for this design problem is a task too complex for manual solution, when considering all possible system components and security features. Design space exploration (DSE) tools are a suitable way to support the designers in finding an optimal design solution. These tools take as input a description of the system's behavior, the system's architecture, and application and architectural characteristics to find an optimal solution for task mapping and architecture selection. State-of-the-art DSE tools support the formulation of constraints on performance and power consumption [3]. Other DSE tools integrate security or safety requirements on an abstract basis [4].

[1]https://www.esat.kuleuven.be/cosic/fast-furious-and-insecure-passive-keyless-entry-and-start-in-modern-supercars/

To the best of our knowledge, we are presenting a novel approach for the integration of security requirements into the DSE. We introduce the Security Attack aware Design Space Exploration (SAaDSE) framework. With the SAaDSE framework we offer a system designer a way to describe the system from three distinct perspectives: the functional, the architectural, and the security attack view. Based on these descriptions, the framework finds the optimal process mapping and component selection for cyber security, performance or power consumption. Figure 1 shows the basic overview of the SAaDSE framework.

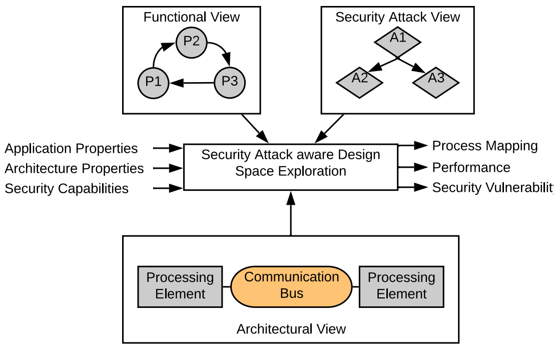


Fig. 1: System design space exploration framework overview.

We are making the following contributions with the approach described in this paper:

- To the best of our knowledge, we are the first to combine security attack graphs and the DSE approach. Thus, we

offer a framework, in which the impact of security attacks on the system changes the system's requirements.
- We show that the computational overhead of the process to component mapping regarding the security attack probability is feasible and can be optimized.
- We describe the system's functionality using processes, operations, and data entities, offering a finer grained model, avoiding ambiguous task descriptions.

The paper is structured as follows: in Section II, contributions on DSE and security attack modeling are described; in Section III our proposed method is shown; in Section IV the implementation details are outlined; Section V describes a use case, showing our framework's feasibility; Section VI concludes this paper and gives an outlook on future work.

II. Related Work

Cyber-security risk assessment and management is a field of research which has attracted much attention in the past decade. Its application ranges from supporting network system administrators to choose optimal measures for securing their networks to helping business managers to take appropriate actions for reducing the security risk affecting their infrastructure. Karabacak et al. [5] describe a method to calculate the risk of specific security problems in a company's information system based on surveys. They focus on simplifying the survey process to enable non security experts to participate.

Feng et al. [6] and Poolsappasit et al. [7] presented work which not only considers survey based cyber security risk analysis, but also uses attack trees to represent possible security exploits. Poolsappasit et al. [7] consider dynamic cyber-security risk of network systems by analyzing possible attacks using attack trees. By assigning each node of the attack tree with a respective probability of success, the overall security risk at the root of the attack tree can be computed by representing the attack tree as a Bayesian network. The authors also consider the impact of cyber security mitigation techniques on the overall security risk to the system. Feng et al. [6] propose a technique, which is also based on Bayesian networks for calculating cyber security risk. In their work, the authors describe a security risk management tool which takes into account historical security incidents as well as security expert judgment. Based on these inputs they formulate a risk analysis method. The authors do not consider techniques to mitigate security vulnerabilities, or decrease the security risk.

DSE for embedded devices is a well described problem in the literature. Classical DSE tools consider the power and performance characteristics of feasible mappings of tasks to software or hardware solutions, always aiming at optimizing the overall system performance, costs, or power consumption [3], [8]–[10]. In addition to these classical optimization goals, extended work introduced aspects such as functional safety [11], or cyber security [4], [12]–[14] as further constraints to be fulfilled by the system that is being designed.

Xie et al. [12] optimize the data throughput of a *Controller Area Network* bus taking possible attacks on the communication into consideration. They secure the communication by adding message authentication codes (MACs) to messages vulnerable to manipulation. The additional overhead induced by these MACs influences the communication delay and system performance. The authors consider the influence of MACs on the performance constraint and describe a method to simultaneously optimize the system's timing behavior and secure the communication. Hasan et al. [13] deal with the integration of security aware tasks into an already existing task schedule on a multicore-system. They describe a solution with which tasks performing security checks can be added to a predefined schedule without breaking real-time constraints. Kang [14] published a tool which supports system designers on their decision-making concerning, what security mitigation techniques should be used for a network system. Stierand et al. [4] introduce security parameters into automatic DSE for mapping tasks to processors with and without a hardware security extension. Their focus lies on the communication between the different types of processors and assessing the attack vulnerability of the channels connecting them. In their method, the attacker's capability determines the system's vulnerability. Mitigating these attacks, the authors propose to assign such vulnerable tasks to processor types which provide hardware security extensions only.

The cyber security aware DSE solutions presented so far either concentrate on very specific parts of the overall system, providing solutions for detailed problems at hand [12]–[14], or abstract the system in a way that allows an holistic view on the system design, but strongly simplifies the security requirements [4]. The SAaDSE framework presented in this paper adds security relevant details, without losing the holistic view on the overall system under design, unlike [12]–[14], and without abstracting security details as in [4]. The methodology builds on the framework presented in [15], introducing security attacks aiming at system components. Therefore, we propose a framework offering an integration of security attack graphs and mitigation techniques into the early stages of system design. By considering not only the security vulnerabilities of the individual tasks, but also the attack mitigation techniques of system components, the framework supports designers with the possibility of constructing secure systems at an early stage.

III. System and Security Attack Representation

The system representation is split into three distinct views: the system functionality view, the system architecture view and the system security attack view. The distinct views allow the system designer to consider the system under design from different perspectives and they serve as inputs to the design space exploration framework described in this paper. The single views and their components are described in this section.

A. System Representation

Describing cyber security as an additional constraint for the design space exploration of a device in the course of design makes a more detailed description of the system's behavior and architecture necessary. Before considering security features within an automated task allocation and system partitioning, the representation of the system's behavior and the system's architecture, building on [15], is introduced.

1) System Behavior View: The functionality of the system is described by a process graph, as described in [9]. To achieve a more distinct description of the behavior of the individual tasks and to avoid ambiguous interpretations of a task's security constraints, we propose to extend the task's description using data entities and connecting them with a range of operations. An example process graph and the partitioning of the process into tasks, operations, and data entities is depicted in Figure 2. The task's operations comprise: reading, writing, transmitting, receiving and storing the data entity. For better distinction we denote a process p as a task connected to a set of data entities using a set of operations. This split can be seen in Figure 2.

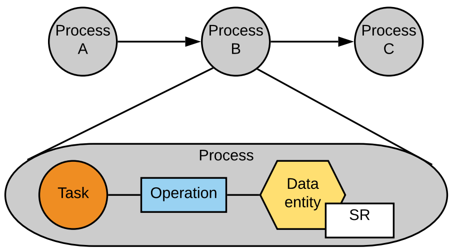


Fig. 2: Process graph represented by three processes. Each process is consists of a task linked to a data entity using a set of operations. Each data entity is attributed with security requirements.

2) System Architecture View: The system architecture is described by a set of processing elements (PEs) and communication bus systems connecting them. Each PE is described with a set of characteristics (chip area, memory size, etc.) and a set of modes. Each mode is described with its static and dynamic power consumption, a set of security capabilities and the corresponding security mitigation factors. The security capabilities describe the PE's capabilities on encryption, authentication, secure storage, etc., and may induce additional power consumption and performance overhead. The security mitigation factor describes the PE's susceptibility to cyber-security attacks. The performance of a process mapped onto a PE running in a distinct mode is described by its worst case execution time (WCET). The WCETs of the mapped processes also define the PE's dynamic power consumption. Figure 3 shows the representation model of two PEs. The details of the mapping process are described in Section III-B.

3) Security Attack View: For modeling the security threats a combination of security attack graphs [16] and Bayesian networks [17] is used, commonly known under the term Bayesian network based attack graphs (BNAGs). BNAGs are a widely used approach to model attack scenarios on network systems [7], [18]–[20]. An attack A is represented by a distinct node in the BNAG. Furthermore, this attack A is described by a state A_{state} which can either be success ($S_{success}$) or no-success ($S_{no-success}$). The successful execution of an attack, denoted by the Bernoulli random variable at, is described

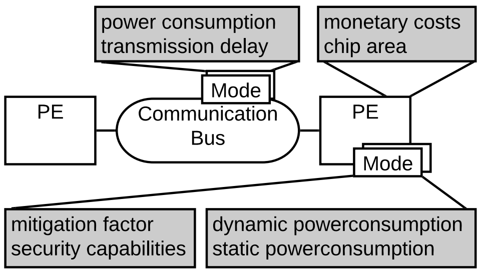


Fig. 3: Two PEs connected by a communication bus. The system components are characterized by mode dependent and independent parameters.

by the state transition $S_{no-success} \rightarrow S_{success}$. This successful execution can be accomplished by the attacker with the probability $P(at)$. These probabilities reflect the security risk of successfully executing said attack and can be assessed following techniques described in [21]. Within the BNAG the $P(at)$ of a certain attack might depend on the states of the preceding attacks. This dependence is encoded into the attack's conditional distribution table (CDT) and is depicted in Figure 6. The joint probability of the variables in the BNAG is calculated using the chain rule 1, where $Pa[at_i]$ denotes the conditional probability of the parent of attack node i. The unconditional success probabilities of each attack are obtained by merging the marginal cases of each attack node.

$$P(at_1, ..., at_n) = \prod_{i=1}^{n} P(at_i|Pa[at_i]) \tag{1}$$

The SAaDSE framework integrates the security attacks represented by the BNAGs into the system behavior and architecture view. This integration assigns each security attack to a distinct process in the process graph of the functional view. This attribution between attacks and processes is depicted in Figure 4.

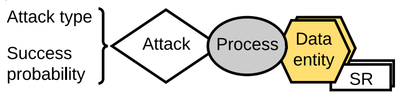


Fig. 4: Security attack on a specific task. The task operates on a set of data entities.

The single attacks are organized in BNAGs to describe system security attacks, which are described by using distinct attack scenarios. Figure 5 depicts such a system security attack consisting of two separate attack scenarios. Each scenario consists of one or more security attacks, forming a Bayesian network. Each scenario aims at successfully reaching the attack goal situated at the end of the network. The security attack goals can further be used to identify the security vulnerability of the overall system. The system's vulnerability can be limited by assigning thresholds to the individual security goals. The scenarios are used to cluster the security attacks for easier evaluation by security experts. Attacks are not limited to being only dependent on attacks in the same scenario. This is depicted by the dependency of *attack 6* on *attack 1* of *Scenario 1*. Figure 6 shows the CDTs of the single security attacks in *Scenario 1*. To show how the security attack probability propagates through the network, the unconditional success probabilities of each security attack are depicted as $P(Ai)$ in Figure 6.

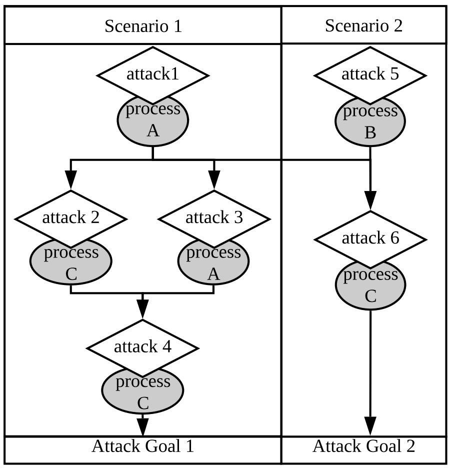


Fig. 5: Example of activity threads describing attack scenarios with two distinct attack goals. Activity dependency is not limited by scenario boundaries.

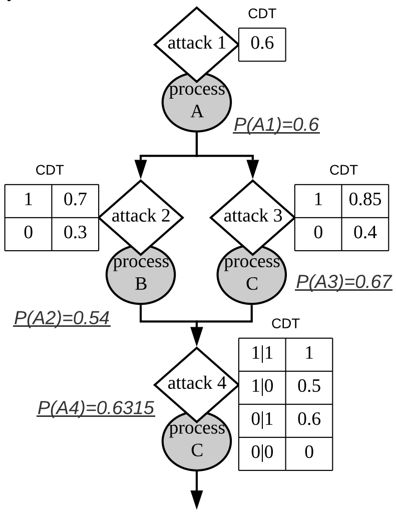


Fig. 6: Detailed view on *attack scenario 1* of 5. The CDT's of the individual attacks, as well as their unconditional probabilities are shown.

The success probabilities of the single attacks in the BNAG are evaluated based on expert judgments. Although these expert judgments are not in the focus of this paper, two assumptions have to be made. First, each attack success probability considers the attacker's capability and motivation to perform the distinct attack. The motivation must take into account the values of the data entities the process operates on. Second, the expert must judge the success probability on the distinct processes as if they are executed on a hardware platforms not offering any security mitigation techniques.

To support the system designers in the process of designing the product, the SAaDSE framework uses the provided descriptions on system functionality, architecture and security attacks to find the optimal choice on PE's, their modes, and process allocation. Thereby, it considers the overall system's performance, power consumption and information security. The details on the design space exploration with focus on the security constraints are given in III-B.

B. Integration of the BNAG into Design Space Exploration

The SAaDSE framework's goal is to find an optimal solution for a system in the course of design, investigating different options for architectural blocks and process allocation. To find this solution, it is given the system's functionality, architecture, and security attack scenarios. The framework finds the optimal solution for either the performance, power, or security, or all feasible solutions. This section explains the basic mapping process of processes to PE's and its influence on the system's performance. Furthermore, the calculation of the feasibility for the security attack scenarios is explained in detail.

1) Mapping process: The mappings of the processes to the PEs are performed by using the WCETs of each process executed on a PE which runs in a distinct mode. These WCETs are measured or estimated by the system designers. Additionally, the mapping of a process to a PE is limited by the mapping of its parent processes. A process can only be mapped to the same PE as its parent processes, or on a PE which is connected to the PE running its parent processes. This connection can be provided by a communication bus, or some other means of physical connection. Furthermore, the number of processes executed on a PE(s) is limited by its memory capacity, which must not be exceeded. These basic physical constraints on the process allocation is further limited by the security attacks, aiming at the process and the security capability the PE offers. These security constraints are explained in Section III-B2.

2) Security constraints calculation: The security constraints on the overall system are calculated using the security attack goals of the single attack scenarios, the security requirements of the data entities handled by the processes, and the security capabilities of the PEs. For each process operating on a set of data entities, the security operations the process must be able to perform on the data entity are calculated. The set of operations which can be executed on a data entity by a process are reading (r), writing (w), transmitting (tx), receiving (rx) and storing (st) the data entity. This set is denoted by the tuple $o = (r, w, rx, tx, st)$. From this set and the security requirements (confidentiality ($conf$), authenticity ($auth$), integrity (int)), represented by the tuple $sr = (conf, auth, int)$, the set of secure operations (encryption (enc), signature ($sign$), secure storage (st_{sec})) $o_{sec} = (enc, sign, st_{sec})^{\mathsf{T}}$ is calculated according to (2). The values of both operations and security requirements can either be 0 or 1.

$$o_{sec}(o, sr) = \begin{pmatrix} (r \vee w) \wedge conf \\ (r \vee w) \wedge auth \\ st \wedge (auth \vee conf \vee int) \end{pmatrix} \tag{2}$$

Using the secure operations of the individual processes, the mapping of processes to PEs is constrained in a security supportive way. This means that a process is only feasible to be mapped to a PE which runs in a mode supporting the secure operations. This secure operation support depends on the security capabilities (sc) offered by the PE's mode, denoted as the tuple $sec_{cap} = (sc_{enc}, sc_{auth}, sc_{tss})$, standing for encryption (enc), authentication ($auth$) and tamper-safe storage (tss). This security operation to capability mapping is defined in (3). The mapping operation must be performed for all possible mappings.

Furthermore, the possible processes mappings are limited by the security goals of the attack scenarios. The calculation of the security goals is performed by applying the Bayesian chain rule (1) on the the attack nodes of the BNAG and merging the marginals for each attack goal. However, the mapping of the processes to the PEs influences the success probability of the attack nodes, aiming at the specific processes and, thus, changes the success probability of the security goals. More specifically, for each process to PE mapping the conditional success probabilities (an_p) of the attacks aiming at this process are reduced by the mitigation factor of the mode in which said PE is running. The mitigation factor is expressed by $m_{fact} \in \mathbb{Q} : m_{fact} \in [0, 1]$. As the mitigation factor reduces the security attack success probability, the reduction factor by which this success probability is reduced is the complementary probability of m_{fact}. This reduction of the attack node's conditional success probability is only applicable if the node's attack type (an_{at}) can be mitigated by the PE's mode. The reduction of the attack node's conditional probabilities (red_{an_p}) based on PE's mitigation factors is performed according to (4).

$$map(o_{sec}, sec_{cap}) = \bigcap_{i=1}^{n} o_{sec}[i] \wedge sec_{cap}[i] \tag{3}$$

$$red_{an_p} = \begin{cases} an_p * (1 - m_{fact}), & \text{if } an_{at} \wedge sec_{cap} \\ an_p, & \text{otherwise} \end{cases} \tag{4}$$

Using $map(o_{sec}, sec_{cap})$ and the security goal calculation, feasible and infeasible process to PE mappings are defined. Process mappings not supporting all necessary security operations, or by which at least one security goal exceeds the defined threshold are rendered infeasible.

As mentioned earlier, the assignment of the security capabilities' mitigation factors must be performed by the security experts. An accredited reference point for this rating can be found in the Common Criteria (CC) certification process. The CC certification provides a standardized way of assessing the security features of security products. The CC certification defines evaluation assurance levels (EALs), which state to what extent the security capabilities of the specific target of evaluation have been tested. Seven EALs are specified, ranging from simple functional testing (EAL1) to a formal verification and testing (EAL7). Additionally, each product must match a defined protection profile ensuring that it provides all expected security features [22]. A method on how to derive mitigation factors from said EALs is to be described in future work. A list of CC certified products and their certification documentations can be publicly found at the CC's official web page [2]. The implementation of the security mapping constraints is explained in more detail in Section IV.

IV. Implementation

The implementation of the SAaDSE framework is based on the *DeSyDe* framework which is publicly available at *Github*[3]. The *DeSyDe* framework finds an optimal mapping from processes to PEs and selection of process modes considering power consumption and performance. Rosvall et al. published two of papers discussing in detail how the scheduling, performance, and power consumption was calculated [3], [8]. The following extensions were introduced for the implementation of the SAaDSE framework.

Task data handling: The task representation was changed by adding the definition of data entities and operations to the tasks, forming the models of processes. As several processes can work on the same instance of a data entity by passing them to one another, the communication channels between the processes were adapted with the information of the passed data entities as well. The message size is further used for calculating the communication overhead.

Communication bus connections: In contrast to the *DeSyDe* framework the SAaDSE framework models the communication between PEs using communication bus connections. The usage of these physical connections induces two changes. First, the mapping of processes is limited in a way that only PEs physically connected with each other are allowed to run processes, having a direct communication channel. Second, the communication via the communication bus affects the overall performance of the system. The transmission delay ($delay_{tx} = conn_{ef} * conn_{speed} * data_{size}$) of the communication between two processes over a communication bus is calculated by using the bus' encoding factor ($conn_{ef}$), transmission speed ($conn_{speed}$), and the exchanged data size ($data_{size}$). What data entity is exchanged between two processes is determined by the processes' operations (rx, tx). This additional transmission delay is used for calculating the worst case communication time, which propagates into the overall system performance, as described in [3].

[2]https://www.commoncriteriaportal.org/
[3]https://github.com/forsyde/DeSyDe

Security constraints: As discussed in section III-B2, the calculation of the security constraints builds upon the knowledge of the security capabilities, the security requirements, and the security goals success probabilities. The mapping between the security capabilities and process' security operations (3) is realized in a straight forward check, what security operations are satisfied by a PE's security capability. This check induces a computational overhead of $O(N_P^{N_{PE_M}})$, with N_P being the number of processes in the application graph, and N_{PE_M} being the number of all modes of all PEs. However, as the calculation of the security goals is based on the BNAG, the calculation of these goals is a computationally hard problem. Already in 1990, G. F. Cooper showed that the exact probabilistic inference in Bayesian network is an NP hard problem [23]. This problem is further aggravated by the fact that each process (p) to PE (pe) mapping influences the attack nodes' CDTs. Thereby, the recalculation of the security goals is necessary for each mapping. The NP hardness of the exact probabilistic inference can be reduced by using approximations instead. Such approximate inferences can be calculated using samplers, e.g. the likelihood weighting sampler [24].

For reducing the recalculation of the security goals induced by the different mapping possibilities, the SAaDSE framework first finds all processes, which attacks influence the security goals. All other processes are disregarded. Then, the SAaDSE framework orders all PEs according to the mitigation factors of their respective modes. The PE mode's mitigation factors are denoted by the function $\theta(pe(m))$. This ordered set of PE mode mitigation factors $\Theta = (\theta_1(pe_x(m_y)) \geq, ..., \geq \theta_n(pe_{x'}(m_{y'}))$ is then used for calculating the security goals for each possible mapping of p to $pe(m)$, denoted $m(p, pe(m))$. It must be noticed that not all permutations are feasible, as a system mapping $m(p_x, pe_a(m_b))$ cannot map $m(p_y, pe_a(m_c))$. These infeasible combinations are eliminated up front. After this elimination, the framework permutes all feasible m using Θ and calculates the security goals for each permutation. Each permutation is described as $P = (m_1(p_1, pe_a(m_b)), ..., m_{N_P}(p_{N_P}, pe_{N_{PE}}(m_{N_m})))$, with N_{PE} being the number of all pe and N_m the number of modes of the current pe. As Θ is ordered descending, the security goal calculation can be halted when a P_{insec}, which does not satisfy at least one security goal threshold, is found. All following P can be rendered insecure without further security goal calculation, until a P' is reached, in which at least one $m'(p_x, pe_{a'}(m_{b'}))$ has a $\theta'(pe_{a'}(m_{b'}))$ greater than $\theta(pe_a(m_b))$ of $m(p_x, pe_a(m_b))$ in P_{insec}. After finding P' the algorithm continues calculating the security goals for all following P until reaching the next P_{insec}.The algorithm's computational worst case is not finding any mapping dissatisfying the security goals' thresholds. In all other cases the algorithm will reduce the number of necessary inference calculations in the BNAG.

V. Experiments and Results

The functionality of the SAaDSE framework is shown by designing a keyless entry system which consists of a lock and a key fob. For the evaluation of the framework only the design of the lock is presented. This design takes into consideration following simplified security attack scenarios: spoofing the authentication of the key fob, breaking the lock's cryptography by leaking the used key material, and hijacking the session between lock and key fob.

The authentication between lock and key fob is based on a challenge ($chall$) response ($resp$) - request ($requ$) exchange. This authentication is performed by using a pre-shared secret in form of a master key (mk). After authenticating the key fob against the log, both devices are bound to each other by using a binding key, denoted as long term key (ltk). Reducing the time frame in which an attack can compromise the lock's access control, the final open request is based on the session key (sk), which is derived from ltk and renewed periodically. Figure 7 depicts a simplified representation of the lock's behavior using an application graph, comprising the authentication, binding, and access process.

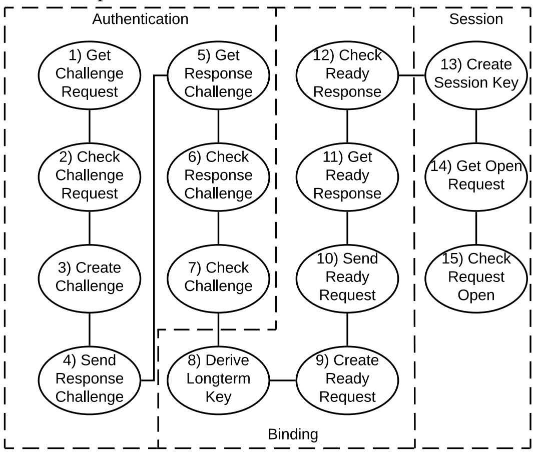


Fig. 7: Task graph depicting the key fob's functionality, split into authentication, binding, and session steps.

In table I the security relevant tasks, the data blocks they are using and the operations they perform on these data entities are listed. Security relevant means, that a possible attacker could aim the attacks on the process' functionality or the used data.

TABLE I: SECURITY RELEVANT TASK AND DATA

Task Name	Data Entities	Operations
Check Chall Requ	$requ^{fob}(chall)$	rx, r
Create challenge	$resp^{lock}(chall)$	w, tx
	mk	r
Check challenge	$resp^{fob}(chall)$	rx, r
	mk	r
Derive LT key	ltk	w, st
	mk	r
Create ready request	ltk	r
	$requ^{lock}(ready)$	w, tx
Create session key	sk	w
	ltk	r
Check open request	sk	r
	$requ^{fob}(open)$	w

The following abstract components were considered for the hardware platform of the lock: for communication to the key fob the lock is equipped with a BLE radio; for storing the cryptographic key material the lock uses an integrated secure element (SE); additionally, the lock might use a microcontroller (MCU) which offers cryptographic functionality using an integrated co-processor (CryptCP) or software based cryptography (SWCrypt), but lacking the capability to store data in a tamper safe manner. The cryptographic co-processor for the mcu is assumed to use a tripple DES algorithm with bit security of 168 bits. The components are connected to each other using dedicated communication bus systems. Table II lists the possible components, which can be selected by the SAaDSE framework for the hardware platform of the lock, their mitigation factors, their security capabilities, and the estimated WCETs (in ms) of the security relevant tasks running on these components. The WCETs also consider the cryptographic overheads for encryption and authentication. These timings are abstracted and serve the purpose of showing the framework's correct mapping and component selection. The mitigation factors were judged by security experts based on the EALs assigned by CC certifications. A methodology on how to derive these mitigation factors must be described in future work. The security capabilities (SC) are abbreviated with e (encryption/decryption), s (signing/verifying), and t (tamper safe storing).

TABLE II: PROCESSING ELEMENTS AND WCETS

PE	MF	SC	p2	p3	p6	p8	p11	p13
SE_1	0.75 (EAL4+)	e, s, t	100	130	120	180	180	100
SE_2	0.85 (EAL5+)	e, s, t	120	150	140	200	200	120
SE_3	0.95 (EAL6+)	e, s, t	150	180	160	220	220	150
MCU_1	0.7 (CryptCP)	e, s	80	100	110	200	200	80
MCU_2	0.2 (SWCrypt)	e, s	200	220	200	300	300	200

Three attack scenarios demonstrate the security aware mapping process represented by Figure 8. The Figure shows both unconditional probabilities, as well as the security goal thresholds. The attacker is assumed to be a technically experienced person, who is able to sniff the BLE channel and to install malicious firmware into the lock using update processes. However, the attacker has no physical access to the lock's hardware. The first attack aims at spoofing the key fob's identity, tricking the lock into a binding with a malicious device. The attacker might spy the challenge request-response exchange, tamper (tamp) or manipulate (man) the request when received by the lock or even try to disclose (discl) the master key. The second attack aims at breaking the lock's cryptography by reading out or even manipulating both the long term and master key. The third attack aims at hijacking the session and, thus, gaining a one time access to the lock, by intercepting and forging the open request, or by reading out or even manipulating the session key via the lock's firmware.

Using the described system's functionality, architectural possibilities and the security attack scenarios, the SAaDSE framework identified 798 solutions to be secure. By comparison, without the security constraints, 3830 solutions were found. Hence, the solution space was already reduced by 79% in this simple example. The usecase was executed on a computer with 16 GiB of RAM and an Intel® Core™ i7-4600U CPU with 2.10 GHz. When using the full calculation of all possible mappings without sorting the PEs' modes according to the mitigation factors, the first solution was found after 1252 seconds, with the termination criterion after 561 seconds. Hence, a speed up of 44.8% could be achieved. Figure 9 shows the most secure system hardware architecture and process mapping, as well as the solution with the highest performance, which is still able to mitigate the security attack scenarios such that no security thresholds are exceeded. It can be noticed that the most secure solution only makes use of SE_3 directly connected to the BLE radio, without using the MCU at all. The best performing solution utilizes MCU_1 in combination with SE_3. MCU_2 is not considered in any solution found by the SAaDSE framework.

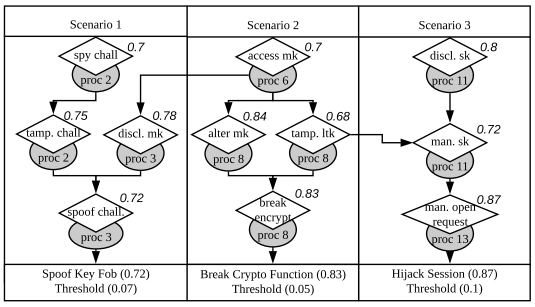


Fig. 8: Excerpt of security attack scenarios aiming at the key fob's functionality.

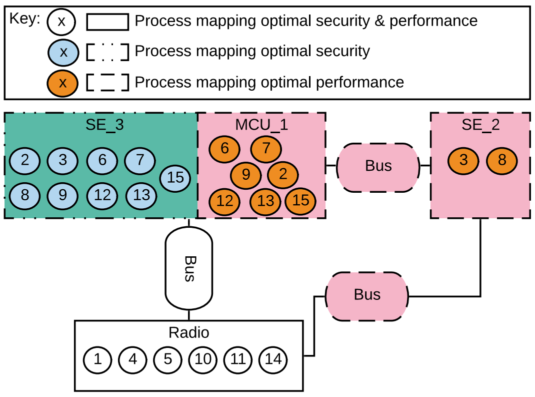

Key:
Process mapping optimal security & performance
Process mapping optimal security
Process mapping optimal performance
SE_3
2 3 6 7 8 9 12 13 15
MCU_1
6 7 9 2 12 13 15
Bus
SE_2
3 8
Bus
Bus
Radio
1 4 5 10 11 14

Fig. 9: Proposed system architecture and process allocation for optimal security and optimal performance.

This usecase describes how the framework is used in a design process to find the most secure product and the product with the best performance, which still fulfills the security constraints defined by the designers.

VI. Conclusion and Future Work

In this paper we presented a novel approach for introducing security attack scenarios into the system design process of embedded systems. The SAaDSE framework that we introduce allows a designer to specify the system's functionality, the possible hardware architecture components running the defined functionality, and to model attack scenarios used for breaking the designed system. The framework uses these descriptions to suggest an optimal hardware component selection and task to component mapping. Section V shows the framework's feasibility for finding not only the most secure, but also the e.g. best performing and still secure solution was shown.

The framework presented in this work can be seen as a first important step for introducing detailed security requirements into the design space exploration of embedded systems. Taking related work into consideration, it can be seen here that security constraints can be formalized more accurately. Using the SAaDSE framework will simplify the design process of secure embedded systems and emphasize the importance of cyber security in the system design process. In this sense, the framework is not meant to replace the work of security experts in designing secure system, but rather utilizes their knowledge. Thus, the framework supports system designers not experienced in cyber security during the initial phases of system design. However, this security expert knowledge must also be seen as a limiting factor, as the mitigation factors of the distinct hardware components must be rated a priori. The CC certification provides guidance concerning the security level of secure devices. However, this guidance is not of direct utility for judging to what extent security attacks can be mitigated. Hence, a central repository storing this mitigation information for secure devices would be a major step for supporting the design process of secure embedded systems.

For future work, the SAaDSE framework will be extended to also suggest appropriate security protocols, including the automatic suggestion of key material and the keys' placement on the system. Furthermore, the derivation for security constraints can be extended from using security attack graphs to full security risk estimation methods. Such an extension would add critical and important information to the DSE and allow the framework to consider additional constraints, such as the financial losses caused by potential security attacks.

Acknowledgment

Project partners are NXP Semiconductors Austria GmbH and the Technical University of Graz. This work was supported by the Austrian Research Promotion Agency (FFG) within the project UBSmart (project number: 859475).

References

[1] Mohammed Nasser, Rabiah Ahmad, Warusia Yassin, Aslinda Hassan, Zaheera Zainal, Nabeel Salih, and Karrar Hameed. Cyber-Security Incidents: A Review Cases in Cyber-Physical Systems. *International Journal of Advanced Computer Science and Applications*, 9(1), 2018.

[2] Ruozhou Yu, Guoliang Xue, Vishnu Teja Kilari, and Xiang Zhang. Deploying robust security in internet of things. *2018 IEEE Conference on Communications and Network Security, CNS 2018*, 2018.

[3] Kathrin Rosvall, Nima Khalilzad, George Ungureanu, and Ingo Sander. Throughput Propagation in Constraint-Based Design Space Exploration for Mixed-Criticality Systems. *Proceedings of the 9th Workshop on Rapid Simulation and Performance Evaluation: Methods and Tools - RAPIDO '17*, 2017.

[4] Ingo Stierand, Sunil Malipatlolla, Sibylle Froschle, Alexander Stuhring, and Stefan Henkler. Integrating the security aspect into design space exploration of embedded systems. *Proceedings - IEEE 25th International Symposium on Software Reliability Engineering Workshops, ISSREW 2014*, 2014.

[5] Bilge Karabacak and Ibrahim Sogukpinar. ISRAM: Information security risk analysis method. *Computers and Security*, 24(2), 2005.

[6] Nan Feng, Harry Jiannan Wang, and Minqiang Li. A security risk analysis model for information systems: Causal relationships of risk factors and vulnerability propagation analysis. *Information Sciences*, 256, 2014.

[7] Nayot Poolsappasit, Rinku Dewri, and Indrajit Ray. Dynamic Security Risk Management Using Bayesian Attack Graphs. *IEEE Transactions on Dependable and Secure Computing*, 9(1), jan 2012.

[8] Kathrin Rosvall and Ingo Sander. A Constraint-based Design Space Exploration Framework for Real-time Applications on MPSoCs. *Proceedings of the Conference on Design, Automation & Test in Europe*, 2014.

[9] Bastian Knerr. *Heuristic Optimisation Methods for System Partitioning in HW / SW Co-Design*. PhD thesis, Vienna University of Technology, 2008.

[10] Kathrin Rosvall, Tage Mohammadat, George Ungureanu, Johnny Oberg, and Ingo Sander. Exploring power and throughput for dataflow applications on predictable NoC multiprocessors. *Proceedings - 21st Euromicro Conference on Digital System Design, DSD 2018*, 2018.

[11] Domiian Tma-Selicean and Paul Pop. Design Optimization of Mixed-Criticality Real-Time Embedded Systems. *ACM Transactions on Embedded Computing Systems*, 14(3), 2015.

[12] Yong Xie, Gang Zeng, Ryo Kurachi, Hiroaki Takada, and Guoqi Xie. Security/Timing-aware Design Space Exploration of CAN FD for Automotive Cyber-Physical Systems. *IEEE Transactions on Industrial Informatics*, PP(8), 2018.

[13] Monowar Hasan, Sibin Mohan, Rodolfo Pellizzoni, and Rakesh B. Bobba. A design-space exploration for allocating security tasks in multicore real-Time systems. *Proceedings of the 2018 Design, Automation and Test in Europe Conference and Exhibition, DATE 2018*, 2018-Janua, 2018.

[14] Eunusk Kang. Design Space Exploration for Security. *IEEE Cybersecurity Development Design*, 2016.

[15] Lukas Gressl, Christian Steger, and Ulrich Neffe. A Security Aware Design Space Exploration Framework. In *Proceedings of the Fourteenth International Conference on Systems ICONS 2019*, Valencia, Spain, 2019. ThinkMind(TM) Digital Library.

[16] Vivek Shandilya, Chris B. Simmons, and Sajjan Shiva. Use of attack graphs in security systems. *Journal of Computer Networks and Communications*, 2014, 2014.

[17] David Heckerman and John S. Breese. Causal independence for probability assessment and inference using Bayesian networks. *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans.*, 26(6), 1996.

[18] Erik Miehling, Mohammad Rasouli, and Demosthenis Teneketzis. Optimal defense policies for partially observable spreading processes on bayesian attack graphs. In *Proceedings of the Second ACM Workshop on Moving Target Defense*, MTD '15, 2015.

[19] Xiaoyan Sun, Jun Dai, Peng Liu, Anoop Singhal, and John Yen. Using Bayesian Networks for Probabilistic Identification of Zero-Day Attack Paths. *IEEE Transactions on Information Forensics and Security*, 13(10), 2018.

[20] Marcel Frigault and Lingyu Wang. Measuring Network Security Using Bayesian Network-Based Attack Graphs. 2008.

[21] D.W. Hubbard, R. Seiersen, D.E. Geer, and S. McClure. *How to Measure Anything in Cybersecurity Risk*. Wiley, 2016.

[22] ISO/IEC. Common Criteria for Information Technology Security Evaluation Part 2. *Security*, (September), 2012.

[23] G Cooper. Computational Complexity of probabilistic inference using Bayesian belief networks (research note). *Machine Learning*, 42, 1990.

[24] Haipeng Guo and William Hsu. A Survey of Algorithms for Real-Time Bayesian Network Inference. *Papers from the Workshop on Real-Time Decision Support and Diagnosis Systems*, (1), 2002.

Security Driven Design Space Exploration for Embedded Systems

Lukas Gressl
Institute of Technical Informatics
Graz University of Technology
Graz, Austria
email: gressl@tugraz.at

Christian Steger
Institute of Technical Informatics
Graz University of Technology
Graz, Austria
email: steger@tugraz.at

Ulrich Neffe
NXP Semiconductors Austria GmbH
Gratkorn, Austria
email: ulrich.neffe@nxp.com

***Abstract*—With the advent of the Internet of Things (IoT) and Cyber Physical Systems (CPS), embedded devices have been gaining importance in our daily lives, as well as industrial processes. Independent of their usage, be it within an IoT system or a CPS, embedded devices are always an attractive target for security attacks, largely due to their continuous network availability and the importance of the data they handle. Thus, the design of such systems requires a thorough consideration of the various security constraints they are liable to. Introducing these security constraints, next to other requirements (e.g. power consumption, performance, etc.), increases the number of design choices that must be taken. As the various constraints are often conflicting each other, designers are faced with the complex task of balancing them. To support a system designer in this job, Design Space Exploration (DSE) tools can be facilitated. However, available DSE tools only offer a limited way of considering security constraints during the design process. In this paper we introduce a novel DSE framework, which allows the consideration of security constraints, in the form of attack scenarios, and attack mitigations, in the form of security tasks. Based on the descriptions of the system's functionality and architecture, possible attacks, and known mitigation techniques, the framework finds the optimal design for an secure IoT device or CPS. Our framework's functionality and its benefits are shown based on the design of a secure sensor system.**

***Index Terms*—Cyber Security; Embedded System Design; IoT Systems; industrial CPS; Mixed Criticality Design Space Exploration;**

I. Introduction

Cyber security has become an increasingly important factor when considering the utilization of Internet of Things (IoT) devices and Cyber Physical Systems (CPSs) for industries. Numerous large scale attacks on power plants, steel factories, and other industrial facilities have been reported in recent years [1], [2]. Especially the utilization of CPSs in the form of smart sensors, constantly connected to control systems, opened new ways of attacking these facilities. Therefore, the design of IoT devices and CPSs, planned to be used in the industry must consider all possible security attacks targeted on them. As the industrial environment demands fast reaction times and low maintenance costs, the used CPSs must guarantee high performance and low power consumption at the same time [3]. The consideration of the various constraints during the design process is a challenging problem for system designers, who must decide which functionality is implemented on what device. Finding the best solution among all the alternatives means solving a multi objective optimization problem, a task too complex to be performed manually.

To support the designer in this task, Design Space Exploration (DSE) frameworks are commonly utilized. These frameworks take as an input the functional description of the system, the descriptions of possible system architectures, and application and architecture characteristics. Based on these inputs such frameworks find the optimal architecture selection and function mapping. Classical DSE frameworks optimize for power or performance [4]. Other frameworks additionally consider security constraints either on an abstract or detailed level [5].

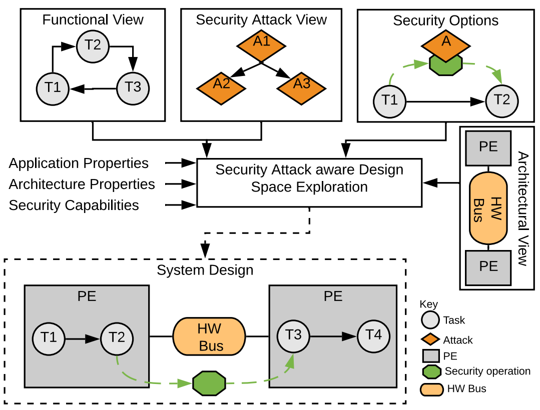


Fig. 1: System design exploration framework overview.

In this paper we present a novel method for the integration of security constraints and security capabilities into the DSE. We present a new framework, which supports designers in finding the optimal solution for a device under design using four distinct views: (i) the functional view lets the designer describe the system's behavior in form of a task graph; (ii) the architectural view describes the platform options of the device under design; (iii) the security attack view describes

978-1-7281-4113-8/19/$31.00 ©2019 IEEE

the view of a potential attacker on the system; (iv) the security options describe the various mitigation techniques offered by the system to counter potential attacks. These views serve as inputs to the framework. Based on these inputs, the framework finds the optimal selection of the system components, mapping of tasks to the selected components, and choice on the security mitigation techniques fulfilling the security constraints and optimizing the overall system's performance or power consumption. Figure 1 shows the basic overview of the tool, its distinct views and the solution produced by the framework.

With the approach described in this paper we make the following contributions: (i) to the best of our knowledge we are the first to introduce security constraints based on attack scenarios into the DSE, automatically considering security mitigation techniques; (ii) the presented approach is the first to explicitly select security operations based on the probability of security attacks; (iii) we show that a calculation of the attack probabilities based on the task mappings is feasible and can be optimized.

The paper is structured as follows: in Section II, various contributions in DSE, security modeling and attack description are discussed; Section III describes the modeling approach for designing secure systems; Section IV explains the framework's implementation; Section V shows the framework solving an example use case; Section VI concludes this paper and gives an outlook on future work.

II. Related Work

Information security modeling and threat analysis has attracted much interest in both research and industry in the last decades. To consider security measures in design phase of products, various modeling tools, and languages, such as the Unified Modeling Language, integrate extensions allowing the modeling of security [6]. Such tools support the designers in reflecting e.g. security protocols within the system's behavior, but do not consider the system's hardware architecture, nor the security's performance overhead, nor the attacker's view on the system under design. The framework described in this paper integrates security attacks and the architecture's security capabilities into the design flow, automatically proposing security operations.

Integration of the attacker's view is used for assessing the security of networks. Feng et al. [7] and Poolsappasit et al. [8] describe the representation of possible attackers using attack trees. Both authors use Bayesian Networks to assess the way security attacks propagate through the systems under design. These so called Bayesian Network Attack Graphs (BNAGs) can be used to localize security vulnerabilities in the overall system and assess the efficiency of security measures to reduce the system's information security risk [7]. These BNAGs are described by security experts, or can be built from historical security incidents [8]. The framework presented in here adds to these BNAGs, the functional, as well as the architectural description of the overall system. These additional views allow the selection of hardware components fulfilling the security constraints, and the selection of feasible security operations.

Classical DSE tools for embedded systems are well described in literature. These tools consider the power and performance of the individual implementation alternatives of the system's behavior on distinct components. Based on these power and performance characteristics the tools select the optimal system components, functionality implementations and scheduling [4], [9]–[11]. These classical optimization goals have been extended by functional safety [12] and cyber security [5], [13]–[16] in several works.

Related projects focusing on the integration of cyber security into DSE can be put into two categories: (i) works putting a detailed focus on one distinct problem at hand [13]–[15] and (ii) works introducing security aspects into the design space on an abstract level [5], [16]. The works in the first category optimize the throughput of special bus systems [13], seamlessly integrate security surveillance tasks into an already existing task schedule [14], or consider specific network security schemes for system integration [15]. As they deal with very specific problems, they do not consider the system's whole design space. The projects in the second category perform a mapping of tasks to hardware platform based on security requirements and security hardware extensions [5], or integrate security mechanisms into industrial control loops [16]. The SysML-sec project [17] integrates cryptographic mechanisms during hardware/software partitioning based on predefined attack scenarios. Unlike the framework presented in here, the contributions in (ii) do not consider the placement of the key material on the target platform, or the probability of successfully executing attacks. These projects consider security requirements and capabilities on an abstract level and do not go into detail of the security vulnerabilities or mitigation techniques. The framework presented in this paper introduces security attacks and mitigation techniques on a detailed level, without losing the overall view on the whole system design space. Thereby, it focuses on information security. The framework supports designers in their decisions on what functionality to implement on which architectural component, considering performance, power consumption, security attacks, and security capabilities.

III. Design Views

The system is represented by four distinct views: (i) the functional view, (ii) the architectural view, (iii) the security options view, and (iv) the security attack view. These views are used by designers to describe the distinct design options for the system, considering the different perspectives. With these perspectives, a design space is spanned, serving as an input to the framework, in which it searches for the optimal solution considering the system's security, performance, and power consumption.

A. Functional and Architectural View

The functionality of the system is described using task graphs [10], [11]. As attackers aim at compromising data handled by the system, we extend the common task graph model by adding a set of operations a task performs on a set of data entities. Each data entity is characterized with a set of security requirements (sr), defining its confidentiality ($conf$), authenticity ($auth$) or integrity (int) [18].

The architectural view comprises a set of processing elements (PEs) connected by bus systems. Both PEs and bus systems are described by a set of characteristics (chip area, costs, etc.) and distinct modes (power consumption, performance, etc.). Each PE mode defines a set of security capabilities, describing the PE's capability on encryption, authentication, and tamper safe storage. Additionally, the PE modes' are annotated with implementation vulnerability risks, defining the probability of a potential attacker to find a vulnerability to lever out the security capability. Figure 2 depicts the components of the functionality and architecture view.

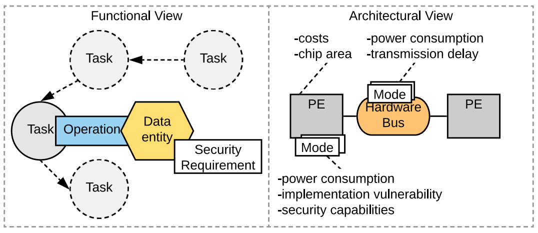


Fig. 2: Functional and platform architecture view.

The execution time of a task implemented on a PE running in a specific mode is described by its worst case execution time (WCET). This WCET must be defined for each mapping the designer wants to be explored by the framework.

B. Security Option and Attack View

Modeling security threats on systems is widely used for describing the security of network [8], [19]–[21]. These models build on the combination of attack graphs and Bayesian networks, forming so called Bayesian network based attack graphs (BNAGs) [22], [23]. Each node in the BNAG represents a distinct security attack and knows two states ($state_{no-success}$ and $state_{success}$), defining if the attack has been executed successfully or not. The state transition $state_{no-success} \rightarrow state_{success}$ describes the successful execution of an attack (at) with probability $P(at)$. The dependencies of the security attacks are represented by the directed edges between the nodes and the conditional distribution tables (CDTs). The CDT of each node represents its $P(at)$ considering the states of its parent nodes. The leaves of the BNAG describe attack goals on the overall system. The success probabilities of the attack goals are calculated using the BNAG's joint distribution table, calculated by the Bayesian chain rule. The system designers assign each attack goal a success probability threshold stating the maximal probability with which the attacker might reach his goal. The attack types used by the framework are derived from the STRIDE threat model [24], focusing on spoofing, tampering and information disclosure.

Integrating the attack view into the system design, attack nodes in the BNAG are assigned distinct tasks of the functional view as their victims. Each attack is furthermore described with an attack type (an_t) defining what security requirement ($conf$, $auth$, or int) of the handled data it aims to break. Figure 3 shows an example BNAG.

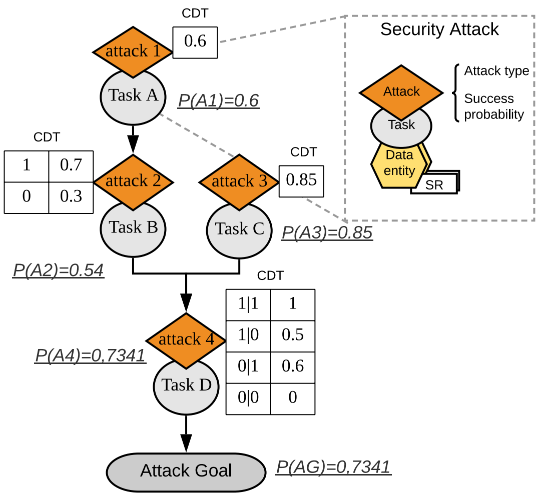


Fig. 3: BNAG example with detailed model of an attack node.

In the security operation view the system designers describe the available security operations, feasible to counter security attacks. Each security operation ($secOp$) is described by its security type and the probability of an attacker breaking it. This security break is modeled as an attack whose victim is the $secOp$ itself, and is assigned a $P(at)$. Modeling cryptographic functions (encryption, authentication, etc.) as $secOps$ also includes the declaration of the used key material. This key material is further categorized by defining its validity time. The validity time allows the designer to add the information if a $secOp$ uses e.g. a session key, master key, etc. The validty time influences the attacker's motivation to compromise the $secOp$, as e.g. compromising a master key valid over the whole product lifetime has a much greater impact on the overall system's security than merely compromising a session key. The probabilities of the attacks aiming both on victims and security operations must be estimated by experts. The experts must consider the attacker's capabilities and motivation for each attack. Judging the motivation of the attacker, the experts must take into account the value of the data entities the attacked tasks operate on. Additionally to the attack graph, the used $secOp$ also influence the system's performance, as each $secOp$ comes with an specific computational overhead.

C. Integration of Distinct Views

Based on the inputs from the functional, architectural, attack and security operations view and the optimization goal, the framework selects the architecture blocks, chooses the security operations including the used key material and maps the tasks to architecture blocks. Thereby, it produces the solution with the lowest success probability goals (security optimal), the solution with the best performance still satisfying all security thresholds (security/performance optimal), or the solution with the best performance, neglecting security.

1) General Task Mapping: The general mapping of tasks to PEs is performed using the task's WCETs. The WCETs reflect a task's execution time running on a PE in a specific mode (denoted by $pe(m)$) and must be defined to allow a successful task to PE mapping (denoted by $m(t, pe(m))$). The task to PE mapping is further restricted by the physical connections of the architecture platform. Given two connected tasks (t_1 and t_2) the mappings $t_1(pe_x(m_y))$ and $t_2(pe'_x(m'_y))$ are only valid if pe_x and pe'_x are physically connected. The mapping of t_1 and t_2 on the same PE is always valid.

2) Security Constraints Calculation: Additionally to the the general task mapping constraints all task PE mappings must fulfill the system security constraints. Fulfilling these security constraints, two requirements must be satisfied by the task PE mapping. First, each task must be mapped to a $pe(m)$ supporting the task's security functions. Second, no attack goal's success probability is allowed to exceed its threshold.

The security functions are determined by the operations said task performs on it's data entities [18]. The framework describes a basic set of operations (read (r), write (w), receive(rx), transmit (tx) and store(st)), denoted $o = (r, w, rx, tx, st, ...)$. Using this basic set of security requirements $sr = (conf, auth, int, ...)$ the set of security functions, comprising encryption (enc), signature ($sign$), secure storage (st_{sec}), denoted $sec_{func} = (enc, sign, st_{sec}, ...)$ is calculated using (1). Values for o and sr must be 0 or 1. Based on the information of a PE mode's security capabilities $sec_{cap} = (sc_{enc}, sc_{auth}, sc_{tss}, ...)$ (comprising encryption (sc_{enc}), authentication (sc_{auth}) and tamper-safe storage (sc_{tss})), it is determined if the mapping $m(t, pe(m))$ supports all sec_{func} using the $map(sec_{func}, sec_{cap})$ function as described in (2), with n being the number of all sec_{func} defined by the designer. Hence, it is checked whether $m(t, pe(m))$ fulfills the security mapping constraint. The sets o, sr, sec_{func}, and sec_{cap} are extendable by the user. The feasibility of their mappings must be added to said functions.

$$sec_{func}(o, sr) = \begin{pmatrix} (r \lor w) \land conf \\ (r \lor w) \land auth \\ st \land (auth \lor conf \lor int) \\ \ldots \end{pmatrix} \tag{1}$$

$$map(sec_{func}, sec_{cap}) = \bigcap_{i=1}^{n} sec_{func}[i] \land sec_{cap}[i] \tag{2}$$

The task's sec_{func} and PE mode's sec_{cap} are further used to determine what $secOp$ are used to secure the data entities handled by the task. For each sec_{func} the framework is given a set $secOp$ representing cryptographic operations, or other security operations, such as secure storage, etc. The type of the operation (which can either be sc_{enc}, sc_{auth}, or sc_{tss}, etc.) must be present in the PE mode's sec_{cap}. Cryptographic operations are further described by the used key material ($secOp(km)$). What $secOp(km)$ can be used on which $pe(m)$ is further restricted by sec_{cap}. Key material with a validity time covering the whole product's lifetime (e.g. master key, root certificate, etc.) is only allowed to be used on a $pe(m)$ supporting sc_{tss}. Furthermore, a task's $secOp(km)$ is restricted by the secure connection $sec_{conn} = (t_x(pe_y(m_z)), ..., t_{x'}(pe_{y'}(m_{z'})))$ of which it is a part of. Considering the task graph, the starting task of sec_{conn} is the last task performing a $secOp(km)$ mapped to a $pe_y(m_z)$. The end task of sec_{conn} is the last task performing the same $secOp(km)$ mapped to a neighboring $pe_{y'}(m_{z'})$. The used $secOp(km)$ of all tasks in the sec_{conn} is determined by the starting task. Figure 4 depicts an example secure connection.

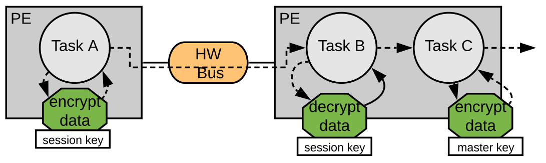


Fig. 4: Example of a connection secured by encryption spanning over multiple tasks mapped to two PEs.

Using the calculation of the secure connections, the influence of the used $secOp$ on the BNAG is determined. For each $t(pe(m)$ in sec_{conn} the attack nodes of the BNAG are determined. These attacks are mitigated by the $secOp$ if the attack's type an_t equals $secOp_t$. Each tasks securing its data using a $secOp$ forces the attacker to break the $secOp$ enabling the original attack targeting said task. Attacks aiming at the mitigating $secOp$ are, thus, added as parents to the node attacking t. The parent attacks' states are added to the node's conditional distribution table as an OR function. Thus, the breaking of at least one $secOp$ would enable the attacker to perform the original attack on the victim task. Figure 5 depicts an example integration of a security operation's attack into an existing BNAG.

Additionally to the inclusion of the $secOp$ attack nodes into the BNAG, each $pe(m)$ comes with a distinct implementation vulnerability risk $impl_{vuln} \in \mathbb{Q} : impl_{vuln} \in [0, 1]$. This risk reduces the attack success probabilities an_p of $secOp$ attack nodes used by the tasks mapped to said PE. This reduction is performed by multiplying $impl_{vuln}$ with an_p.

The assignment of the security operations' implementation vulnerability risks must be performed by the security experts. An accredited reference point for this rating can be found in the Common Criteria (CC) certification process. The CC certification provides a standardized way of assessing the security features of security products. The CC certification defines evaluation assurance levels (EALs), stating in what extend the security operations have been tested for possible vulnerabilities. These EALs range from simple functional testing (EAL1) to a formal verification including tests (EAL7). Additionally, each product must match a defined protection profile ensuring that all expected security features are provided [25]. The CC's official web page[1] provides a list of CC certified products and their certification documentations.

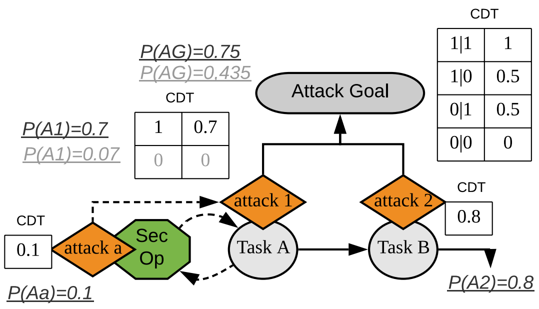


Fig. 5: Integration of attack on *secOp* into a predefined attack scenario described as a BNAG.

Based on the BNAG adapted with the security operations, the attack goals' success probabilities are calculated. The calculation of the attack goals is performed by applying the Bayesian chain rule on the BNAG, merging the marginals for each attack goal. Only if each attack goal's attack probability is below its defined threshold, the overall system functionality mapping to the architecture satisfies the security constraint. The implementation of the security mapping constraints is explained in more detail in Section IV.

IV. Framework Implementation

The framework is implemented based on the *DeSyDe* framework publicly available at *Github*[2]. The *DeSyDe* framework finds an optimal selection of PE modes and mapping of tasks to PEs considering power consumption and performance. The details on how the scheduling, performance, and power consumption was calculated can be found in the papers of Rosvall et al. [4], [9]. The following extensions and changes to the *DeSyDe* framework were implemented to achieve the before described calculation of an optimal and secure solution.

Task and hardware bus representation: The task representation was extended by adding the definition of data entities and operations to the tasks. Also the communication channels connecting the tasks were added the information of the passed data entities. The passed data entity's size is further used for calculating the communication overhead. The communication between PEs is changed using distinct physical connections. The usage of these connections changes the mapping of tasks to PEs as explained in Section III-C1. Furthermore, the communication via the physical connections affects the overall system performance. The transmission delay of the communication between two tasks over a physical link is calculated by multiplying the link's encoding factor with the transmission speed, times the exchanged data size. The data entity exchanged via the physical connection is determined by the communication channel between the tasks mapped to the linked PEs. This additional transmission delay is used for calculating the worst case communication time propagating into the overall system performance, as described in [4]. Considering the system's performance, the computational overhead induced by the security operations is integrated by the framework by adding it to the WCET of the respective tasks.

Security constraints: The calculation of the security constraints builds upon the knowledge of the security capabilities, security requirements, attack goals, and available security operations. The mapping between the security capabilities and tasks' security functions (2) is realized in a straight forward check, coming with an computational overhead of $O(N_T^{N_{PE_M}})$, with N_T being the number of tasks in the application graph, and N_{PE_M} being the number of all modes of all PEs. The framework determines the set of all sec_{conn}, denoted $SEC_{conn} = (sec_{conn_1}, ..., sec_{conn_\phi})$, with ϕ being the maximum number of all found sec_{conn}. The calculation of the attack goals is based on the BNAG and, thus, is a computationally hard problem. It was shown by G. F. Cooper that the exact probabilistic inference in a Bayesian network is an NP hard problem [26]. This computational overhead is further aggravated by the fact that each $m(t, pe(m))$ changes the BNAG by adding the respective *secOp*(km) attacks with the supported key material. Hence, the recalculation of the attack goals is necessary for each task to PE mapping. The NP hardness of the exact probabilistic inference can be reduced using approximate inferences through sampling [27].

Reducing the number of necessary attack goal calculations induced by the different mapping possibilities, the BNAG is searched for all attacks influencing the attack goals, disregarding all other attacks. All PEs are ordered according to the implementation vulnerabilities of their respective modes. The PE modes' implementation vulnerabilities are denoted by the function $\delta(pe(m))$. The ordered set $\Delta = (\delta_1(pe_x(m_y)) \leq, ..., \leq \delta_n(pe_{x'}(m_{y'}))$ is then used for calculating the attack goals for each possible mapping $m(t, pe(m))$. The framework permutes all feasible $m(t, pe(m))$ using Δ and calculates the attack goals for each permutation described as $P = (m_1(p_1, pe_a(m_b)), ..., m_{N_P}(p_{N_P}, pe_{N_{PE}}(m_{N_m})))$, with N_{PE} being the number of all pe and N_m the number of modes of the current pe. Considering the permutations, not all of them represent feasible task mappings, as a system mapping $m(p_x, pe_a(m_b))$ cannot map $m(p_y, pe_a(m_c))$. These infeasible combinations are eliminated before starting the attack goal calculation. For each P the possible key sets of the used *secOp* are determined by traversing all sec_{conn} in SEC_{conn} and checking the sec_{cap} of each $m(p, pe(m))$ of sec_{conn}. Based on sec_{cap} the set of possible keys KM for all involved *secOp* is determined (*secOp*(KM)). In each P the combinations of

[1]https://www.commoncriteriaportal.org/
[2]https://github.com/forsyde/DeSyDe

all $secOp(KM)$ are iterated. In each iteration the respective BNAG is formed to calculate the attack goals (ag) of said iteration ($ag(i)$). At the end of each iteration, $ag(i)$ is added to the set $ag(P)$ which is checked upon at the end of said permutation P. As Δ is ordered descending, the calculation of $ag(P)$ can be halted when a P_{insec} is found, not containing any $ag(i)$ satisfying the defined attack thresholds. All following P can be rendered insecure without further attack goal calculation, until a P is reached, in which at least one $m'(p_x, pe_{a'}(m_{b'}))$ has a $\delta'(pe_{a'}(m_{b'}))$ smaller than $\delta(pe_a(m_b))$ of $m(p_x, pe_a(m_b))$ in P_{insec}. Thereafter, the algorithm continues calculating the security goals for all following P until reaching the next P_{insec}.The algorithm's computational worst case is not finding any mapping dissatisfying the security goals' thresholds. In all other cases the algorithm will reduce the number of necessary inference calculations in the BNAG.

V. Evaluation and Results

The functionality of the framework is shown by designing a sensor system consisting of a mobile control device and a system of sensor devices. In this example the design of the control device and one sensor is shown. The use case is inspired by the system presented in [28]. The simplified system's functionality is presented in Figure 6. It consists of the control device configuring the sensor, the configured sensor accumulating environmental data, and the control device retrieving this data. Both control and sensor device operate on the data entities *config* and *sensor data*. Both data entities come with security requirements: $conf$, $auth$, and int.

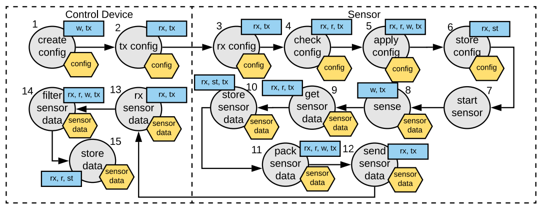


Fig. 6: Functionality of the sensor usecase showing tasks for the control device and the sensor.

Securing the data elements used by the control device and sensor, the system is given a set of security operations (encryption, decryption, sign, verify) which can either be used with a long time master key, or a short time session key. As breaking a security system using a long time master key potentially leads to a higher impact than breaking a short time session key, the attack probability on the master key was rated with 0.3, on the session key with 0.1. This rating reflects the attackers motivation. For the hardware platforms of the control device and the sensor, the following abstract components were considered: for communication between control device and sensor, both devices are equipped with a BLE radio. The sensor also contains a sensing instrument. Both devices can be equipped with an integrated secure element (SE), and a microcontroller (MCU). The MCU offers cryptographic functionality using a co-processor (CryptCP) or software based cryptography (SWCrypt), but lacks the ability for storing data in a tamper safe way. The components are internally connected to each other using bus systems. Table I lists the possible components, which can be selected for the hardware platform of the control- and sensor-device, and the WCETs of the security relevant tasks running on these components. The WCETs serve the purpose of showing the performance differences of the secure vs. non-secure mappings. The components are applicable for both the control, as well as the sensor device.

TABLE I: Tasks and WCETs on respective PEs

PE	t1	t4	t5	t6	t9	t10	t11	t14	t15
MCU_1	70	60	80	100	100	110	150	130	80
MCU_2	75	65	90	120	120	140	200	180	85
SE_1	80	90	100	130	150	120	180	180	100
SE_2	120	130	150	180	170	160	220	220	150

Additionally to the estimated tasks' WCETs, table II lists the components security operation implementation vulnerabilities, their security capabilities, as well as the computational overheads of the security operations performed on the distinct components. The implementation vulnerability risks (IV) are estimated based on the EALs assigned by CC certifications, which can be found at their webpage[3]. The security capabilities (SC) are abbreviated with e (encryption/decryption), s (signing/verifying), and t (tamper safe storing). The timings and the IVs are abstracted and serve the purpose of showing the framework's correct mapping and component selection.

TABLE II: PEs' characteristics (IV, SC) and security operation WCETs

PE	IV	SC	enc	dec	sign	ver
SE_1	0.2 (EAL4+)	e, s, t	8	8	12	12
SE_2	0.05 (EAL6+)	e, s, t	12	12	18	18
MCU_1	0.35 (CryptCP)	e, s	8	8	10	10
MCU_2	0.5 (SWCrypt)	e, s	30	30	40	40

To show security aware mapping process an attack scenario, a BNAG as depicted in Figure 7 was used. To estimate the probabilities of the individual attack nodes, we considered the system to be attacked by technically experienced persons, who are able to sniff and intercept the wireless communication and have access to the internal bus systems of the sensor and configuration device. The attacks' aims are categorized into three scenarios. In the first scenario, the sensed data is attacked on the sensor device. In the second scenario, the sensed data is attacked on the control device. In the last scenario the configuration data is attacked. The unconditional probability of each attack node is shown in Figure 7.

To discard insecure PE selection and task mapping, the security goals were assigned thresholds. All attack scenarios were rated with a threshold of 5% each. The thresholds were chosen in such a way that a maximal difference for non-secure and secure mappings are visible.

[3]https://www.commoncriteriaportal.org/products/#AC

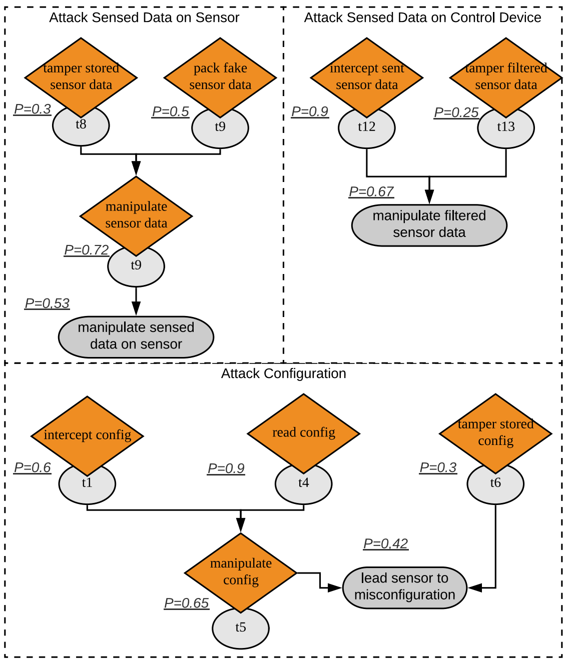

Attack Sensed Data on Sensor
Attack Sensed Data on Control Device
tamper stored sensor data
pack fake sensor data
P=0.3
t8
P=0.5
t9
intercept sent sensor data
tamper filtered sensor data
P=0.9
t12
P=0.25
t13
manipulate sensor data
P=0.72
t9
P=0.67
manipulate filtered sensor data
P=0.53
manipulate sensed data on sensor
Attack Configuration
intercept config
P=0.6
t1
read config
P=0.9
t4
tamper stored config
P=0.3
t6
P=0.42
manipulate config
P=0.65
t5
lead sensor to misconfiguration

Fig. 7: Security attack scenarios described as BNAGs.

Using the described system's functionality, architectural possibilities, security attack scenarios, and possible security operations with respective key material, 40 solutions were found to be secure. In comparison, without the security constraints, 6096 solutions were found. Hence, only around 0.6% of the overall solution space was found to be secure. Calculating all possible mappings without sorting the PEs' modes according to the implementation vulnerabilities, the first solution was found after 5 hours 26 min. Using the breaking condition of the implementation vulnerabilities, the calculation took 1 hours 28 minutes. Hence, the break condition increased the performance 73%. Figure 8 shows the most secure system platform composition and task mapping, the solution with the highest performance, still satisfying all security constraints, and the solution with the highest performance but without considering security constraints at all. It can be seen that the most secure solution only makes use of SE_3 directly connected to the radio. The secure solution with optimal performance utilizes MCU_1 in combination with SE_1 for both the control device, as well as the sensor. Without considering the security of the system, the framework would not include any SE into the platform's architecture at all, but map all tasks to MCU_1. For the most secure, as well as the secure but performance optimal solution, the framework only chooses the usage of session keys. For the most secure solution, the framework places all tasks on the SE with the highest EAL rating, without utilizing any other components at all which might not be feasible for the system's implementation. This must still be considered by the designer using the framework.

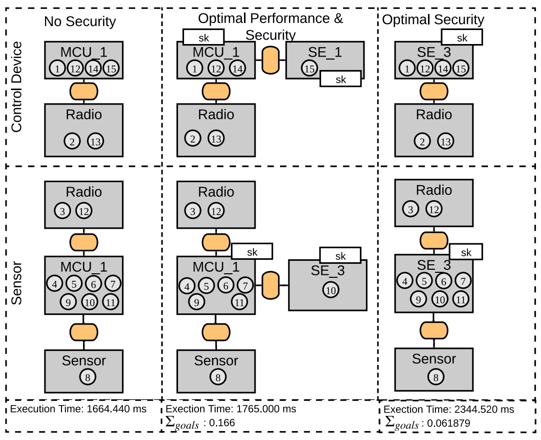

No Security
Optimal Performance & Security
Optimal Security
Control Device
Sensor
MCU_1
SE_1
SE_3
sk
Radio
Sensor
Execution Time: 1664.440 ms
Exection Time: 1765.000 ms
Σ_{goals} : 0.166
Exection Time: 2344.520 ms
Σ_{goals} : 0.061879

Fig. 8: Proposed system architecture and process allocation for optimal performance, optimal performance still satisfying all security goals and highest security, including performance values and accumulated security goals.

This industrial sensor usecase shows how the framework is used in a design process to find the best suited platform and task mapping when considering optimal performance with or without fulfilling security constraints.

VI. Conclusion and Future Work

In this paper we describe a novel approach for introducing security attack scenarios and security functionality selection into a system design process of embedded systems. The introduced framework allows a designer to specify the system's functionality, the possible hardware architecture components running the defined functionality, model based attack scenarios, and a range of security operations usable by the system. The framework uses these descriptions to suggest an optimal hardware component selection, task to component mapping, and an optimal usage of security operations with respective key material. Based on an embedded sensor system use case, the functionality of the framework during design time was described and its feasibility for finding not only the most secure, but also the e.g. best performing and still secure solution was shown.

This work can be seen as a first important step for introducing detailed security requirements into the design space exploration of embedded systems based on attack probabilities. Looking into related work security constraints and usable security operations can be formalized more specifically. The framework is meant as a tool to simplify the design process of secure embedded systems and emphasize the importance of information security at an early stage of the system design process. In this sense, the framework is meant to introduce the experience and knowledge of security experts in the design flow in an easier way. Thus, the framework supports system designers not experienced in information security during the first phases of system design. The security expert knowledge is, however, also the limiting factor in this process. The information about the assurance level of a security operation's implementation must be provided by the security experts a priori. The CC certification can only give a guidance about the security level of certain secure devices. However, this guidance is not directly usable for judging what vulnerability can be found by attackers considering the security operations. As these security operations are meant to secure the tasks running on the platform, this information is critical to the overall system design. Hence, a standardized method for judging these security level of such security operations would be needed. Based on this method, the information could be stored in a repository, which could be used to support the design process of secure embedded systems.

For future work, this work will be extended to support not only security attack scenarios but a complete security risk estimation. This extension would add critical and important information to the DSE process and allow the framework to consider additional constraints, such as expected financial loss caused by successfully executed security attacks, etc. Furthermore, we want to show how the framework can also be used on a large scale system, covering not only the embedded world, but also higher layers, such as web servers and applications, etc.

Acknowledgment

Project partners are NXP Semiconductors Austria GmbH and the Technical University of Graz. This work was supported by the Austrian Research Promotion Agency (FFG) within the project UBSmart (project number: 859475).

References

[1] Mohammed Nasser Al-mhiqani, Rabiah Ahmad, Warusia Yassin, Aslinda Hassan, Zaheera Zainal Abidin, Nabeel Salih Ali, and Karrar Hameed Abdulkareem. Cyber-Security Incidents : A Review Cases in Cyber-Physical Systems. 2018.

[2] Chih Ta Lin, Sung Lin Wu, and Mei Lin Lee. Cyber attack and defense on industry control systems. *2017 IEEE Conference on Dependable and Secure Computing*, 2017.

[3] Lorenzo Pagliari, Raffaela Mirandola, and Catia Trubiani. Multi-modeling Approach to Performance Engineering of Cyber-Physical Systems Design. *Proceedings of the IEEE International Conference on Engineering of Complex Computer Systems, ICECCS*, 2018.

[4] Kathrin Rosvall, Nima Khalilzad, George Ungureanu, and Ingo Sander. Throughput Propagation in Constraint-Based Design Space Exploration for Mixed-Criticality Systems. *Proceedings of the 9th Workshop on Rapid Simulation and Performance Evaluation: Methods and Tools - RAPIDO '17*, 2017.

[5] Ingo Stierand, Sunil Malipatlolla, Sibylle Froschle, Alexander Stuhring, and Stefan Henkler. Integrating the security aspect into design space exploration of embedded systems. *Proceedings - IEEE 25th International Symposium on Software Reliability Engineering Workshops, ISSREW 2014*, 2014.

[6] Jan Jürjens. Sound methods and effective tools for model-based security engineering with UML. *Proceedings. 27th International Conference on Software Engineering, ICSE*, 2005.

[7] Nan Feng, Harry Jiannan Wang, and Minqiang Li. A security risk analysis model for information systems: Causal relationships of risk factors and vulnerability propagation analysis. *Information Sciences*, 2014.

[8] Nayot Poolsappasit, Rinku Dewri, and Indrajit Ray. Dynamic Security Risk Management Using Bayesian Attack Graphs. *IEEE Transactions on Dependable and Secure Computing*, 9(1), 2012.

[9] Kathrin Rosvall and Ingo Sander. A Constraint-based Design Space Exploration Framework for Real-time Applications on MPSoCs. *Proceedings of the Conference on Design, Automation & Test in Europe*, 2014.

[10] Bastian Knerr. *Heuristic Optimisation Methods for System Partitioning in HW / SW Co-Design.* PhD thesis, Vienna University of Technology, 2008.

[11] Kathrin Rosvall, Tage Mohammadat, George Ungureanu, Johnny Oberg, and Ingo Sander. Exploring power and throughput for dataflow applications on predictable NoC multiprocessors. *Proceedings - 21st Euromicro Conference on Digital System Design, DSD 2018*, 2018.

[12] Domiian Tma-Selicean and Paul Pop. Design Optimization of Mixed-Criticality Real-Time Embedded Systems. *ACM Transactions on Embedded Computing Systems*, 14(3), 2015.

[13] Yong Xie, Gang Zeng, Ryo Kurachi, Hiroaki Takada, and Guoqi Xie. Security/Timing-aware Design Space Exploration of CAN FD for Automotive Cyber-Physical Systems. *IEEE Transactions on Industrial Informatics*, 2018.

[14] Monowar Hasan, Sibin Mohan, Rodolfo Pellizzoni, and Rakesh B. Bobba. A design-space exploration for allocating security tasks in multicore real-Time systems. *Proceedings of the 2018 Design, Automation and Test in Europe Conference and Exhibition, DATE 2018*, 2018.

[15] Eunusk Kang. Design Space Exploration for Security. *IEEE Cybersecurity Development Design*, 2016.

[16] Bowen Zheng, Peng Deng, Rajasekhar Anguluri, Qi Zhu, and Fabio Pasqualetti. Cross-Layer Codesign for Secure Cyber-Physical Systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 35(5), 2016.

[17] Letitia W. Li, Florian Lugou, and Ludovic Apvrille. Security-aware Modeling and Analysis for HW/SW Partitioning. *Proceedings of the 5th International Conference on Model-Driven Engineering and Software Development*, 2017.

[18] Lukas Gressl, Christian Steger, and Ulrich Neffe. A Security Aware Design Space Exploration Framework. In *Proceedings of the Fourteenth International Conference on Systems ICONS 2019*, Valencia, Spain, 2019. ThinkMind(TM) Digital Library.

[19] Erik Miehling, Mohammad Rasouli, and Demosthenis Teneketzis. Optimal defense policies for partially observable spreading processes on bayesian attack graphs. In *Proceedings of the Second ACM Workshop on Moving Target Defense*, MTD '15, 2015.

[20] Xiaoyan Sun, Jun Dai, Peng Liu, Anoop Singhal, and John Yen. Using Bayesian Networks for Probabilistic Identification of Zero-Day Attack Paths. *IEEE Transactions on Information Forensics and Security*, 13(10), 2018.

[21] Marcel Frigault and Lingyu Wang. Measuring Network Security Using Bayesian Network-Based Attack Graphs. 2008.

[22] Vivek Shandilya, Chris B. Simmons, and Sajjan Shiva. Use of attack graphs in security systems. *Journal of Computer Networks and Communications*, 2014, 2014.

[23] David Heckerman and John S. Breese. Causal independence for probability assessment and inference using Bayesian networks. *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans.*, 26(6), 1996.

[24] Shawn Hernan, Scott Lambert, Tomasz Ostwald, and Adam Shostack. Threat modeling-uncover security design flaws using the stride approach. *MSDN Magazine-Louisville*, 2006.

[25] Common Criteria for Information Technology Security Evaluation Part 2. 2012.

[26] G Cooper. Computational Complexity of probabilistic inference using Bayesian belief networks (research note). *Machine Learning*, 42, 1990.

[27] Haipeng Guo and William Hsu. A Survey of Algorithms for Real-Time Bayesian Network Inference. *Papers from the Workshop on Real-Time Decision Support and Diagnosis Systems*, (1), 2002.

[28] Thomas Ulz, Thomas Pieber, Christian Steger, Sarah Haas, Rainer Matischek, and Holger Bock. Hardware-Secured Configuration and Two-Layer Attestation Architecture for Smart Sensors. *Proceedings - 20th Euromicro Conference on Digital System Design, DSD 2017*, 2017.

Security Based Design Space Exploration for CPS

Lukas Gressl
Institute of Technical Informatics,
Graz University of Technology
gressl@tugraz.at

Alexander Rech
Institute of Technical Informatics,
Graz University of Technology
rech@tugraz.at

Christian Steger
Institute of Technical Informatics,
Graz University of Technology
steger@tugraz.at

Andreas Sinnhofer
NXP Semiconductors Austria
GmbH
andreas.daniel.sinnhofer@nxp.com

Ralph Weissnegger
CISC Semiconductor GmbH
r.weissnegger@cisc.at

ABSTRACT

Security is vital for Cyber-Physical Systems (CPS). CPS have become important in the industry, as they are often used to process sensitive data. Hence, they are valuable targets for attackers. As they are subject to manifold constraints, such as performance, power dissipation, etc., and security measures always induce additional delay, energy consumption, etc., designing secure CPS is a complex task. To find an optimal solution considering these various constraints, a design space exploration (DSE) must be performed. In this paper, we present a framework, capable of considering security constraints described as attack scenarios and automatically selecting appropriate security measures and secure key placement. We show the framework's feasibility by designing a secure sensor system.

CCS CONCEPTS

• **Security and privacy** → **Formal security models**; *Security requirements*; *Logic and verification*; • **Computer systems organization** → *Embedded systems*;

KEYWORDS

Cyber Security; Embedded System Design; Industrial CPS; Design Space Exploration

ACM Reference Format:
Lukas Gressl, Alexander Rech, Christian Steger, Andreas Sinnhofer, and Ralph Weissnegger. 2020. Security Based Design Space Exploration for CPS. In *The 35th ACM/SIGAPP Symposium on Applied Computing (SAC '20), March 30-April 3, 2020, Brno, Czech Republic.* ACM, New York, NY, USA, Article 4, 3 pages. https://doi.org/10.1145/3341105.3374058

1 INTRODUCTION

The utilization of Cyber-Physical Systems (CPS) in the industry opened a new attack surface for potential attackers. In the last decade, numerous attacks were reported [1], which used CPS as a weak point. This trend emphasizes the need for considering cybersecurity attacks when designing CPS. Additionally to the security constraints, CPS must often adhere to requirements considering their energy consumption and performance. As security measures induce additional delay and power dissipation, their selection must be performed in conformity with all other system constraints. With these opposing constraints, finding an optimal task allocation and system partitioning poses a complex task, usually solved with the help of design space exploration (DSE) tools [6, 9]. In this paper, we present a framework capable of finding an optimal system partitioning and task scheduling, based on security constraints modeled as Bayesian attack graphs (BAGs). The presented framework automatically selects the optimal security measures and key placement under consideration of potential security attacks and overall system performance. The framework's feasibility is shown in an industrial use case considering multiple layers.

2 RELATED WORK

Modeling security attack scenarios has been well studied in the field of secure network systems, in which the attacker's perspective is used to formalize attack paths modeled as BAGs. These BAGs are either taken from historic data or formalized by security experts [2]. Considering DSE for secure embedded systems, related projects either take an abstract or focused approach. Works taking a focused approach lack the consideration of the overall system design space [9], whereas works abstractly introducing security constraints cannot formalize system attack scenarios unambiguously [8, 10]. In contrast, the framework presented here aims at providing system designers with an approach to model security constraints in a detailed way without losing the ability to consider the whole system design space.

3 FRAMEWORK DESIGN

The framework's design builds on [4]. Its design consists of four perspectives, with which the designer can describe the system under design.

Task Graph (I) and System Architecture (II): The system's functionality is described as a task graph, in which each task performs a set of operations on a range of data entities. The data entities come with a distinct set of security requirements (sr). The system's architecture is described by hardware components connected with communication buses. Both hardware components and communication buses are characterized with costs, power consumption, etc. Each hardware component is described by a set of security capabilities (*secCap*), defining the components' ability to support cryptographic algorithms, task encapsulation, tamper safe storage, etc. The vulnerability risk (VR), given for each component, defines the probability of an attacker finding a weakspot in the implementation of the *secCap*. The performance of a task implemented on a distinct hardware component is described by its worst case execution time (WCET). Each hardware component is further assigned to a predefined device, which is used for determining the placement of the key material.

Attack Graph (III) and Security Functions (IV): The attack scenarios are modeled using BAGs. Each node in the BAG represents a distinct attack step aiming at a specific task and is assigned a success probability. Each BAG node has a distinct attack type, specifying what sr of the victim task's data entities it aims to break. The BAG's leafs represent the goals an attacker aims to reach. Using the Bayesian chain rule, the goals' success probability is calculated. Furthermore, the DSE framework is given a set of security functions capable of mitigating potential attacks, based on their attack types. Certain security functions, such as cryptographic algorithms, rely on specific key material. These security functions and the used key material (including keys derived from other keys) might also be subject to attacks.

BAG and attack goal calculation: Using the inputs of the four perspectives, the framework selects the platform components, allocates the tasks, and selects the security functions to secure the data entities from potential attackers. Thereby, the framework ensures the general reachability of the task allocation. For each task, the framework calculates a set of security operations (*secOp*) based on the data entities' sr it operates on. Based on these *secOp*, the framework calculates the security functions (sec_{func}) with which to ensure the data entities' sr. The framework only considers a task allocation to be secure if all security functions of the task are executable by the *secCap* of the hardware component it is allocated on. Only solutions fulfilling these constraints are further regarded by the framework.

From each mapping fulfilling the security allocation constraints, the framework builds the BAG graph, considering the additional attacks on a single sec_{func} and the used key material. The key placement is further restricted by the validity time of the key and the *secCap* provided to the sec_{func} using this key. The framework adds the attacks of each used sec_{func} to the attacks aiming at said task as parents, including the key disclosure attacks aiming at the used key material. The success probabilities of the added attacks are further adapted by multiplying them with the VR of the hardware components, the victim tasks are mapped on. Hence, the distinct mappings influence the BAG, resulting in different attack goal probabilities for each solution.

Table 1: Options for SE, the MCU, and the SP. VR and security overhead were estimated [3, 5, 7].

HWC	Security Options	VR	Security Overhead
SE	EAL 5+	0.1	tss(30µs), te (10µs)
			sym(8µs), asym(150ms)
	EAL 6+	0.05	tss(30µs), te (2µs)
			sym(8µs), asym(150ms)
MCU	HWC, FW, TZ	0.15	te (2µs)
			sym(16µs), asym(100ms)
	SWC, TZ	0.35	sym(8µs), te(2µs)
SP	HWC, TZ, tss	0.05	tss (30µs), te (2µs)
			sym(7µs), asym(150ms)
	HWC, TZ, HSM supp.	0.1	tss (30µs), te (2µs)
			sym(8µs), asym(170ms)
	SWC tested, SZ	0.55	sym(12µs), te (2µs)
	SWC	0.7	sym(6µs)

4 EXPERIMENTS

Using the framework, an industrial sensor system was designed considering different possibilities for the system realization. The system comprises a sensor device, a gateway, and an analysis server. The sensor and the gateway are equipped with a sensing module, a microcontroller (MCU), a Bluetooth Low Energy (BLE) radio and an optional secure element (SE). The gateway comes with and additional WiFi radio. The data analysis is either implemented on a security-enhanced server, a commonly used server with Hardware Secure Module support (HSM supp), or a server with limited security capabilities. The security extensions come in form of symmetric (sym) and asymmetric (asym) hardware (HWC) or software-based cryptography (SWC), task encapsulation (te) with Firewall (FW) and TrustZone (TZ), and tamper safe storage (tss). These architecture options, comprising the VR and their performance overheads, are listed in Table 1. The VR were estimated by security experts, using the Common Criteria[1] certifications as a basis, where possible.

The functionality of the overall system setup consists of three phases: during the configuration phase, the sensor is configured and activated via the gateway; in the data accumulation phase, the sensor gathers information and sends it to the gateway, which filters the received sensor information; in the data analysis phase, the filtered data is sent to the analysis server, which processes and monitors the data. Possible attacks on the system encompass compromising of the sensor configuration, the faking of the sensor activation message, manipulating the accumulated sensor data, faking the filtered data, and disclosing the analyzed data.

Table 2 lists the available security functions and their probabilities of being successfully attacked (AP). Furthermore, the table lists the key material and their risks of being exposed by potential attackers. The key material comprises (i) a symmetric master key (*smk*), (ii) a binding key (*sbk*), (iii) a session key (*ssk*) that is derived from the *smk*, and (iv) a asymmetric certificate (*cert*). The AP for both security functionality and the key material in Table 2 were estimated based on assuming that neither the security function nor

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

Table 2: Used security functions and key material.

Security Function	Key	AP
Cryptography	Master Key	0.1
	Binding Key	0.05
	Session Key	0.01
	Certificate	0.1
Task Encapsulation	/	0.05
Secure Storage	/	0.05

Key (Type)	Lifetime	AP	Derivation
Master Key (sym)	long	0.05	/
Binding Key (sym)	long	0.02	/
Session Key (sym)	short	0.01	Master Key
Certificate (asym)	long	0.05	/

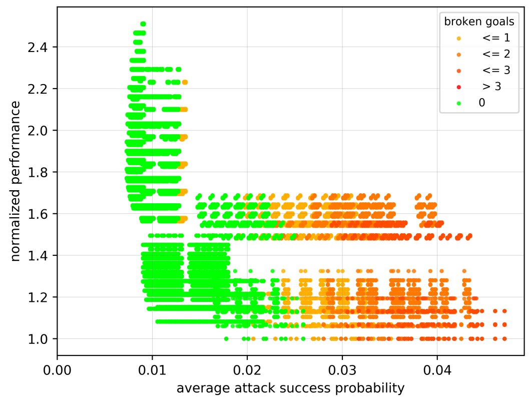


Figure 1: Solutions found for system under design.

the key material was mapped to a component offering attack mitigation. Said mitigation is based on the capability of the hosting hardware component. Furthermore, the AP is influenced by the used key material and what the attacker gains by the attack. The same assumptions were made when estimating the APs on the disclosure of the key material itself.

Figure 1 shows the solutions found by the framework based on the average attack probability (avg ap) and performance normalized to the solution with the highest performance. Table 3 shows the most secure, the fastest, and the fastest secure solution. These solutions use the *ssk* for securing the exchanged data. Furthermore, the most secure system relies on using the *sbk* for securing the communication within the single devices (sensor, sensor controller gateway, server platform). The system configuration with optimal performance puts as many tasks on the same hardware component as possible. Hence, task encapsulation is used to secure the intra-component-communication, saving security overhead for cryptography.

5 CONCLUSION AND FUTURE WORK

In this paper, a security-based DSE framework, considering security attacks and key material placement is presented. The paper shows the framework's capability of finding the most secure or performance optimal, yet secure solution, considering a secure system partitioning, task allocation, security functionality selection, and key material placement for a complex system comprising multiple abstraction layers. A known limitation to the framework is its reliance on security expert knowledge for assessing the attack scenarios and security vulnerability risk of the utilized components. Future work will focus on integrating other security assessment techniques.

Table 3: Solutions found by the framework

HWC	Most secure	Fastest	Fastest secure
MCU	HWC, FW, TZ	SWC, TZ	HWC, FW, TZ
SE	EAL 6+	EAL 5+	EAL 6+
SP	HWC, TZ, tss	SWC	HWC, TZ, HSM
avg ap	0.01	0.039	0.019
norm perf.	~ 2.57	1.0	~ 1.014
Key Placement			
MCU	*ssk*, *sbk*	*ssk*	*ssk*, *sbk*
SE	*sbk*, *smk*, *cert*	*ssk*, *smk*, *cert*	*sbk*, *smk*
SP	*ssk*, *smk*, *cert*	*ssk*, *smk*, *cert*	*ssk*, *smk*

ACKNOWLEDGMENTS

Project partners are NXP Semiconductors Austria GmbH, CISC Semiconductor and Graz University of Technology. This work was supported by the Austrian Research Promotion Agency (FFG) within the project STIP (project number: 867914).

REFERENCES

[1] M. N. Al-mhiqani et al. 2018. Cyber-Security Incidents : A Review Cases in Cyber-Physical Systems. *Int. Journal of Advanced Computer Science and Applications* (2018).
[2] N. Feng et al. 2014. A security risk analysis model for information systems: Causal relationships of risk factors and vulnerability propagation analysis. *Information Sciences* 256 (2014).
[3] Sharon Levy. 2015. Performance and Security of ECDSA. *Computer Science* (2015).
[4] L. Gressl others. 2019. Consideration of Security Attacks in the Design Space Exploration of Embedded Systems. In *2019 22nd Euromicro Conf. on Digital System Design (DSD).*
[5] L. Raju and M. Sumathi. 2015. Secured High Throughput of 128-bit AES Algorithm based on Interleaving Technique. (2015).
[6] K. Rosvall et al. 2018. Exploring power and throughput for dataflow applications on predictable NoC multiprocessors. *Proc. of 21st Euromicro Conf. on Digital System Design, DSD 2018* (2018).
[7] T. Schläpfer and A. Rüst. 2019. Security on IoT Devices with Secure Elements. *Embedded World Exhibition and Conf.* (2019).
[8] I. Stierand et al. 2014. Integrating the security aspect into design space exploration of embedded systems. *Proceedings - IEEE 25th International Symp. on Software Reliability Engineering Workshops, ISSREW 2014* (2014).
[9] Y. Xie et al. 2018. Security/Timing-aware Design Space Exploration of CAN FD for Automotive Cyber-Physical Systems. *IEEE Trans. on Industrial Informatics* (2018).
[10] B. Zheng et al. 2016. Cross-Layer Codesign for Secure Cyber-Physical Systems. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems* (2016).

Towards an Automated Exploration of Secure IoT/CPS Design-Variants

Lukas Gressl[1], Michael Krisper[1], Christian Steger[1], and Ulrich Neffe[2]

[1] Graz University of Technology (TU Graz), Austria
(gressl, michael.kripser, steger)@tugraz.at
[2] NXP Semiconductors Austria GmbH, Austria
ulrich.neffe@nxp.com

Abstract. The advent of the Internet of Things (IoT) and Cyber-Physical Systems (CPS) enabled a new class of connected, smart, and interactive devices. With their continuous connectivity and their access to valuable information in both the digital and physical world, they are highly attractive targets for security attackers. Integrating them into the industry and our daily used devices adds new attack surfaces. These potential threats call for special care of security vulnerabilities during the design of IoT devices and CPS. Due to their resource-constrained nature, designing secure IoT devices and CPS poses a complex task, considering the selectable hardware components and task implementation alternatives. Researchers proposed a range of automatic design tools to support system designers in their task of finding the optimal hardware selection and task implementations. Said tools offer a limited way of modeling attack scenarios for a system under design. The framework proposed in this paper aims at closing this gap, offering system designers a way to consider security attacks and security risks during the early phase of system design. It offers designers the possibility to model security constraints from the view of potential attackers, assessing the probability of successful security attacks and the resulting security risk, alike. We demonstrate the framework's feasibility and performance by revisiting an industry partner's potential system design of a future IoT device.

Keywords: Cyber Security · Embedded System Design · Secure IoT Systems · Secure CPS · Secure Embedded Consumer Devices

1 Introduction

The increasing utilization of the Internet of Things (IoT) in the commercial market and cyber-physical systems (CPS) in the industry, opened a new attack surface. In the last decades, numerous cybersecurity exploits have been documented [1, 11]. The ongoing integration of such systems demands the consideration of cybersecurity exploits throughout the whole system design process. Introducing security measures causes additional performance delay and power consumption, contradicting the systems' requirements for fast response times and high energy efficiency [19]. Considering the hardware and task implementation alternatives, finding the optimal solution satisfying performance and security poses a multi-objective optimization problem. Designers rely on automatic design space exploration (DSE) tools are used. There exist both classical DSE tools focusing on performance and power consumption [8,13], and DSE frameworks offering the consideration of security constraints in a limited way [6,7,10,16,18,20].

The framework presented in this paper introduces a new approach to introducing security constraints in early IoT/CPS design, based on both attack graphs and risk trees. Among a set of possible hardware components and task implementation alternatives, the framework finds the optimal selection of hardware components and task placements considering the system's power consumption, performance, security attack mitigation capability, and security risk exposure. In this paper, we make the following contributions: (i) To the best of our knowledge, the framework presented here is the first to allow the consideration of security constraints modeled as Bayesian attack graphs (BAGs) and risk trees during early IoT/CPS design. (ii) We integrate both approaches and show their advantages and disadvantages. (iii) We show the framework's feasibility based on a secure consumer device use case and the scalability of our approach.

The paper is structured as follows: Section 2 discusses related projects in DSE, security attack and risk modeling; Section 3 describes the security modeling approach, the framework's design and implementation; Section 4 shows the impact of both security modeling approaches on the secure consumer device use-case; Section 5 gives a conclusion and discusses future work.

2 Related Work

Network administrators commonly use attack graphs when modeling attack scenarios on networks. They model attacks as consecutive steps, represented as nodes within the graph. Modeling them as BAGs adds information about the dependency of the distinct steps and the probability of their successful execution [3, 12]. Attack tree analysis (ATA) and fault tree analysis (FTA), generally used in safety analysis, use a similar modeling approach. Both scientists and engineers commonly use ATA and FTA [2]. RISKEE describes risk propagation within a system, and assesses said risk based on a tree representation [9].

A range of DSE tools considering functional safety or security constraints, in addition to the classical optimization goals, e.g., performance, power consumption, and others, have been presented in recent years [6, 7, 14, 16–18, 20]. A range of these tools focus on the abstract representation of security constraints in the design space, such as restricting the mapping of security vulnerable tasks to processor types with security extensions [16], integration of security functions into system design [20], or securing control loops [10]. In [5], security constraints and mitigation capabilities are introduced based on distinct security levels. Other works consider distinct security problems, e.g., integration of intrusion detection tasks [6], consideration of network security [7], or optimization of communication protocols regarding message authentication [18]. These works cannot directly integrate the attacker's perspective on the system into the DSE. Hence, they do not reflect the effect of security mechanism integration on distinct attack scenarios.

Contrasting, the framework presented in this paper allows the direct representation of security constraints in the form of BAGs and risk trees, allowing the representation of the overall system's attack vulnerability and monetary security risk. Depending on the used modeling approach, the designer directly sees the effect of the system partitioning and task allocation on both security risk and security attack vulnerability. The framework allows the seamless interchange between the risk tree and the BAG representation for describing the security constraints posed on the IoT device / CPS under design. Considering the security performance and power overhead of the distinct solutions allows a detailed assessment of the costs and benefits of particular system designs, including their security attack mitigation capabilities.

3 Proposed Methodology

The framework allows the designer to model the system's functionality, available architecture components, and security attack scenarios using four perspectives, as shown in Figure 1. The work presented in [4] describes a preliminary approach to introducing security attack vulnerability into DSE. In this paper, we present a more elaborate approach, allowing the designers to describe the dependencies of the distinct security assets using rule sets. Furthermore, this paper introduces the usage of risk trees in addition to the BAG based approach. This usage of risk trees allows the framework to perform more detailed modeling of the impacts caused by successfully performed security attacks, shown in Section 4. However, the usage of risk trees induces additional computation time, also described in Section 4. The following paragraphs shortly describe the models behind the distinct perspectives serving as an input to the framework.

System Architecture and Task Representation
A task graph describes the system's functionality with its nodes representing the tasks and the edges modeling the task's dependencies (logical channels). Each task performs operations (OP) on a set of data entities coming with a set of security requirements (SR). High-level hardware components represent the system architecture, including communication buses that connect these components. Each hardware component has security mechanisms (SM) and mitigation capabilities. Each SM comes with a distinct performance overhead and power consumption. For each possible implementation of a task on a hardware component, the designers estimate the implementation's worst-case execution time (WCET).

Security Constraint Representation
The sets for OP, SR, SM, security operations ($SecOp$), and attack types (AT) are defined by the designer. The designer also defines a set of rules stating the relations between: OP, SR and $SecOp$; SR and AT; $SecOp$ and SM. The rules are described by input sets (in), connected with boolean operators, and a resulting output set (out), e.g. $(in_a \vee in_b) \wedge in_c \mapsto out_x$. Given the tasks and their

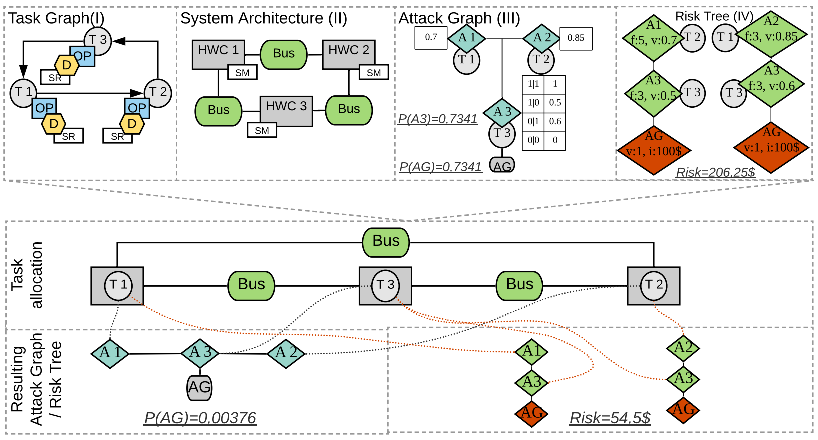


Fig. 1: Framework overview. Inputs consist of: tasks (T) operating (OP) on data entities (D) with security requirements (SR); Hardware components (HWC) connected via communication buses; Attacks modeled as BAGs or risk trees. Outputs consist of: HWC selection and T allocation; security vulnerability (*P(AG)*) and security risk.

$SecOp$, the framework calculates the set of secure communications ($secCom$). A $secCom$ is spanned between two tasks (a source (t_{src}) and a destination task (t_{dst})) performing the same communication securing operations ($secOpComm \in SecOp$) on a particular data entity.

Security Attack View: The framework allows the designer to model cybersecurity threats as BAGs or risk trees. In both methods, each attack (excluding goals) aims at a certain task and comes with a distinct attack type taken from AT. Based on the defined ruleset, each attack type aims at a specific security requirement (defined in SR) of the data handled by the attacked task. Within the BAG, each node represents a distinct attack step, with its leafs describing attack goals. The edges define paths an attacker must traverse to reach an attack goal. Each attack has a distinct success probability provided using a conditional probability table. The attack goals' success probabilities are defined by their marginals in the joint distribution table, calculated by the Bayesian chain rule. Each goal has a maximum allowed success probability defined by the designer.

Risk based Attack Trees: The risk-based method uses RISKEE [9], which is a methodology for risk assessment based on attack trees with the enhancement of also modeling the consequences (impacts) of an attack, and accounting for multiple attacks over time (in the form of attack frequencies) instead of just simulating single events. The key feature of RISKEE is the usage of probability distributions for the estimation of uncertain values (which are inherent in risk assessment), providing a benefit compared to classical single-point estimates, which neglect uncertainties. The mean risk value, which is one of the results returned by RISKEE, is used as a metric for each defined attack goal. Attack goals come with a maximum allowed mean risks defined by the designer. By integrating RISKEE into the framework, we are the first to allow the consideration of risk-based security constraints during the automatic DSE for IoT/CPS.

Security Attack Mitigation: Additionally to the SM, each hardware component defines to what extend said mechanisms are capable of mitigating attacks. This attack mitigation ($m \in \mathbb{R} : m \in [0, 1]$) states the component's defensive capabilities. Assessing the attack mitigation is based on the judgment of the attacker's expertise and available time for breaking said defensive capabilities. Designers can deduce this mitigation capability from security assessments such as Common Criteria (CC)[3], from historical data recording known security incidents, or by expert judgments if no other information is available. The estimated mitigation factor reduces the attack probabilities (BAG) or vulnerabilities (RISKEE) $\lambda, \lambda_m \in \mathbb{R}$ of all attacks on tasks allocated on this particular hardware component, giving the mitigated probability λ_m ($\lambda_m = \lambda * (1 - m)$).

Secure Task Allocation and Partitioning: Based on the system's architecture, functionality, and the given attack scenarios, the framework finds a system partitioning and task allocations which meet the defined security constraints and optimizes either for performance or power consumption. Figure 1 depicts the BAG and RISKEE based approach and the influence of the partitioning and task allocation on the attack success probability and risk value. Hence, the task allocation must comply with a set of restrictions. (I) All tasks directly communicating with each other must be allocated on the same component or different components connected via a communication bus. (II) Each task must map to a hardware component capable of executing its $SecOp$, according to the rules defining the mapping of $SecOp$ to SM. (III) Any task allocation and platform partitioning must fulfill the security attack constraints (in both the BAG- or RISKEE-based security attack modeling approach), meaning that for all attack goals, the defined thresholds on attack success probability or mean risk value must lie within the defined bounds.

Performance and Power Consumption Calculation: The execution times of the individual tasks depend on their component allocations, as each possible implementation of a task on a given component comes with a distinct WCET. Hence, the overall system performance depends on the selected components and the task allocations. The system power consumption consists of the component's static power dissipation and their dynamic power consumption, induced by the task implemented on them. Additionally, each component comes with a distinct security performance and power overhead for each SM. For each $secComm$, the framework adds the performance and power overhead of the SM used by the $secOpComm$ of t_{src} and t_{dst} to the tasks' overall execution times and the component's power consumption, alike. For all tasks performing $SecOp$ not included in any $secComm$, the framework considers the performance and power consumption overheads as well. The $secComm$ must be considered separately, as a task can be both t_{src} and t_{dst} in different $secComm$. Without this consideration, the

[3] https://www.commoncriteriaportal.org/

number of SM executions would not be integrated into the security overhead calculation correctly.o

Optimization of security calculation: The implementation of the framework is based on the open-source *DeSyDe* framework[4]. The framework spends its main computational effort calculating the attack probabilities (ap)/risks for each partitioning and task allocation, as for every new allocation or component selection, the BAG/RISKEE must be recalculated based on the altering attack mitigation. The framework orders the components in descending order according to their mitigation capabilities. In each calculation of the ap/risks, the framework checks if any of the said ap/risks do not fulfill the predefined limits. Upon reaching this break condition, the framework renders all further allocations on components with lesser mitigation capabilities to be insecure. Both the RISKEE and BAG based methods use the same graph structure. Hence, it is feasible to make a comparison between both methods. Opposed to BAGs, in which attack nodes can have multiple parents, the current design of RISKEE only considers single path attack scenarios. Hence, to guarantee a similar structure of the attack scenarios, the framework implements a graph-unwrapping method, turning a BAG into a set of RISKEE trees representing said BAG.

4 Experiments and Results

Using the framework, an use case based on a secure ranging system targeted for the consumer market was revisited. Table 1 describes the security rules defined by the designer to model the security aspects of the use case. The set of OP defines reading (r), writing (w) and storing (s) of data. The set of *SecOp* defines encryption (so_{enc}), authentication (so_{auth}) and secure storage (so_{sst}). The set of SR defines confidentiality ($conf$), authenticity ($auth$) and integrity (int). The set of security mechanisms (SM) defines cryptographic functionalities (sm_{crypt}), task encapsulation (sm_{te}) and tamper safe storage (sm_{tss}). The restriction of $internal$ holds if both t_{src} and t_{dst} of $secComm$ are placed on the same hardware component.

Table 1: Security rules defined to model security aspects of the use case

SecOp derived from OP and SR	AT attacking SR	*SecOp* using SM
$OP, SR \mapsto SecOp$	$AT \mapsto SR$	$SecOp \mapsto SM$
$(r \vee w) \wedge conf \mapsto so_{enc}$	$at_{inf} \mapsto conf$	$so_{enc} \vee so_{auth} \mapsto sm_{crypt}$
$(r \vee w) \wedge auth \mapsto so_{auth}$	$at_{spoof} \mapsto auth$	$(so_{enc} \vee so_{auth}) \wedge internal \mapsto sm_{te}$
$s \wedge (auth \vee conf \vee int) \mapsto so_{sst}$	$at_{tamp} \mapsto int$	$so_{sst} \mapsto sm_{tss}$

The system consists of a ranging node and a ranging anchor. The node authenticates to the anchor using a shared secret (master key) and setting up a secure session (session key). Within this session, node and anchor perform a two way ranging secured by a continually updated ranging key. The node determines its distance to the anchor in a secure way, without comprising its

[4] https://github.com/forsyde/DeSyDe

distance to potentially spying devices, or receiving faked ranging messages from attackers. The functionality consists of two phases, the authentication and the ranging phase, which is described by a task graph comprising 46 nodes. The authentication phase uses an external radio (e.g., Bluetooth Low Energy), the ranging phase uses ultra-wideband. Table 2 lists the security-relevant options for the hardware components available for both the anchor and the node device, giving their estimated performance (Perf) and power consumption (PWC) for their distinct SM. The devices consist of an application processor (AP), a secure element (SE), and a UWB Radio (UR). The security options comprise hardware supported cryptography (HW crypto), side-channel (sc) secured software cryptography library (SW crypto-lib sc sec.), software-based but not tested cryptography (SW crypto functional), Trusted Execution Environment (TEE) and Trust Zone (TZ), secure storage (sec. store), and hardware firewall (HW firewall). Only the SE offers secure storage.

Table 2: Hardware components with security options. Mitigation factor (MF), performance (Perf) given in µs, and power consumption (PWC) in mW

HWC	Security Feature Description	MF	Perf			PWC		
			sm_{crypt}	sm_{tss}	sm_{te}	sm_{crypt}	sm_{tss}	sm_{te}
AP	HW crypto; TEE	0.8	50	/	5	60	/	5
	SW crypto-lib sc sec., TEE	0.7	60	/	5	50	/	5
	SW crypto-lib sc sec.	0.5	40	/	/	50	/	/
	SW crypto functional	0.3	30	/	/	30	/	/
SE	HW crypto, sec store, (EAL 6+)	0.99	500	50	15	60	20	10
	HW crypto, sec store, (EAL 5+)	0.95	500	50	15	60	20	10
	HW crypto, sec store, (EAL 4+)	0.9	500	50	15	60	20	10
UR	HW crypto, TZ, HW firewall	0.8	80	/	15	50	/	10
	HW crypto, TZ	0.7	80	/	5	45	/	5
	HW crypto, 2 separate MCUs	0.85	80	/	20	50	/	10
	SW crypto-lib sc sec., TZ	0.5	160	/	5	90	/	5
	SW crypto functional	0.3	60	/	/	30	/	/

The attacks on the overall system comprise the disclosure of the key material, faking the secure authentication, which builds on a challenge request-response exchange, hijacking the ranging session, and compromising the exchanged ranging frames. Security analysts modeled these attacks using 56 nodes, both for the BAG and the RISK tree. Table 3 lists all security-relevant tasks as identified by modeling the attack scenarios, including their SR and WCETs on the hardware components on which system designers considered their implementations. Confidentiality (c), authenticity (a) and integrity (i) were considered as SR. The assessment of the attack success probabilities of the distinct attack steps for the BAG and the vulnerabilities for the RISKEE based approach were estimated using the Common Vulnerability Scoring System [15], using its *Base Metrics*

We used the described use case as input to the framework and configured it to find the fastest, the most secure, the fastest secure, and most power-efficient and secure solution, both using the BAG and RISKEE based method. The overall system power consumption and performance was normalized. We assume that

Table 3: WCETs of security relevant tasks given in µs.

Device	Task Name	SR	AP	SE	UR
Key	create challenge	c,a	100	150	-
Lock	check challenge	c,a	100	170	120
Key & Lock	derive session key	c,a,i	100	110	
Key & Lock	derive ranging key	c,a	-	190	140
Lock	start session	c,a	80	170	120
Key & Lock	create secure nonce	c,a	-	120	200
Key & Lock	create ranging message	c,a	120	-	-
Lock	calculate distance	c,a	-	350	230

the described system performs distance-based access control. Hence, an attacker breaking the session key temporarily gains access to the secured location and might acquire the authorization to perform further criminal actions. Depending on the secured location, a successful attack might enable the disclosure of secret information, the theft of valuable items, or other critical actions. An attacker who can also disclose the keyless entry system's master key could perform such an attack on multiple locations, depending on the key distribution policy.

Table 4: Most secure and fastest solution.

HWC	Options (most secure)	Options (fastest)
AP (node & anchor)	HW crypto; TEE	SW crypto functional
SE (node & anchor)	EAL 6+	EAL 4+
UR (node & anchor)	HW crypto; 2 separate MCUs	SW crypto functional
avg ap / avg rv	0.0005 / 114.4$	0.016 / 4911$
norm perf.	~2.57	1.0

Based on these considerations and a documented real-life incident[5], risk experts set the impact of disclosing the system's session key to 100.000$, the impact of disclosing the master key to 10.000.000$. This estimation bases on the assumption that with the session key, the attacker can only access one car temporarily. However, with the master key, the attacker might gain access to multiple cars. In this latter case, also the experts considered the reputational damage. They set the frequency for disclosing the session key to 10, and the frequency for the master key disclosure to 5 per year. We modeled these estimated impacts and frequencies in the RISKEE based approach. One must note that the attacks' vulnerabilities and the attack success probabilities are equal for the RISKEE and BAG based approach. We set the maximum allowed risk value of 1.000$ for all attack goals. For the BAG based method, we configured the framework to regard all solutions, in which at least one attack goal's attack success probability exceeds the threshold of 0.002, as insecure. Table 4 describes the fastest, and the most secure system architecture found by the framework. The table shows that the framework can correctly identify optimal solutions based on distinct optimization criteria.

[5] https://www.wired.com/story/hackers-steal-tesla-model-s-seconds-key-fob/

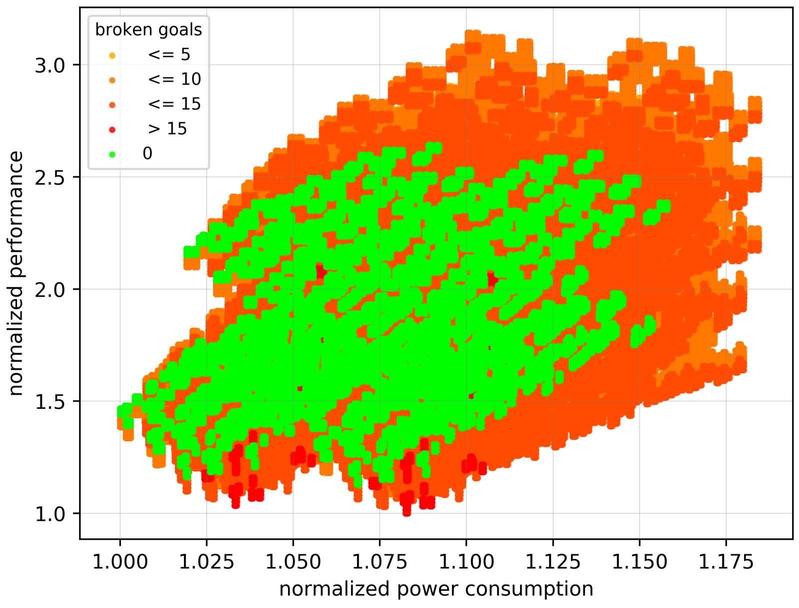


Fig. 2: Solution space identified by the framework using the BAG based method.

Figures 2 and 3 show all solutions found by the framework based on their normalized system performance, power consumption and the number of exceeded security goals for BAG and RISKEE based security constraint calculation, respectively. Both the BAG and the RISKEE based method only consider a small number of solutions to meet their respective security constraints. Both approaches found the same solution space. Out of 5.898.240, the RISKEE based method only considered 320, the BAG 1.643 solutions to be secure. In comparison, the RISKEE method reduced the solution space of a secure solution by another 80.52%. Considering the solutions found using the BAG and the RISKEE based method, one must notice the difference in the selection of options for the distinct hardware components. This difference only comes from the frequency and the impact with which the risk experts considered the attacks on the key material in the RISKEE based approach. The BAG based method does not reflect these two attributes.

Figures 5 and 4 show the numbers of found solutions ordered by their average attack success probability and average mean risk, respectively. One can see that for the BAG based calculation, the majority of the found solutions (41.67%) has an average attack success probability of less than a fourth (~0.0005) of the solution with the highest attack success probability. Considering the RISKEE based calculation, the majority of solutions (64%) identified by the framework lies between 1406$ and 2700$ of the average mean risk value. For both calculation

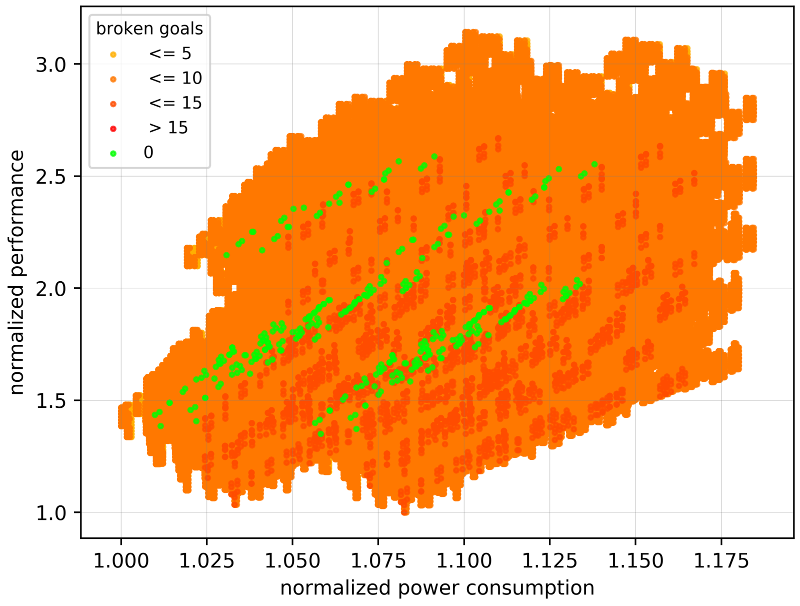


Fig. 3: Solution space identified by the framework using the RISKEE based method.

approaches, the framework found the least number of solutions (1.58% and 0.4% respectively for BAG and RISKEE based approach) in the most insecure fourth considering their average attack success probability/average mean risk.

Table 5 describes the fastest secure solution found by the BAG and RISKEE method. Table 6 the most power-efficient secure solutions, given their average attack probability and average mean risk. One must notice that for both the secure solutions with optimal performance and power consumption, the RISKEE based solution chooses options with higher security attack mitigation capabilities than the BAG based approach, for both the SE and the AP of the anchor and node device. The increased level of security chosen for the SE is due to the high impact, with which the disclosure of the session key and the master key comes. Said impact increases the influence of a successful key disclosure on the average mean risk of the overall system dramatically. A similar result can be seen when considering the most power-efficient and secure solutions, regarding their average attack success probability and mean risk value, respectively. Also, for this optimization criteria, the BAG based method chose less secure options for the SE, but also for the node's AP, compared to the RISKEE based method.

Based on these results, we observed that a risk-based analysis, such as provided by RISKEE, improves the level of detail with which one can model attack scenarios. This higher granularity in the security constraints comes with addi-

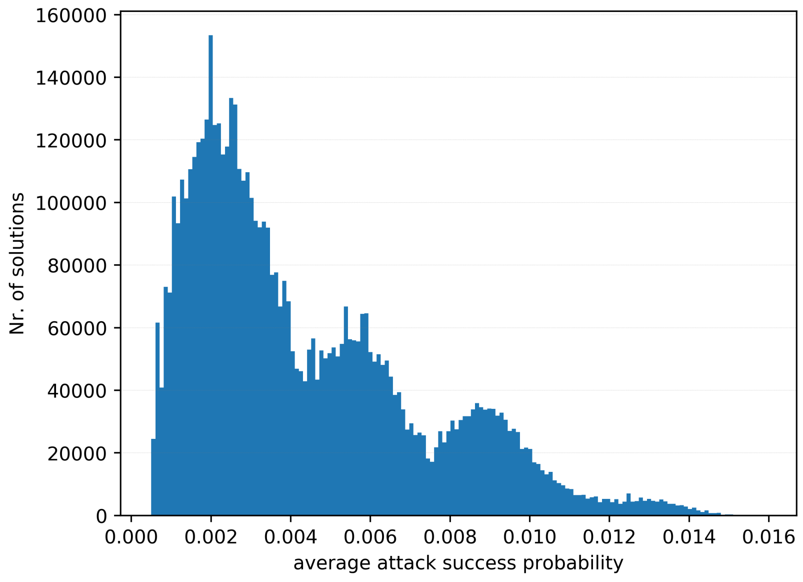


Fig. 4: BAG based solutions found by the framework categorized according to their average attack success probability. Stepsize of $9.95 * 10^{-5}$

Table 5: Fastest secure solutions found based on average attack probability (avg ap), average risk value (avg rv) and performance

HWC	fastest secure (BAG)	fastest secure (RISKEE)
AP (node & anchor)	HW crypto; TEE	HW crypto; TEE
SE (node)	EAL 4+	EAL 6+
SE (anchor)	EAL 4+	EAL 5+
UR	HW Crypto, TZ, HW firewall	HW Crypto, TZ, HW firewall
avg ap / avg rv	0.00069	199.5$
norm. perf.	~1.13	~1.35

tional computational overhead. The use case scenarios were executed on a system comprising 16 GB of RAM and a Intel® Core™ i7-4600U CPU with 2.10 GHz.

Table 7 shows the results of assessing the framework's scalability and the computational overhead of calculating the security constraints using the BAG and RISKEE based methods. We executed both methods with attack graphs comprising 18, 37, and 56 attack nodes (AN), both with and without using the break criteria for the calculation of secure solutions, as described in Section 3. It includes the ratio between the execution times of the full security constraint calculation and the optimized approach, both for the BAG and RISKEE based calculation. For the BAG based method, one must notice that the break criteria can speed up the calculation by $\sim 5\%$ to $\sim 9\%$. For the RISKEE based method, the calculation time is reduced by $\sim 50\%$ to $\sim 70\%$. In general, one can see that the RISKEE based method can capture more details for calculating security constraints. However, its calculation takes ~ 2.5 to ~ 6.3 times longer, when

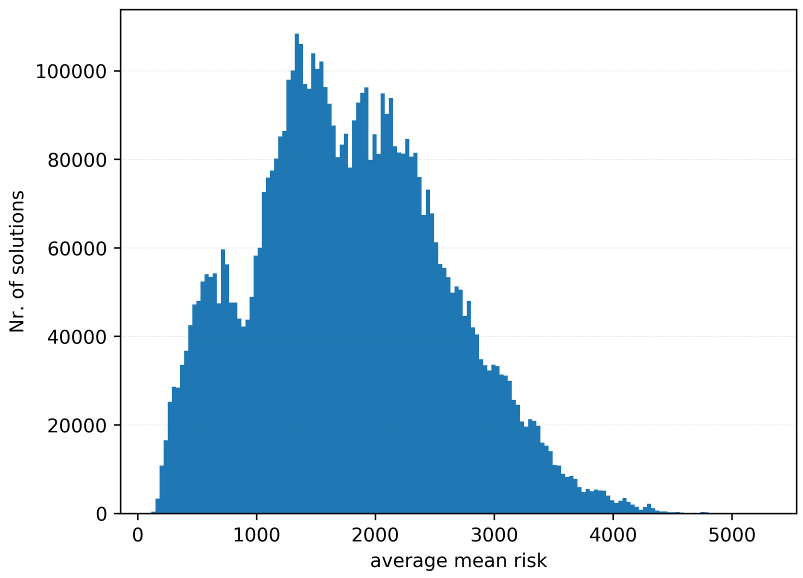


Fig. 5: RISKEE based solutions found by the framework categorized according to their average mean risk. Stepsize of 31

compared to the BAG based method. The higher reduction of the computational overhead for the RISKEE based method comes from the relatively higher risk calculation delay induced by this method. Hence, the more risk calculation the framework can skip, the higher the speedup of the overall calculation becomes. This speedup also shows that the attack probability calculation using the BAGs is much more efficient.

With the consumer device based use case presented in this section, we showed the difference in the BAG and RISKEE based calculation of secure system solutions. We showed that the additional information regarding an attack's impact and frequency, used in the RISKEE based approach, can lead to vastly different results regarding the security constraints. This additional information leads to more time-consuming computation. Considering the maximal calculation time of the RISKEE based method ($\sim$ 6h30min), a more efficient approach must be found. For future work, we will develop a combination of BAG and RISKEE based attack graphs.

5 Conclusion and Future Work

In this paper, we presented a DSE framework, which offers the designers to model cybersecurity threats as BAGs or risk trees. Thereby, the DSE framework automatically calculates a set of security constraints from these modeled security

Table 6: Most power efficient and secure solutions found based on average attack probability (avg ap), average risk value (avg rv) and power consumption (power cons)

HWC	most power eff. secure (BAG)	most power eff. secure (RISKEE)
AP (node)	SW crypto-lib sc sec.; TEE	HW crypto; TEE
AP (anchor)	SW crypto-lib sc sec.; TEE	SW crypto-lib sc sec. TEE
SE (node)	EAL 4+	EAL 6+
SE (anchor)	EAL 4+	EAL 4+
UR (node)	HW Crypto, TZ, HW firewall	HW Crypto, TZ, HW firewall
UR (anchor)	HW Crypto, TZ, HW firewall	HW Crypto, TZ, HW firewall
avg ap / rv	0.00074	198.67$
power cons	~1.014	~1.025

Table 7: Computational overhead for BAG and RISKEE based security constraint calculation for attack graphs with different number of attack nodes(AN)

# of AN	BAG (break)	BAG	RISKEE (break)	RISKEE
18	502s	551s/1.09	2021s	3509s/1.74
37	1943s	2052s/1.05	3315s	5597s/1.69
56	8556s	9337s/1.09	15826s	23670s/1.5

attack scenarios and finds an optimal and secure system partitioning and task allocation, with additional consideration of performance, power consumption, and other constraints. Based on a commercial consumer device use case, we showed the framework's feasibility and the distinct methods' scalabilities.

The approach's main limitation is the source from which to draw the information about the attack success probabilities and the attack frequencies for both BAG and RISKEE based calculation. At the moment, only security expert knowledge serves as input. One must also consider the same limitation for the assessment of the mitigation capabilities of hardware components. No method has yet been published on how to rate a system's ability to withstand security attacks. Hence our assumptions for the component's mitigation capabilities are based on CC certifications. In future work, we will focus on proposing such a method and on a combined calculation utilizing both the BAG and the RISKEE approach within the DSE framework.

Acknowledgment

Project partners are NXP Semiconductors Austria GmbH and the Technical University of Graz. This work was supported by the Austrian Research Promotion Agency (FFG) within the project UBSmart (project number: 859475).

References

1. Al-mhiqani, M.N., et al.: Cyber-Security Incidents : A Review Cases in Cyber-Physical Systems. International Journal of Advanced Computer Science and Applications **9**(1) (2018)

2. Ammann, P., et al.: Scalable, graph-based network vulnerability analysis. In: Proc. of the 9th ACM Conf. on Computer and Communications Security (2002)
3. Feng, N., et al.: A security risk analysis model for information systems: Causal relationships of risk factors and vulnerability propagation analysis. Information Sciences (2014)
4. Gressl, L., et al.: Consideration of Security Attacks in the Design Space Exploration of Embedded Systems. In: 2019 22nd Euromicro Conf. on Digital System Design (DSD) (2019)
5. Gressl, L., et al.: A Security Aware Design Space Exploration Framework. In: Proc. of the 14th Intern. Conf. on Systems ICONS 2019. ThinkMind(TM) Digital Library (2019)
6. Hasan, M., et al.: A design-space exploration for allocating security tasks in multicore real-Time systems. Proc. of the 2018 Design, Automation and Test in Europe Conference and Exhibition, DATE 2018 (2018)
7. Kang, E.: Design Space Exploration for Security. IEEE Cybersecurity Development Design (2016)
8. Knerr, B.: Heuristic Optimisation Methods for System Partitioning in HW / SW Co-Design. Ph.D. thesis, Vienna University of Technology (2008)
9. Krisper, M., et al.: RISKEE : A Risk-Tree Based Method for Assessing Risk in Cyber Security. In: Proc. of EuroSPI 2019: European System, Software & Service Process Improvement & Innovation (2019)
10. Li, L.W., et al.: Security-aware Modeling and Analysis for HW/SW Partitioning. Proc. of the 5th Int. Conf. on Model-Driven Engineering and Software Development (2017)
11. Nasser, M., et al.: Cyber-Security Incidents: A Review Cases in Cyber-Physical Systems. Int. Journal of Advanced Computer Science and Applications **9** (2018)
12. Poolsappasit, N., et al.: Dynamic Security Risk Management Using Bayesian Attack Graphs. IEEE Transactions on Dependable and Secure Computing (2012)
13. Rosvall, K., et al.: Exploring power and throughput for dataflow applications on predictable NoC multiprocessors. Proc. - 21st Euromicro Conf. on Digital System Design, DSD 2018 (2018). https://doi.org/10.1109/DSD.2018.00011
14. Roudier, Y., Apvrille, L.: SysML-Sec - A Model Driven Approach for Designing Safe and Secure Systems. Proc. of the 3rd Int. Conf. on Model-Driven Engineering and Software Development (2015)
15. Schiffman, M.: Common Vulnerability Scoring System (CVSS). URL https://www.first.org/cvss/v3.1/specification-document (2019)
16. Stierand, I., et al.: Integrating the security aspect into design space exploration of embedded systems. Proc. of IEEE 25th Int. Symp. on Software Reliability Engineering Workshops, ISSREW 2014 (2014)
17. Tamas-Selicean, D., Pop, P.: Design Optimization of Mixed-Criticality Real-Time Embedded Systems. ACM Transactions on Embedded Computing Systems **14**(3) (2015)
18. Xie, Y., et al.: Security/Timing-aware Design Space Exploration of CAN FD for Automotive Cyber-Physical Systems. IEEE Transactions on Industrial Informatics (2018)
19. Yu, R., et al.: Deploying robust security in internet of things. 2018 IEEE Conf. on Communications and Network Security, CNS 2018 (2018)
20. Zheng, B., et al.: Cross-Layer Codesign for Secure Cyber-Physical Systems. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (2016)

A Design Exploration Framework for Secure IoT-Systems

Lukas Gressl[*], Alexander Rech[*], Christian Steger[*], Andreas Sinnhofer[†] and Ralph Weissnegger[‡]
[*]Graz University of Technology
Institute of Technical Informatics, Inffeldgasse 16, A-8010 Graz
Email: {gressl}{rech}{steger}@tugraz.at
[†]NXP Semiconductors Austria GmbH
Mikron-Weg 1, A-8101 Gratkorn
Email: andreas.daniel.sinnhofer@nxp.com
[‡]CISC Semiconductor GmbH
Burgring 18, A-8010 Graz
Email: r.weissnegger@cisc.at

***Abstract*—Cybersecurity is vital for embedded systems, especially for Internet of Things (IoT) systems. IoT systems have become essential in our daily lives, as they are usable for various application areas. They are usually small, connected with other systems, and perform a wide range of tasks. They are subject to multiple constraints in terms of performance, power consumption, chip area, etc. Attackers often target such devices as they are in constant interaction with each other or connected to the internet during private data processing. Cybersecurity, thus, plays a vital role in the design of IoT systems. Hence, designing secure IoT systems is a complex task, particularly for designers with limited security know-how. Security measures increase both computation time and power consumption, creating a conflict between these constraints, which must be solved by the designers. Balancing these constraints is a highly complex task. In this paper, we propose a new approach for considering security constraints in design space exploration, including possible security attacks on embedded systems. The method simplifies the consideration of security requirements at the start of the system design. We introduce a security attack based design space exploration framework, capable of finding the optimal design for an IoT system, based on its architectural, behavioral, and security attack description. The paper shows the feasibility and benefits of the framework, employing a secure sensor use case.**

***Index Terms*—Cyber Security; Embedded System Design; IoT Systems; Early System Design;**

I. Introduction

The advent of Cyber-Physical Systems (CPS) and the Internet of Things (IoT) with their utilization in the industry opened a new attack surface for cybersecurity attacks. In the last decade, scientists reported numerous cybersecurity attacks on industrial facilities [1], [2]. This trend emphasizes the need for considering cybersecurity attacks for CPS and IoT systems. Security mechanisms, such as cryptographic algorithms, cause additional performance overhead and power consumption on the overall system. As industrial use cases demand fast response times, and high energy efficiency on the used CPS [3], system designers must consider the optimal selection of security mechanisms throughout the whole design phase. With these various and often opposing constraints, finding an optimal system design becomes a complex task. Classic Design Space Exploration (DSE) tools help designers to find the optimal system partitioning and task scheduling while considering power consumption and performance [4]–[6]. Advanced DSE frameworks also offer the consideration of security or safety constraints to a limited extent [7]–[12].

In this paper, we present a new approach for considering security constraints in DSE. The framework we present in this paper finds an optimal system partitioning and task scheduling based on four inputs. (I) A graph-based description of the system functionality. (II) A description of suitable architectural components with their performance, power, security characteristics, etc. (III) The security attack view contains various possible attack scenarios described as Bayesian Attack Graphs (BAGs). (IV) Security options available to the system to counter the potential attacks. With these inputs, the framework finds the solution with the lowest probability of successful attacks, or the most power-efficient or fastest solution not exceeding a predefined attack success probability. With our approach, we make the following contributions. (I) The framework presented here is the first to consider security constraints based on BAGs during DSE, automatically suggesting security mitigation techniques and key placement. (II) We show the feasibility of the framework based on an industrial use case on multiple layers, considering the design of a secure sensor module, a control gateway, and an analysis platform. (III) based on the results of the industrial use case, we suggest components for a real-world implementation of the system.

The paper is structured as follows: Section II discusses related work; Section III describes the modeling approach and the framework's implementation; Section IV discusses the use case; Section V concludes this paper and gives an outlook on future work.

II. Related Work

Considering security attack scenarios in the design phase of systems has been well studied in the field of network security solutions. These modeling approaches use the attacker's view and formalize security attacks as graphs. The nodes in such attack graphs represent distinct attack steps. The edges represent the attack steps' sequences and dependencies. By attaching a success probability to each attack step in the graph, a BAG can be formalized. The framework uses these BAGs to calculate the success probability of the overall attack scenario. Such BAGs build on expert knowledge or on historical data describing security incidents [13], [14].

In addition to the standard DSE tools which focus on performance, power consumption, system area, costs, etc., a range of industrial and scientific projects have been considering functional safety [15] and security constraints [7]–[11], [16] during first stage system design. The DSE tools focusing on security requirements either abstractly introduce them or solve a particular problem at hand. DSE approaches, which consider one distinct problem, such as the integration of security surveillance tasks [8], throughput optimization for authenticated messages on distinct bus systems [7], or the integration of network security schemes [9], do not take into account the whole design space of the overall system, as opposed to the framework presented in here. Tools introducing abstract security considerations during DSE put their focus on the overall system design. These frameworks introduce security constraints by preventing the mapping of security demanding tasks to insecure hardware [10], introducing security mechanisms into industrial control loops [11], or mitigating security attacks by including security functions at specific points of the system design [12].

In contrast to the related projects described in this section, the framework presented in this paper introduces security constraints into automatic DSE based on attack scenarios formulated as BAGs. Based on these attack scenarios, the framework selects security functions capable of mitigating the attacks, chooses the hardware components supporting these security functions, and selects the right security keys for the cryptographic algorithms. Thereby, the framework considers the secure placement of used keys and the additional computational overhead, power consumption, and costs induced by the security functions. The framework introduces security attacks and mitigation techniques on a detailed level, without losing the overall view on the overall design space. It supports designers in their decisions on what functionality to implement on which architectural component and calculates its impact on the security attack success probability based on the BAGs.

III. Framework Design and Implementation

The framework's design consists of several perspectives, with which the designer can describe the system under design. These perspectives and their integration are visualized in Figure 1. The framework's design builds on the work presented in [17]. However, the design proposed here introduces more modeling flexibility using security rules and spanning secure channels between tasks, thus, making the framework usable on multiple design layers for IoT systems. This section explains the concept after which the framework was implemented, providing details on the computationally exhaustive problem the framework aims to solve, as well as its implementation.

Task Graph (I) and System Architecture (II): The designers describe the system's functionality as a task graph, representing the tasks as graph nodes and their interactions by the graph's edges. Each task performs a set of operations on a range of data entities, which come with distinct security requirements (SR). This description allows a more accurate modeling of the functionality. The system's architecture is described by hardware components connected by communication buses. Both hardware components and communication buses are characterized by needed chip-area, costs, power consumption, etc. Each hardware component is described by a set of security capabilities ($secCap$) defining the components' ability to support, for example, cryptographic algorithms, task encapsulation, etc. The components' vulnerability risk (VR) describes the level with which the implementations of these $secCap$ have been verified. It defines the probability of an attacker finding a weak spot in the security function's implementation, enabling him to bypass it. The worst-case execution time (WCET) describes the performance of a task implemented on a distinct hardware component [4]. Furthermore, each hardware component maps to a predefined device. This device association determines the key placement.

Attack Graph (III) and Security Functions (IV): Network security analysis widely employs security attack graphs. Each node in the attack graph describes a distinct security attack step, which might result from a preceding step and on which a succeeding attack might follow. Adding the success probability to each attack step allows the formulation of the attack graph as a BAG. The BAG calculates the probability of an attacker reaching its leaves using the Bayesian chain rule. These BAG leaves are defined as attack goals and assigned a threshold that determines the maximum probability with which the attacker might reach it. Each attack node in the BAG aims at a victim task in the task graph. An attack-type (at) further characterizes each attack node. Attack goals may have a victim task and an at. The at defines which SR of the victim task the attack aims at, considering the data entities the task operates on [14], [18]–[20]. The victim's SR stem from a STRIDE analysis [21].

The security functions ($secFunc$) describe the set of security options available to the DSE framework. Each $secFunc$ is described by its security type. This type maps the $secFunc$ to the at it can mitigate. For example, encryption mitigates attacks aiming at the data's confidentiality; authentication prevents attacks compromising the data's authenticity, etc. Designers must also take into consideration that $secFunc$ themselves are targets for potential attackers. $secFunc$ using cryptographic algorithms use secret keys. A key might derive another key. Key disclosure attacks potentially expose these keys. The framework reflects the key derivation in the hierarchy of the key disclosure attacks, as breaking a key's parent key renders the derived key insecure. Adding the key's disclosure attack as a parent to the security function's attack reflects the dependency of a security function's soundness

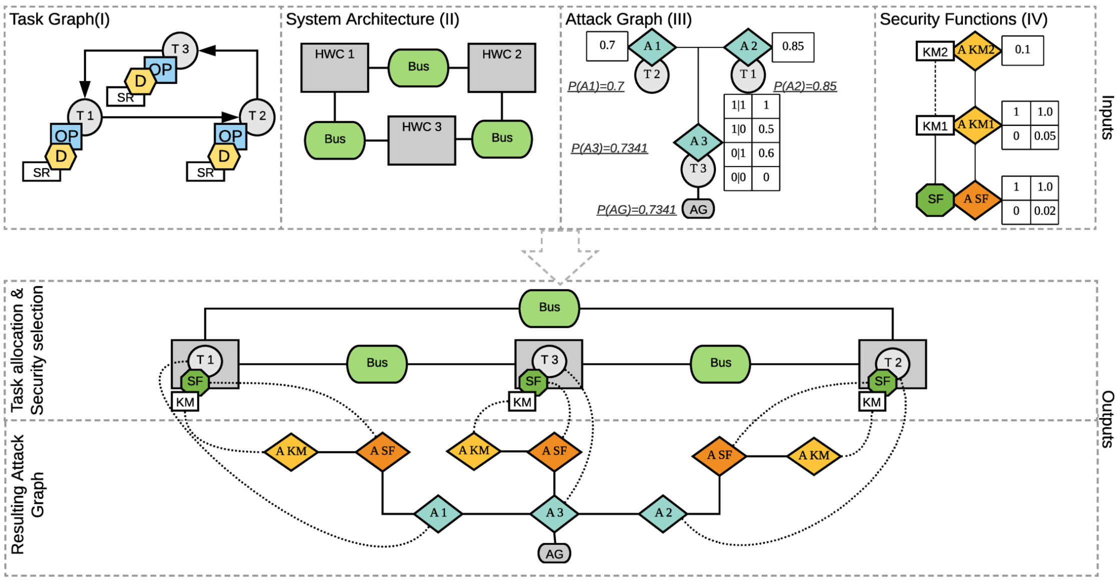


Fig. 1: The four perspectives from which the system under design is described. From these perspectives the framework tries to find a secure platform partitioning and task allocation with optimal performance or power, depending on the designer's optimization goal. Tasks (t) operate (op) on data entities (D). Attacks (A) lead to an attack goal (AG). Security functions ($secFunc$) use keys (KM), both targets to attacks.

on its used key. Cryptography is such a security function. Depending on the key's validity time and its utilization, the designer defines the probability of a key disclosure attack. A master key valid over the whole product's lifetime (long term) is a higher valued attack goal than a session key only valid for a short period (short term). We use the Common Vulnerability Scoring System (CVSS) to assess the success probabilities of all the attacks aiming at tasks, security functions, and keys. Thereby we rely on *Base Metrics* of the CVSS [22].

Task allocation and selection of security functions: The framework uses the inputs of the four perspectives to select the platform components, allocates the tasks, and selects the $secFunc$ to secure the data entities from potential attackers. Depending on the configuration, the framework finds the solution with the best performance, the lowest power consumption, with or without security consideration. The general allocation of a task (t) on a hardware component (hwc) uses the WCETs characterizing the execution time of said t allocated on the hwc. The framework also ensures the general reachability of the task allocations. The framework calculates the security constraints of the system under design based on the data entities' SR, the tasks' operations (op) on said data entities, and the attack scenarios modeled by the BAGs.

In a first step, the framework calculates the security allocation constraints by determining the security operations ($secOp$) a task must execute to ensure its data entities' SR, depending on its op. Based on these $secOp$, the framework defines what communication channels ($secChan = (t_{src}, t_{dst})$) between tasks need to be secured. A $secChan$ is spanned between a source task (t_{src}) and a destination task (t_{dst}), where both tasks perform the same $secOp$ on a distinct data entity. Such a $secChan$ can be internal (int), if both tasks are mapped to the same hwc, or external (ext), if they are mapped to different hwc. The dependencies between op, SR, $secOp$, etc. are defined by the system designers and provided to the framework in form of rules. These rules consist of inputs combined with Boolean operators, mapping to outputs (for example $in_a \wedge in_b \mapsto out_c$). Table I lists the rules applied for the case study presented in here. The op comprise writing (w), reading (r) and storing (s); SR comprise confidentiality (sr_c), authenticity (sr_a) and integrity (sr_i); $secOp$ comprise encryption (so_e), authentication (so_a) and tamper safe storage (so_t); attack types comprise information disclosure (at_i), spoofing (at_s) and tampering (at_t); $secFunc$ comprise the usage of cryptographic algorithms (sf_{crypt}), task encapsulation (sf_{te}) and secure storage (sf_{ss}); $secCap$ comprise the support of cryptography (sc_{crypt}), task encapsulation (sc_{te}) and the tamper safe storage of data (sc_{tss}).

Resulting attack graph: From each resulting mapping (hwc selection and task allocation) rendered to fulfill the security allocation constraints, the framework builds the BAG graph, considering the additional attacks on single $secFunc$ and the used keys. For each mapping, the framework determines the possible keys used for securing the distinct $secChan$. The hwc selection, as well as the key's validity time and device

TABLE I: Security rules used in the use case study

$op, sr \mapsto secOp$	$at \mapsto sr$
$(r \vee w) \wedge sr_c \mapsto so_e$	$at_i \mapsto sr_c$
$(r \vee w) \wedge sr_a \mapsto so_a$	$at_s \mapsto sr_a$
$s \wedge (sr_c \vee sr_a \vee sr_i) \mapsto so_t$	$at_t \mapsto sr_i$
$secOp \mapsto secFunc$	$secFunc \mapsto secCap$
$(so_e \vee so_a) \wedge ext \mapsto sf_{crypt}$	$sf_{crypt} \mapsto sc_{crypt}$
$(so_e \vee so_a) \wedge int \mapsto sf_{te}$	$sf_{te} \mapsto sc_{te}$
$so_t \mapsto sf_{ss}$	$sf_{tss} \mapsto sc_{tss}$

restriction, determine the utilization of the keys. The device restriction makes said keys usable only for $secChan$ with their t_{src} and t_{dst} allocated on a hwc belonging to the same device. The validity time further restricts the utilization of the keys. Only $secChan$ of which both t_{src} and t_{dst} map to hwc supporting sc_{tss} can use long term keys. The same restriction holds for short term keys derived from long term keys. If derived from long term keys, both devices allocating t_{src} and t_{dst} must contain an hwc supporting sc_{tss} and a secure connection must be possible between this hwc and the hwc allocating t_{src} and t_{dst}, respectively.

The framework uses the selected keys and $secFunc$ combinations to add the respective attack nodes to the original BAG. Thereby, the framework adds the attacks aiming at the $secFunc$ as parents to the attacks targeting the tasks using them. It extends the original attack's conditional distribution table in such a way that the potential attacker can perform this original attack only after successfully attacking the added $secFunc$ attacks. The framework only adds security function attacks as parents, if the security function is capable of mitigating the initial attack. It adds the attack aiming at the key used by the security function as a parent to the attacks on said $secFunc$. This extension modifies the conditional distribution table of the security function's attack node in such a way that a successful attack on the used key also breaks the security function itself. The framework adds the attack on the parent key to the attack of the derived key in the same way. Depending on the task mapping, the VR of the allocating hwc reduces the attack success probabilities of both attacks on the $secFunc$ and keys, multiplying the conditional probabilities with VR. The resulting BAG is used to calculate the unconditional success probabilities of the attack goals. The framework marks the mapping as secure if the success probabilities of all attack goals are below their respective thresholds. Otherwise, it marks it as insecure.

Framework implementation: The framework is implemented based on the *DeSyDe* framework publicly available at *Github*[1]. The *DeSyDe* framework finds the performance or power optimal solution for a platform selection and task allocation. It uses a constraint programming approach, delivering exact solutions. The extensions described in this paper introduce security constraints, which reduce the design space for finding the performance/power optimal solution upfront. The calculation of the security constraints is computationally expensive as the framework must perform a recalculation for every possible mapping of an attacked task to hwc. Hence, the framework must perform $O(n^{(m*k)})$ recalculations, with n being the number of attacked tasks, m being the number of hwc, and k being the number of keys usable to the supported $secFunc$.

The framework aims at reducing the number of recalculations. It orders the hwc according to their VR and the available keys considering their unconditional attack probability, in an ascending way. Using these ordered sets, the framework checks, after each permutation of task mapping and key usage, if all attack goal thresholds hold. If all goals exceed their thresholds, the framework skips the attack goal calculation for all succeeding mappings and key usages, respectively, as no further secure solutions are to be expected. Section IV shows the impact of using these break criteria.

The performance calculation for a solution is based on the WCETs of the tasks allocated on the selected hardware components ($wcet^t_{hwc}$). To these WCETs, the framework adds the bus delay caused by the data entities sent over the communication buses. The data entity's size (len_d) is multiplied with the transmission delay ($delay_{tx}$) of the used bus ($delay_{comm} = len_d * delay_{tx}$). The security overheads are either given as overall execution time, or as a delay per bit of secured data. These security overheads ($delay_{sec}$) of the used hardware components are added to the overall performance delay, based on the used $secFunc$. The overall system delay is the sum of the communication delay, the security delay, and the tasks' WCETs ($delay_{sys} = \sum delay_{comm} + \sum delay_{sec} + \sum wcet^t_{hwc}$). The power consumption is calculated using static pwr_{stat} and dynamic power (pwr_{dyn}) consumption. The pwr_{dyn} depends on the WCETs of the tasks executed on the distinct hwc. Furthermore, the framework adds the power needed by the $secFunc$ (pwr_{sec}), and consumed power for each bus transmission ($pwr_{sys} = \sum pwr_{dyn} * wcet^t_{hwc} + \sum pwr_{stat} + \sum pwr_{sec}$). The influence of the delay and additional power consumption induced by the various $secCap$ is shown in Section IV.

IV. Usecase Study

In the use-case study, an industrial sensor system was designed considering different possibilities for the system realization. The system comprises a sensor device, a sensor controller gateway, and an analysis server (AS). The sensor comes with a sensing module, an optional microcontroller (MCU), a Bluetooth Low Energy (BLE) radio, and a secure element (SE). The sensor's functionality implemented on the MCU can also be entirely realized on the SE. The sensor controller gateway consists of an MCU, a BLE chip, a WiFi radio, and an optional SE. The data analysis is performed on a server, either implemented on a Hardware Secure Module (HSM), a commonly used server with HSM support (HSM supp.), or a server with limited security extension (lim.-sec.), only offering software-based cryptography (SWC). The HSM offers hardware-based cryptography (HWC), a task encapsulation (te) in the form of a trust-zone (TZ), and a tamper safe storage (tss). The SE offers HWC, TZ, and tss, either with the

[1]https://github.com/forsyde/DeSyDe

TABLE II: The different options to use for the SE, the MCU, and the server with their security overhead.

HWC	Sec. Op.	VR	Security Overhead
SE	EAL5+	0.2	tss(40Mb/s), te (10µs)
			sym(500Mb/s), asym(120ms)
	EAL6+	0.1	tss(20Mb/s), te (10µs)
			sym(400Mb/s), asym(150ms)
MCU	sec.-enh.	0.25	te (10µs)
			sym(600Mb/s), asym(150ms)
	lim.-sec.	0.4	sym(1.2Gb/s), te(5µs)
Server	HSM	0.1	tss (20Mb/s), te (10µs)
			sym(800Mb/s), asym(150ms)
	HSM supp.	0.2	tss (20Mb/s), te (5µs)
			sym(1Gb/s), asym(120ms)
	lim.-sec.	0.75	sym(1.2Gb/s), te(5µs)

TABLE III: The different options to use for the SE and the MCU with their power consumption.

HWC	Sec. Op.	Power Consumption
SE	EAL5+	tss(30mW), te(5mW), sym(60mW), asym(300mw)
	EAL6+	tss(30mW), te(5mW), sym(70mW), asym(400mw)
MCU	sec.-enh.	te(5mW), sym(50mW) asym(250mW)
	lim.-sec.	sym(35mW), te(5mW)
Server	HSM	tss(50mW), te(20mW), sym(120mW), asym(600mw)
	HSM supp.	tss(50mW), te(10mW), sym(100mW), asym(600mw)
	lim.-sec.	te(5mW), sym(80mW)

Common Criteria[2] evaluation assurance level (EAL) 5+ or 6+. The sensor's MCU comes with a security enhancement (sec.-enh.) in the form of an HWC and a TZ or with lim.-sec. The lim.-sec. comprises cryptography realized as SWC and HWC (both realizations come without side-channel protection) and a TZ. All other variants of HWC are side-channel protected. The different variants are listen in table II and table III. In both tables, the security overheads and power consumption were estimated based on [23]–[25].

The functionality of the overall system setup consists of different phases executed consequently:

1) *Configuration phase:* During the configuration phase, the sensor is configured and activated via the sensor controller gateway.
2) *Sensor data accumulation phase:* In the data accumulation phase, the sensor gathers information and sends it to the sensor controller gateway, which filters the received sensor information and executes application-dependent operations.
3) *Data analysis and monitoring phase:* In the data analysis phase, the gateway sends the filtered data to the analysis server, which processes and monitors the data.

A task graph consisting of 27 tasks describes these phases. Attacks on the system encompass the compromising of the sensor configuration, the faking of the sensor activation message, the manipulation of the accumulated sensor data, the faking of the filtered data, and the disclosure of the analyzed data.

The framework selects from a range of $secFunc$ and secret keys to mitigate these attack scenarios. Table IV lists the

[2]https://www.commoncriteriaportal.org/

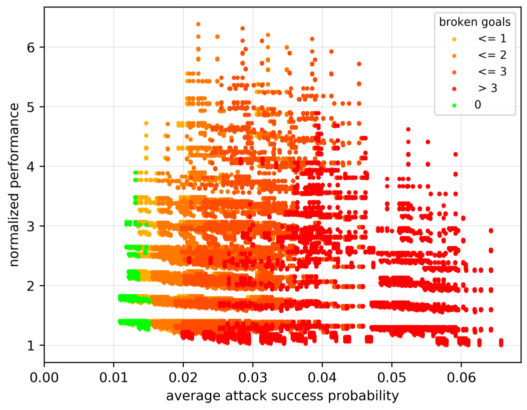
TABLE IV: Security functions and key material used by the system. The attack success probability reflects the attacker's motivation.

Security Function	Key	Attack Success Probability
Cryptography	Master Key	0.4
	Binding Key	0.25
	Session Key	0.1
	Certificate	0.4
Task Encapsulation	/	0.25
Secure Storage	/	0.25

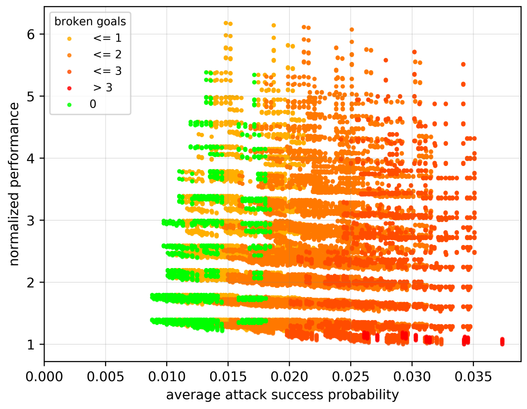
available $secFunc$, including the probability of them being successfully attacked (AP), based on the secret keys they use. The keys comprise (i) a symmetric master key (smk) with AP of 0.25, (ii) a binding key (sbk) with AP of 0.15, (iii) a session key (ssk) with AP of 0.1, derived from the smk, and (iv) a certificate ($cert$) used for asymmetric cryptography with an AP of 0.25. The AP for both security functionality (Table IV) and the secret keys were estimated based on assuming that neither the security function nor the keys reside on a component offering any means of attack mitigation. Said mitigation depends on the capability of the hardware component implementing the security functionality. The used keys and the information about what the attacker gains by the attack further influence the AP, and, hence, the attacker's motivation. Naturally, the more impact a key's disclosure has, the greater the attacker's motivation, and, hence, the disclosure's attack success probability. For example, successfully breaking encryption based on a long term key has a greater impact than breaking a short term key-based encryption, simply caused by the keys temporal validity. The same assumptions were made when estimating the APs on the disclosure of the secret keys themselves.

To better show the influence of the secure data handling on the found solutions, an alternate use case, in which the sensor controller gateway forwards the unfiltered sensor data, was evaluated. This alternative use case is based on the same platform components, $secFunc$ and key material as the base use case. Its only difference is, that the sensor gateway forwards the received sensor data, instead of filtering it. Hence, the sensor gateway does not perform any means of security mechanisms on the received data. The impact caused by this additional security handling is explained in the next paragraph.

Figures 2a shows the solutions found by the framework for the use case in which the gateway filters the sensor data before being sent to the analysis server (use case I). Figure 2b shows the solutions for the use case in which the sensor data is simply forwarded (use case II). For use case I, the framework identifies more insecure solutions regarding the number of broken goals. Additionally, the solutions found to be secure have slower overall performance than compared to the solutions found for use case II. The filtering of the sensor data necessitates the secure handling of the sensor data by the sensor controller gateway in use case I, inducing additional security overhead. Figure 3 shows the influence of the additional security handling induced by filtering the sensor data on the controller gateway on the number of



(a) All solutions found for use case I according to their normalized performance and their average attack success probability, colored according to the number of exceeded attack goal thresholds.



(b) All solutions found for use case II according to their normalized performance and their average attack success probability, colored according to the number of exceeded attack goal thresholds.

Fig. 2: Solutions found for the two variants of the secure sensor use case.

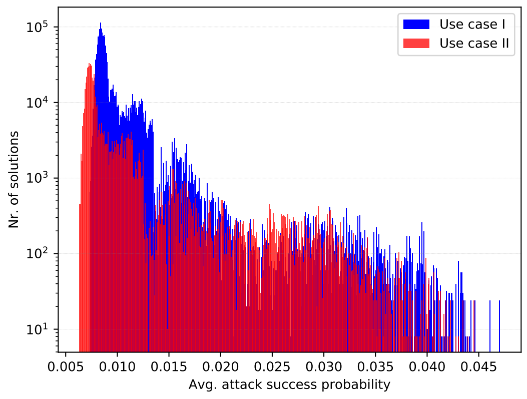


Fig. 3: Number of solutions categorized by average attack success probability for use case I and II. Stepsize of $8.1 * 10^{-6}$

TABLE V: Fastest and most secure solutions found based on average attack probability (avg ap) and execution time normalized to the system with the fastest execution for use case I.

Device	HWC	Most secure	Fastest
Sensor	MCU	/	lim.-sec.
	SE	EAL 6+	EAL 5+
Gateway	MCU	sec.enh.	lim.-sec.
	SE	EAL 6+	EAL 5+
AS	Server	HSM	lim.-sec.
	avg ap	0.011	0.066
	norm delay	~1.4	1.0
	norm pwr.	~1.4	~1.31
Key Placement			
Sensor	MCU	/	*ssk*, *sbk*
	SE	*ssk*, *smk*, *cert*	*sbk*, *smk*, *cert*
Gateway	MCU	*ssk*, *sbk*	*ssk*, *sbk*
	SE	*sbk*, *smk*, *cert*	*sbk*, *smk*, *cert*
AS	Server	*ssk*, *smk*, *cert*	*ssk*, *smk*, *cert*

found solutions and their average attack success probabilities. For use case II, the framework finds solutions with a lower average attack success probability. This effect is caused by the additional attack surface when filtering the sensor data on the sensor controller gateway in use case I, which requires the decryption of the sensor data and the encryption of the filtered sensor data. This additional step opens another attack surface that needs to be mitigated by the system. The smaller number of found solutions in use case II is caused by the missing tasks describing the sensor data filtering process, which reduces the number of possible task mappings.

Table V shows the most secure system configuration, and the system configuration with the highest performance for use case I. For the most secure system realization, the framework places all security-relevant tasks of the sensor on the SE, none are implemented on the MCU. The fastest solution uses the components with the fastest (but most vulnerable) security mechanisms. Both solutions use the ssk for securing the exchanged data. Furthermore, the fastest solution relies on using the sbk for securing the communication within the single devices (sensor, sensor controller gateway, server platform). The system configuration with optimal performance puts as many tasks on the same hardware component as possible. Hence, this solution uses task encapsulation to secure the intra-component-communication, saving security overhead for cryptographic algorithms.

Table VI lists the options selected for the hardware components to get a performance optimal and secure solution for use cases I and II. The main difference lies in the selection of the SE for both sensor and sensor controller gateway. In use case II, the controller gateway does not filter the sensor data. Hence, the framework selects a SE with higher vulnerability risk (but faster security mechanisms) than for use case I. Both solutions

TABLE VI: Fastest secure solutions found based on average attack probability (avg ap) and performance normalized to the system with the highest performance for the use case I and II.

Device	HWC	Use Case I	Use Case II
Sensor	MCU	sec.-enh.	sec.-enh.
	SE	EAL 6+	EAL 6+
Gateway	MCU	sec.-enh.	sec.-enh.
	SE	EAL 6+	EAL 5+
AS	Server	HSM supp.	HSM supp.
	avg ap	0.015	0.014
	norm delay	~1.26	~1.22
	norm pwr.	~1.39	~1.28
Key Placement			
Sensor	MCU	ssk, sbk	ssk, sbk
	SE	sbk, smk, $cert$	sbk, smk, $cert$
Gateway	MCU	ssk, sbk	ssk, sbk
	SE	sbk, smk, $cert$	sbk, smk, $cert$
AS	Server	ssk, smk, $cert$	ssk, smk, $cert$

for use cases I and II rely on the usage of ssk for securing the communication between sensor, gateway, and server. The framework was executed on a system comprising 16 GB of RAM and a Intel® Core™ i7-4600U CPU with 2.10 GHz. The computational overhead for finding all secure solutions was reduced from ~2h56min to ~1h32min for use case I, when using the break criteria based optimization. Hence, using the break criteria saved ~47% of execution time. For use case II the break criteria saved ~35% of execution time, reducing the calculation from ~2h34min to ~1h40min.

Potential System Realization:

A possible system solution, including the description of its components, the hardware, and the communication protocols, is discussed based on the three system setup phases presented before. The system realization further extends the application setup to meet the requirements of a typical secure sensor use case implemented within a climate control system. The use case includes the measurement of temperature values, the task of controlling actuators (heating and cooling system), and the analysis of the data collected on the cloud level. Fig. 4 gives an overview of the overall setup.

In our scenario, one or several sensor devices monitor a certain area. They constantly measure the temperature, encrypt the data, and forward it wirelessly to a nearby sensor control gateway. Bluetooth Low Energy (BLE) could be used as a short-range wireless protocol to enable communication between sensor devices and the gateway. BLE benefits IoT applications, especially due to its power-saving design and robustness against obstacles. Additionally, BLE features different security options on the link layer. Examples are the Elliptic Curve Diffie-Hellmann P-256 (ECDH) key exchange algorithm and AES-CCM 128-bit encryption utilized during the pairing procedure of two devices. After the gateway received the data packages, they are processed and interpreted by it. Depending on the actual value of the temperature readings, corresponding actuators can be triggered, if the temperature falls/rises below/above a certain threshold, the heating/cooling system is turned on or off. Eventually, the data is uploaded to the analysis server by the gateway. HTTPs offers a secure way for this upload. The connection remains private due to the transport layer security (TLS) protocol. In this sense, the protocol provides data integrity and confidentiality via symmetric encryption (DES, AES, etc.) and ensures authenticity through asymmetric cryptography (RSA, ECC in combination with PKI). A possible hardware setup that is compatible with the described distributed application, as well as the mentioned wireless protocols are listed in the following:

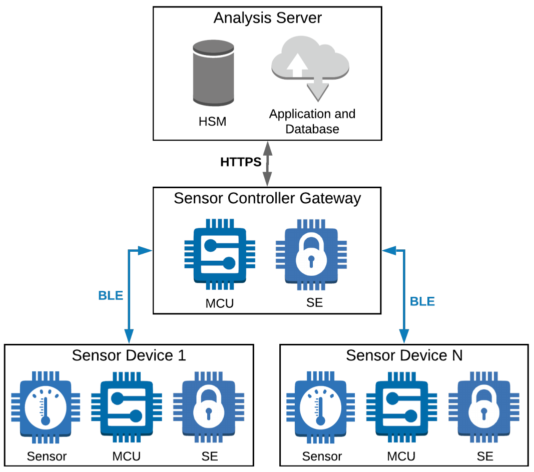


Fig. 4: Possible system setup consisting of one or multiple sensor devices, a sensor controller gateway, and an analysis server.

Sensor Controller Gateway: The gateway utilizes a Raspberry Pi 3 (Model B) as the host controller for the gateway device. The Raspberry Pi offers BLE 4.2 functionality as well as an integrated WiFi module and the possibility to be extended with a GSM module for online access. To match our use case, the host board connects to a SE. We propose the A71CH plug and trust chip from NXP Semiconductors, specifically designed for IoT devices. It provides a root of trust at the IC level and delivers different chip-to-cloud security features. This SE is useful for provisioning embedded devices, for instance. In order to facilitate the integration procedure of the A71CH chip, a potential setup could foresee the usage of the A71CH Mini PCB board (OM3710/A71CHPCB). An I2C interface connects the host controller (Raspberry Pi) and the SE (A71CH Mini PCB board).

Sensor Device: The sensor devices also integrate the A71CH secure element. Different host controller options are possible since the A71CH chip is compliant to any NXP MCU/MPU development board with Arduino compatible header, including i.MX, Kinetis, and LPC boards. In order to guarantee full BLE compatibility with the sensor controller gateway, the sensor devices use BLE chips of version 4.2 or higher.We further propose the usage of the temperature sensor DS18B20.

Analysis Server: The analysis server utilizes an Amazon EC2 instance. For the functionality requiring an HSM, the

TABLE VII: Overview on what and how devices use the symmetric keys and asymmetric certificates.

Key	Usage Example	Components
Master Key	Root of trust	SE
Binding Key	Secure SPI channel	SE ⇄ MCU
Session Key	Encrypt BLE channel	sensor ⇄ gateway
	Encrypt HTTP channel	gateway ⇄ server
Certificate	BLE key exchange	sensor ⇄ gateway
	TLS Authentication	gateway ⇄ server

server uses the AWS CloudHSM. It is a cloud-based hardware security module that enables the creation and management of cryptographic keys on the AWS cloud. The application on the server connects to the HSM using mutually authenticated SSL channels that are established by the client software of the HSM.

Last but not least, Table VII extends Table IV by showing all key types and mapping them to a specific usage example including all active components. The master key is our long term key and acts as the root of trust. Furthermore, while the binding key may secure the local SPI channel of a device, the session keys could be used to encrypt either a BLE or an HTTP channel. Additionally, the asymmetric key inside the certificate enables the key exchange mechanisms and authentication/authorization procedures.

V. Conclusion and Future Work

This paper presents a security-based DSE framework, which considers security attacks, security requirements, and key material placement. The paper shows that the framework is capable of finding the security optimal, secure and performance-optimal or secure and power-optimal solution, considering a secure system partitioning, task allocation, security functionality selection, and key material placement. The evaluation shows that the framework is capable of finding optimal solutions in a large and complex system design space. Within the use case evaluation, a possible system realization, based on the solutions found by the framework, was designed. A known limitation to the framework is its reliance on security expert knowledge for assessing the attack scenarios and security vulnerability risk of the utilized hardware components. The security assessment performed for the use case presented here used a Common Criteria based approach as a basis for security estimations. In future work, a special focus will lie on an integrated security assessment method considering said vulnerability risks and attack probabilities. Future work will extend the framework to also consider the usage of different secure channel protocols and their distinct message overheads. Last but not least, based on *blinded for review*, the set of possible use cases will be extended by involving different communication participants, paying special focus to end-to-end security methodologies, and integrating different authentication and authorization schemes in a distributed context.

Acknowledgment

Project partners are NXP Semiconductors Austria GmbH, CISC Semiconductor and Graz University of Technology. This work was supported by the Austrian Research Promotion Agency (FFG) within the project STIP (project number: 867914).

References

[1] M. N. et al. Al-mhiqani. Cyber-Security Incidents : A Review Cases in Cyber-Physical Systems. 9(1), 2018.
[2] Chih Ta Lin et al. Cyber attack and defense on industry control systems. *2017 IEEE Conf. on Dependable and Secure Computing*, 2017.
[3] Lorenzo Pagliari et al. Multi-modeling Approach to Performance Engineering of Cyber-Physical Systems Design. *Proc. of the IEEE Int. Con. on Engineering of Complex Computer Systems, ICECCS*, 2018.
[4] Kathrin Rosvall et al. Throughput Propagation in Constraint-Based Design Space Exploration for Mixed-Criticality Systems. *Proc. of the 9th Workshop on Rapid Simulation and Performance Evaluation: Methods and Tools - RAPIDO '17*, 2017.
[5] Bastian Knerr. *Heuristic Optimisation Methods for System Partitioning in HW / SW Co-Design*. PhD thesis, Vienna University of Technology, 2008.
[6] Kathrin Rosvall et al. Exploring power and throughput for dataflow applications on predictable NoC multiprocessors. *Proc. - 21st Euromicro Conf. on Digital System Design, DSD 2018*, 2018.
[7] Yong Xie et al. Security/Timing-aware Design Space Exploration of CAN FD for Automotive Cyber-Physical Systems. *IEEE Trans. on Industrial Informatics*, 2018.
[8] Monowar Hasan et al. A design-space exploration for allocating security tasks in multicore real-Time systems. *Proc. of the 2018 Design, Automation and Test in Europe Conf. and Exhibition, DATE 2018*, 2018.
[9] Eunusk Kang. Design Space Exploration for Security. *IEEE Cybersecurity Development Design*, 2016.
[10] Ingo Stierand et al. Integrating the security aspect into design space exploration of embedded systems. *Proc. - IEEE 25th Int. Symposium on Software Reliability Engineering Workshops, ISSREW 2014*, 2014.
[11] Bowen Zheng et al. Cross-Layer Codesign for Secure Cyber-Physical Systems. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 2016.
[12] Letitia W. Li et al. Security-aware Modeling and Analysis for HW/SW Partitioning. *Proc. of the 5th Int. Conf. on Model-Driven Engineering and Software Development*, 2017.
[13] Nan Feng et al. A security risk analysis model for information systems: Causal relationships of risk factors and vulnerability propagation analysis. *Information Sciences*, 2014.
[14] Nayot Poolsappasit et al. Dynamic Security Risk Management Using Bayesian Attack Graphs. *IEEE Transactions on Dependable and Secure Computing*, 2012.
[15] Tma-Selicean, Domiian and Pop, Paul. Design Optimization of Mixed-Criticality Real-Time Embedded Systems. *ACM Trans. on Embedded Computing Systems*, 2015.
[16] Yves Roudier and Ludovic Apvrille. SysML-Sec - A Model Driven Approach for Designing Safe and Secure Systems. *Proc. of the 3rd Int. Conf. on Model-Driven Engineering and Software Development*, 2015.
[17] L. Gressl et al. Consideration of Security Attacks in the Design Space Exploration of Embedded Systems. In *2019 22nd Euromicro Conf. on Digital System Design (DSD)*, 2019.
[18] Erik Miehling et al. Optimal defense policies for partially observable spreading processes on bayesian attack graphs. In *Proc. of the 2nd ACM Workshop on Moving Target Defense*, 2015.
[19] Xiaoyan Sun et al. Using Bayesian Networks for Probabilistic Identification of Zero-Day Attack Paths. *IEEE Trans. on Information Forensics and Security*, 2018.
[20] Vivek Shandilya et al. Use of attack graphs in security systems. *Journal of Computer Networks and Communications*, 2014.
[21] Shawn Hernan et al. Uncover security design flaws using the stride approach (2006). *URL http://msdn. microsoft. com/en-gb/magazine/cc163519. aspx*, 15.
[22] M. Schiffman. Common Vulnerability Scoring System (CVSS). *URL https://www.first.org/cvss/v3.1/specification-document*, 2019.
[23] Tobias Schläpfer and Andreas Rüst. Security on IoT Devices with Secure Elements.
[24] Sharon Levy. Performance and Security of ECDSA. pages 1–4.
[25] Litty Raju and Manickam Sumathi. Secured High Throughput of 128-bit AES Algorithm based on Interleaving Technique. (January), 2015.

Towards Security Attack and Risk Assessment during Early System Design

Lukas Gressl*, Michael Krisper*, Christian Steger* and Ulrich Neffe†
*Graz University of Technology
Institute of Technical Informatics, Inffeldgasse 16, A-8010 Graz
Email: {gressl}{michael.krisper}{steger}@tugraz.at
†NXP Semiconductors Austria GmbH
Mikron-Weg 1, A-8101 Gratkorn
Email: ulrich.neffe@nxp.com

***Abstract*—The advent of the Internet of Things (IoT) and Cyber-Physical Systems (CPS) enabled a new class of smart and interactive devices. With their continuous connectivity and their access to valuable information in both the digital and physical world, they are attractive targets for security attackers. Hence, with their integration into both the industry and consumer devices, they added a new surface for cybersecurity attacks. These potential threats call for special care of security vulnerabilities during the design of IoT devices and CPS. The design of secure systems is a complex task, especially if they must adhere to other constraints, such as performance, power consumption, and others. A range of design space exploration tools have been proposed in academics, which aim to support system designers in their task of finding the optimal selection of hardware components and task mappings. Said tools offer a limited way of modeling attack scenarios as constraints for a system under design. The framework proposed in this paper aims at closing this gap, offering system designers a way to consider security attacks and security risks during the early design phase. It offers designers to model security constraints from the view of potential attackers, assessing the probability of successful security attacks and security risk. The framework's feasibility and performance is demonstrated by revisiting a potential system design of an industry partner.**

***Index Terms*—Cyber Security; Embedded System Design; Secure IoT Systems; Design Space Exploration; Secure Embedded Consumer Devices**

I. Introduction

The Internet of Things (IoT) is continuously revolutionizing both industry and commercial products. It offers the interaction of various devices, making them accessible to the Internet. However, the great benefits offered by the IoT make its devices susceptible to cyber-security attacks. Since its advent, reports have described numerous exploits caused by IoT devices [1], [2]. Designing secure IoT devices has, hence, become a necessity. IoT devices, embedded into larger systems, usually come with a multitude of non-functional requirements, such as timing constraints, limited power dissipation, etc. Security hardening of IoT devices generally conflicts with these requirements [3]. Finding an optimal design satisfying all requirements means solving a multi-objective optimization problem. Design space exploration (DSE) tools aid designers in this complex task during the early system design [4]–[6].

This paper describes a tool capable of finding secure system solutions considering constraints such as performance, power consumption, etc. It performs a selection of hardware components and task allocation, satisfying the non-functional requirements posed by the designer. Thereby, it allows the designer to model the security constraints as an attack tree, a risk tree, or a combined attack and risk tree. The main contributions are: (i) It describes the first DSE tool to offer the modeling of security constraints as Bayesian Attack Graphs (BAGs), risk trees (RISKEE), and a combination of both representations. (ii) It shows the costs and benefits of these approaches using the design of a secure access system.

The paper contains the following sections: Section II describes related work. Section III describes the tool's design and implementation. Section IV discusses the use case and shows the impact of the distinct security modeling approaches, considering the found solutions and the execution times. Section VI concludes the paper.

II. Related Work

Classic DSE tools support system designers by performing task allocations and hardware component selections to satisfy constraints, such as system delay, power consumption, etc. [7], [8]. There exists a range of works in the context of DSE, putting their focus on cybersecurity requirements. Several works in this category focus on distinct problems. Xie et al. consider message authentication codes (MACs) transferred via a Controller Area Network. They optimize the packet sizes transferring the messages and their MACs to meet a global communication delay optimum [5]. Hasan et al. solve the problem of integrating security surveillance tasks into an existing task schedule without breaking existing timing constraints. Other works abstractly consider the integration of cybersecurity requirements. Roudier and Apvrille presented a framework that allows designers to integrate security and safety functions into early system design. The framework considers these functions during its DSE, optimizing its solutions to satisfy security and safety constraints in addition to timing- and power-consumption-constraints, etc. [9]. Zheng et al. integrate security constraints into the design of cyber-physical systems performing control-theoretic operations [10].

The frameworks discussed in this section either lack the holistic view on the system or loose too many details for considering security constraints during the early design phase. Furthermore, none of the frameworks considers the attacker's view of the system under design. The tool presented here aims at closing this gap. It allows the designers to model security requirements as Bayesian attack graphs (BAGs), risk trees, or a combination of both representations. Latter two descriptions allow the further weighing of security risks and their mitigation costs. Furthermore, it supports designers to model security mechanisms used to mitigate the modeled attacks. These mechanisms induce additional timing- and power-consumption-overheads, etc. This additional overhead is considered by the tool when calculating the system performance, power consumption, etc., allowing the designer to perform an assessment of the costs and benefits of each solution.

A range of works has been studying BAGs, mostly in the domain of security analysis for networks. These BAGs split attack scenarios into distinct steps, represented as nodes in the graph. Each node contains the likelihood of successfully performing its represented attack step [11], [12]. The recently presented RISKEE approach models cyber-security in the form of a risk tree [13].

III. Modeling Approach and Constraint Calculation

The framework spans a design space based on the functional-, architectural-, and attack-descriptions. System designers provide these descriptions. Based on these inputs, the framework derives the security constraints, performs different task mappings and hardware component selections, and calculates the distinct system design solutions. This section gives details about the input modeling and the solution calculations. The tool is based on [14] but allows for a more open representation of the relationship between tasks, hardware-components, and attacks. The main differences include the freely adaptable rule set defining the dependencies among the distinct input descriptions and its integration of a risk-based calculation of security constraints, introducing a combined approach using both BAGs and risk trees.

Functional description: The tool supports the description of the system's functionality in the form of a task graph. In this graph, every node represents a distinct task, its edges the sequence of their execution. Each task in the graph might perform operations on a set of data entities. These data operations (O) comprise the transmission (tx), reception (rx), writing (w), reading (r) and storing (st) of the data entities. The data entities come with distinct sizes.

Architecture description: The architectural description comprises hardware components connected via communication buses. The system designers describe the hardware components and communication buses with classic characteristics, such as static and dynamic power consumption, etc. The communication also comprises the transmission speed.

Attack views: For the calculation of the security constraints, the framework takes as additional inputs the potential attack

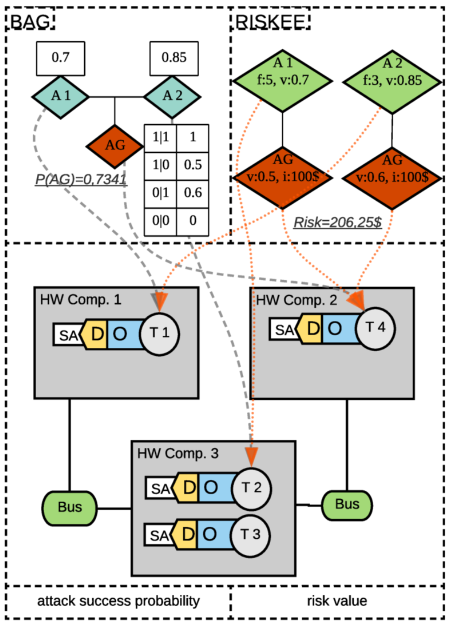


Fig. 1: Framework overview. Attacks (A) aim at tasks (T), allocated to hardware components (HW. Comp.), operating (O) on data entities (D) with security assets (SA).

scenarios described as BAGs or risk trees. Both representations come in a graph form and are, therefore, exchangeable and combinable. The BAG based method describes attack scenarios as distinct attack steps in a graph format. Each step has a distinct probability of being successfully performed, stored within the step's conditional distribution table. The attacker's goal is to reach the leaves of the BAG. The framework calculates the likelihood with which the attacker can reach a distinct goal using the Bayesian chain rule [15].

The second method uses RISKEE [13], which is a graph-based approach for risk assessment. RISKEE allows the modeling of the consequences (or impacts) of a successfully performed attack step. It enables the designer to add attack frequencies, thus accounting for multiple attacks over time. Furthermore, it uses probability distributions to add uncertainty when assessing an attack success probability. Hence, it provides a more detailed description compared to single-point estimates, which do not consider uncertainties. The framework uses the mean risk value (one metric returned by RISKEE) to express the risk induced by reaching an attack goal.

Each attack step in both methods aims at a distinct task of the functional description. Both the RISKEE and the BAG method are graph-based, making them combinable with each other. The only limitation is that for each attack goal, the system designers must model all nodes in the path to this goal consistently using the RISKEE or BAG approach (or both). The framework's configuration defines with what method it calculates the security constraints. The combined process comes with the advantage of the detailed modeling approach of the RISKEE method and faster execution time. As can be seen in Section IV, the combined approach calculates the attack success probability (asp) for each security goal using the BAG first. In a second step, it recalculates the goal using RISKEE, if the BAG based calculation did not exceed the goal's threshold, and its attack path allows it. The timing advantage of this approach is described in Section V.

Security characteristics: Additional characteristics are necessary for the functional and architectural description, as well as in the attack views, to calculate the security constraints. In the functional description, the designer adds a set of security assets (SA) to each data entity. These assets define what needs to be secured by the task, based on its interaction with the data entity (e.g., confidentiality). The assets are freely definable by the designer. The hardware components in the architectural description additionally contain a set of security mechanisms (SM) determining the capabilities with which they secure the SA of their allocated data entities. The SM further define, if they are usable for inter-task security within the same hardware component (internal (int)) or between tasks allocated on different components (external (ext)). Each component has a distinct attack mitigation factor ($amf \in \mathbb{R} : m \in [0, 1]$). This amf describes the degree to which attacks on the SA are mitigated by the component's SM. Hence, it describes the thoroughness with which the components implement their SM. In the attack view, each attack contains an attack-type (at). This attack-type defines what SA of the targeted data entity it aims at. The framework allows the unrestricted definition of SM and at by the designer.

Estimation of attack success probability and mitigation: To estimate the attack asp of an attack, the security experts use the Common Vulnerability Scoring System (CVSS) [16]. Our approach relies on the usage of the *Base Metrics* from the CVSS. These metrics cover an attack's complexity and its impacts, and, hence, can be used to describe the attacker's motivation on performing the attack. For the BAG based approach, the these metrics are combined in the asp. For the RISKEE based approach the impact is separately modeled. The estimation of the amf can be based on Common Criteria (CC) certifications, where available. The CC certifications describe the attacker's strength against whom the certified component should be able to withstand [17]. If no CC certification is available, the security experts must judge a component's attack vulnerability based on historic data or other documents.

Secure task allocation and partitioning: Based on the functional description, the architecture description, and the attack models, the framework calculates the security constraints posed on the system design. The security constraint calculation starts with determining the set of security actions ($SAct$) a task must perform to secure its data entities. These actions depend on the operations a task performs on its data entities and the data entities' SA. The framework then calculates the set of SM each component must offer to allow its mapped tasks to perform their $SAct$. The framework defines the secure data exchanges (SDE), spanned between the distinct tasks, using the $SAct$. An SDE consist of a source and a destination task, where both source and destination task perform the same $SAct$ on a shared data entity. The framework determines for each SDE if the source and destination tasks map to the same or different hardware components. Thus, it checks if tasks' $SAct$ must be supported by internal or external SM. Each mapping between O, SA and $SAct$, $SAct$ and SM, and at to SA is definable by the designer through Boolean expressions. Table I describes the security rules applied in the example use case.

The framework finds a secure task allocation and system partitioning. First, it restricts the task to map only to components capable of performing their $SAct$. Second, it calculates the asp or risk (depending on the used method) for each component selection and task allocation. Thereby, the amf of a hardware component reduces the asp of all attacks aiming at tasks allocated on it ($asp_m = asp * (1 - amf)$). The framework performs this calculation of all solutions and marks them as either secure or insecure (if any goal exceeds its threshold). Figure 1 shows the task allocation, as well as the assignment of the attack nodes to distinct tasks.

Performance and power consumption: The performance and power consumption calculation builds on the estimation of the worst case execution time ($wcet$) of each task and the communication delay of each communication bus. This $wcet$ reflects the execution time of a task when implemented on a distinct hardware component. This task implementation is denoted with $impl(t_a)$ defining on what hardware component task t_a is implemented. Hence, $wcet(impl(t_a))$ denotes the $wcet$ of task t_a on the hardware component it is implemented on. The system performance also depends on the delays induced by the distinct SM used by each task implemented on a hardware component. This delay is denoted as $\delta(SMT(impl(t_a)))$, with $SMT(impl(t_a))$ being the used security measures of the hardware component allocating t_a. The communication overhead depends on the data entities (d) sent over the communication buses ($b_y(d)$) and the bus' transmission speed ($\lambda(b_y)$). The overall system performance ($sys_{perf} = \sum_{i=1}^{n}(wcet(impl(t_i)) + \delta(SMT(impl(t_i)))) + \sum_{j=1}^{m}(b_j(d) * \lambda(b_j))$) is the sum of the $wcet$ of all task mappings, the delays of the used SM, and the communication delays, where n denotes the number of all tasks, and m denotes all used communication buses used in the solution.

The power consumption of the system depends on the static, the dynamic power consumption, and the power consumption induced by the used SM, and the power consumed by the bus communication. The static power consumption $\rho_s = \sum_{j=1}^{m} pwr_s(c_j)$ is the sum of the static power consumption of all hardware components, with m being the number of all components used on the distinct solution. The dynamic power consumption $\rho_d = \sum_{i=1}^{n}(wcet(impl(t_i)) * pwr_d(impl(t_i)))$ is the sum of the tasks' $wcet$ multiplied with the dynamic power consumption of the hardware components they are mapped to. The power consumed by the used security mechanisms ($\phi(SMT(impl(t_a)))$) depends on the SMT used by the tasks allocated on the hardware components. The overall security power consumption is $\rho_{sec} = \sum_{i=1}^{n} \phi(SMT(impl(t_i)))$. The power dissipated by the communication bus depends on the $b_y(d)$ and the bus' power consumption ($\epsilon(b_y)$). The system's communication power consumption is $\rho_{comm} = \sum_{j=1}^{k} (b_j(d) * \epsilon(b_j))$, with k being the number of all communication buses used by the solution. The system power consumption (sys_{pwr}) is the sum of static, dynamic, SM induced, and communication caused power consumption $sys_{pwr} = \rho_s + \rho_d + \rho_{sec} + \rho_{comm}$. This influence can be seen in Section IV.

Implementation: The framework's implementation is based on the *DeSyDe* framework[1]. The *DeSyDe* framework is a DSE tool using a constraint programming approach. It is capable of finding exact solutions, adding break criteria such as maximum delay or power consumption. For the security constraint calculate, the framework performs a limited permutation approach to reduce the number of tasks to component mappings. This approach uses the attack goals' asp, or the risk exceedance, as break criteria. The approach optimizes the security constraint calculation by sorting the hardware components according to their amf. The second optimization comes with the combination of the BAG and the RISKEE method. The RISKEE method has a higher computation time for calculating the security constraints than the BAG approach. In the combined approach, the framework calculates each attack goal using the BAG method, and if the goal yields a asp below its threshold, it uses RISKEE to refine the result of the goal even further. This combined approach achieves highly informative results from RISKEE while still maintaining the performance of the BAG method. This combined approach's speedup is shown in Section V.

IV. Experiment

We used the framework for the early design of a keyless entry system. This system consists of a mobile node and a stationary system comprising several anchors. Both nodes and anchors use an application processor (AP), a secure element (SE), a Bluetooth Low Energy radio, and an ultra-wideband (UWB) radio (UR). The anchor system measures its distance to the node applying a double-sided two-way ranging algorithm using the UR. The devices use a set of SM offered by the hardware components. These SM secure the SA of the data entities and the communication and the overall ranging process between the node and the anchors. We derived these security characteristics conducting a STRIDE and CIA analysis [18], [19]. The rules listed in Table I describe the mappings between the SA, O, $SAct$, SM, and at.

Table II lists the hardware components that are usable for both anchor and the node device. This table lists the components' security features, their amf, and the performance and power consumption of their single security mechanisms (sm). The framework can choose between different options for the AP, the SE, and UR. They comprise the usage of hardware-based (HWC) and software-based cryptography

[1]https://github.com/forsyde/DeSyDe

TABLE I: Security rules of the use case. The at comprise information disclosure (at_i), spoofing (at_s), and tampering (at_t). The $SAct$ comprise encryption (a_e), authentication (a_a), and secure storage (a_s). The SA comprise confidentiality (c), authenticity (a), and integrity (i). The SM comprise cryptography (sm_c), task encapsulation (sm_e), and tamper safe storage (sm_s)

$SAct$ from SA, O	AT to SA	$SAct$ to SM
$a_e = (r \vee w) \wedge c$	$at_i \mapsto c$	$sm_c = (a_e \vee a_a) \wedge ext$
$a_a = (r \vee w) \wedge a$	$at_s \mapsto a$	$sm_e = (a_e \vee a_a) \wedge int$
$a_s = st \wedge (a \vee c \vee i)$	$at_t \mapsto i$	$sm_s = a_s$

TABLE II: Hardware components with security options. Attack mitigation factor (amf), performance (Perf) given in µs, and power consumption (PWC) in mW

HWC	Sec. Feat.	amf	Perf/PWC		
			sm_c	sm_s	sm_t
AP	HWC, TEE	0.75	40/50	-/-	10/10
	SWC scp, TEE	0.6	70/60	-/-	10/10
	SWC scp	0.5	70/60	-/-	-/-
	SWC f	0.2	45/40	-/-	-/-
SE	EAL 6+	0.95	110/70	50/20	20/15
	EAL 5+	0.9	100/60	30/10	15/10
	EAL 4+	0.8	100/50	20/10	10/10
UR	HWC, FW	0.6	50/30	-/-	15/10
	HWC, TZ	0.5	50/30	-/-	10/10
	HWC, MS	0.7	60/40	-/-	20/20
	SWC, TZ	0.4	80/40	-/-	10/10
	SWC, f.	0.2	50/35	-/-	-/-

(SWC), trusted execution environment (TEE), trust zone (TZ), hardware firewall (FW), and microcontroller separation (MS). The SWC comes with side-channel protection (scp) or is purely functional (f). The HWC and SWC only characterize the security mechanisms offered by the hardware components and do not concern the implementation of the tasks mapped on the hardware component.

The functionality of the use-case described in here comprises a task graph with 57 nodes, resembling an authentication, session exchange, and ranging phase. The potential attacker who attacks the system is thought to be capable of sniffing and intercepting the communication between the devices, sniffing the communication within the single devices, tampering with the devices' memory, and intruding the software stack of the devices to a certain degree. The attacker has computational resources to brute-force weak cryptographic algorithms using small secret keys. The attacks comprise the secret key disclosure, faking secure authentication, hijacking the session, and compromising the ranging messages. The overall attack scenario consists of 64 attack nodes. Both the BAG and the risk tree have the same attack nodes. Table III lists the security relevant tasks with their SA and WCETs.

The purpose of the use case is to show the framework's feasibility and capability of finding solutions based on the given constraints and the optimization goal. Hence, we configured the framework to find the most secure solution, the solution with the best overall performance, the fastest yet secure solution, and the secure solution with the least power consumption. The attack goals' thresholds were uniformly set

TABLE III: WCETs of security relevant tasks given in μs.

Device	Task Name	SA	AP	SE	UR
Node	create challenge	c,a	80	100	-
Anchor	check challenge	c,a	70	90	100
Node & Anchor	derive session key	c,a,i	-	140	80
Node & Anchor	derive rng key	c,a	-	120	60
Anchor	start rng session	c,a	-	150	100
Node & Anchor	create sec. nonce	c,a	-	150	80
Node & Anchor	create ranging message	c,a	-	90	50
Anchor	calculate distance	c,a	-	250	140
Node & Anchor	create chall. open	c,a	30	80	50
Node & Anchor	check chall. open	c,a	20	90	50

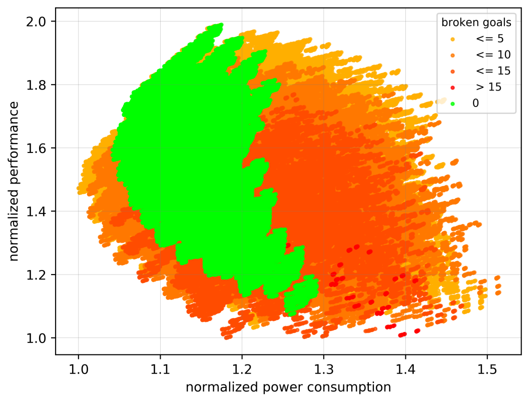


Fig. 2: Solution space identified by the framework using the BAG based method.

to 0.005 for the BAG based security constraint calculation, and to $2,000\$$ for the RISKEE based one. The combined BAG and RISKEE method uses the same thresholds.

The system presented in the use case targets secure access to a vehicle, supporting a secure keyless entry system. It allows access to the car only to authorized persons who are nearby. The distance between the vehicle hosting the anchor system, and the authorized person holding the tracked node, is acquired by executing a ranging protocol based on the exchanging of UWB packets. The system generates a timely limited session key to secure the ranging process. This session key is derived from a master key stored at each device and exchanged after every ranging session. Disclosing the session key enables the attacker to gain access to the car for a short period, whereas revealing the master key means timely unlimited access to one or many cars, depending on the key distribution strategy.

The impact of disclosing the session key was set to $250,000\$$ and for the master key to $30,000,000\$$. We set these values to evaluate the risk induced by the value of the session and master keys. We assumed that the master key is less prone to disclosure, as, naturally, it should be more challenging to access. Hence, we set the frequency with which a potential attacker attempts to discover the master key to 10 times per year, for the session key to 50 times per year. The risk tree used for the use case resembles these impact and frequency values. The vulnerabilities of all attacks in the risk tree equal the asp of the identical attack steps in the BAG. Hence, the differences between the RISKEE and BAG based methods stem from the characterization of the attack impact and frequency.

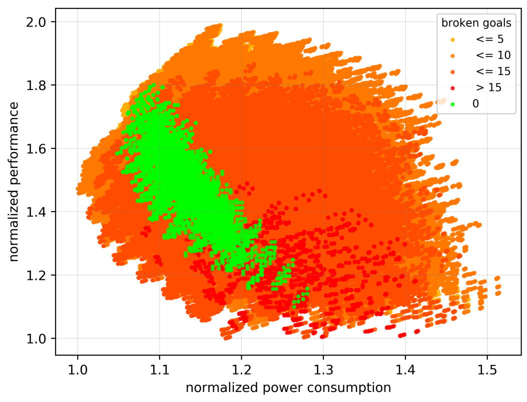


Fig. 3: Solution space identified by the framework using the RISKEE based method.

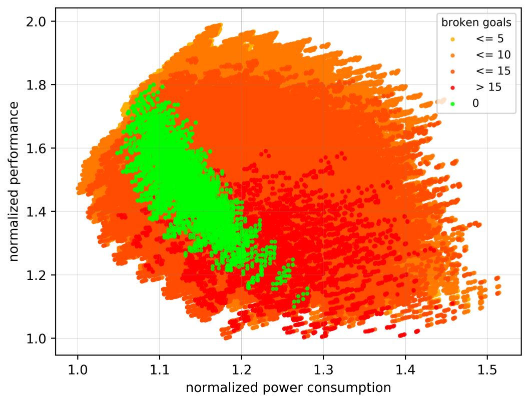


Fig. 4: Solution space identified by the framework using the combined method of RISKEE and BAG.

The framework calculates, additionally to the security constraints (how many attack goals are reached), the average attack probability ($AP_{avg} = \frac{\sum_{i=0}^{G} asp_i}{G}$) and the average mean risk value ($MRV_{avg} = \frac{\sum_{i=0}^{G} mrvi}{G}$), where G is the number of all attack goals, and mrv is the goal's mean risk value. The mean risk value is one parameter returned by the RISKEE. The framework uses this parameter to assess the risk of the attacker reaching the attack goal. The framework calculates these values to characterize the security soundness of a solution. Table IV shows the solution with the lowest security vulnerability (considering the AP_{avg} and MRV_{avg}) and the system setup with the best performance. It shows that the framework can identify the optimal solutions.

Figures 2, 3, and 4 depict the solution space identified using the BAG based, the RISKEE based, and the combined approach. All three approaches found overall 947.072 solutions that fulfill the basic security functionalities. Each solution represents a valid mapping of tasks to hardware components regarding the $SAct$ and SM. The BAG based computation resulted in 576.000 solutions found to meet all attack goal thresholds. The RISKEE based method found 384 solutions that do not exceed any attack goal threshold. The BAG- and RISKEE-based calculation approach largely differ in the number of secure solutions. The BAG-based approach reduces the number of solutions by 39.18%, the RISKEE-based approach by 99.96%. The additional attributes of the RISKEE method allow a narrowing of the solution space. The vulnerabilities in RISKEE and the ap in the BAG are identical. Hence, the reduction of secure solutions only stems from the frequency and the impact of the attacks. The master key's disclosure mainly influences the number of secure solutions.

The impact of a successful key disclosure also influences the fastest and least power consuming secure solutions, as described in Table V and VI. Due to the high impact of the secret key disclosure, the RISKEE based method selects hardware components with a higher amf than the BAG based approach. Especially the selection of the SE with the highest EAL (EAL6+) is of importance, as it stores the master key, and hosts the task deriving the session key. Thus, the framework must select SEs with high-security levels. Otherwise, the high impact of the key disclosure attacks would cause their attached security goals to exceed their thresholds.

The combined approach, which uses the BAG based calculation together with RISKEE, produced the same amount of secure solutions as the RISKEE based approach. The combined approach first calculates the asp of each goal using the BAG method. Only if this calculation produces an asp not exceeding the goal's threshold, the combined approach recalculates the goal's mean risk using the RISKEE method. Hence, by delivering the same amount of secure solutions as the RISKEE based approach, it shows that all solutions found by the RISKEE method are also contained in the secure solutions found by the BAG approach. The combined method is capable of producing the same set of secure solutions. However, for the insecure solutions, it can occur that the combined approach yields more broken goals than the BAG or RISKEE based method alone, as some goals may exceed their thresholds in the BAG but not in the RISKEE based calculation, and vice versa.

Table IV shows the fastest and most secure solutions found by the framework. The thresholds of the attack goals do not constrain the fastest solution. However, the solution still considers the feasibility of the $SAct$ when mapping the tasks to the hardware components. The most secure solution selects the task mapping and system partitioning with the lowest AP_{avg} and MRV_{avg}, respectively.

Table V describes the fastest secure solution found by the BAG and RISKEE method. Table VI the most power-efficient secure solutions, given their average attack probability and average mean risk. One can see that for both the secure solutions with optimal performance and power consumption, the RISKEE based solution chooses options with higher security attack mitigation capabilities than the BAG based approach, for both the SE and the AP of the anchor and node device. The more secure SE is chosen because of the high impact a possible disclosure of the session key or the master key causes. Said impact increases the influence of a successful key exposure on the average mean risk of the overall system dramatically. A similar result can be seen when considering the most power-efficient and secure solutions, regarding their average AP_{avg} and MRV_{avg}, respectively. Also, for this optimization criteria, the BAG based method chose less secure options for the SE and the node's AP, compared to the RISKEE based method.

TABLE IV: Most secure and fastest solution found based on AP_{avg}, and MRV_{avg}, with the delay normalized to system with lowest delay.

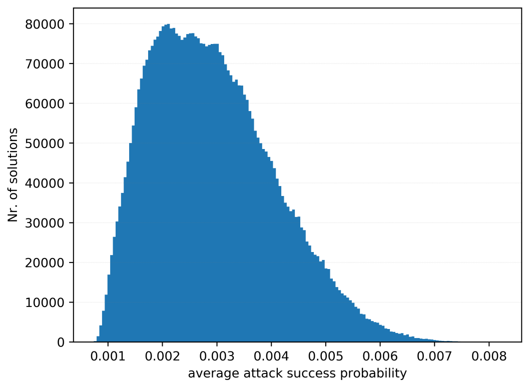
HWC	Options (most secure)	Options (fastest)
AP (node & anchor)	HWC, TEE	SWC f.
SE (node & anchor)	EAL 6+	EAL 4+
UR (node & anchor)	HWC, MS	SWC f.
AP_{avg} / MRV_{avg}	0.0007 / 117.45\$	0.005 / 3905.37\$
norm delay	~1.73	1.0

TABLE V: Fastest secure solutions found based on AP_{avg}, MRV_{avg}, and the delay normalized to system with lowest delay.

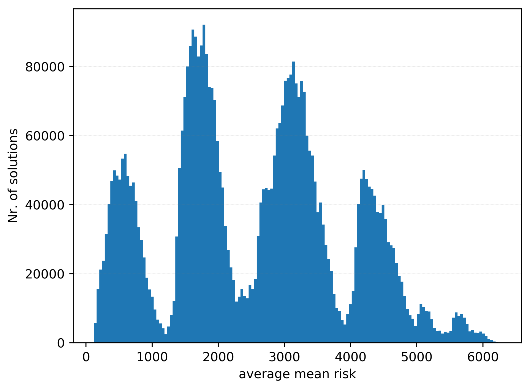
HWC	fastest secure (BAG)	fastest secure (RISKEE)
AP (node & anchor)	HWC, TEE	HWC, TEE
SE (node)	EAL 4+	EAL 6+
SE (anchor)	EAL 4+	EAL 6+
UR	HWC, TZ	HWC, FW
AP_{avg} / MRV_{avg}	0.0017	126.74\$
norm. delay	~1.074	~1.16

Figures 5b and 5a show the numbers of found solutions ordered by their AP_{avg} and MRV_{avg}, respectively. One should notice that the BAG based method produced more secure solutions, considering their number. The majority of the found solutions (43.08%) have an AP_{avg} between 0.0007 and 0.0026. Only 0.38% are found to have an AP_{avg} exceeding 0.006. Considering the RISKEE based approach, the histogram shows four peaks (placed at the MRV_{avg} of 656\$, 1845\$, 3444\$, and 4223\$). These peaks show how the RISKEE based approach delivers more pointedly results compared to the BAG based method. The RISKEE based method also delivers fewer secure solutions. Only 28% of the solutions induce an MRV_{avg} of less than 1658\$, whereas 39.48% come with an MRV_{avg} higher than 1658\$ and lower than 3196\$. Only 4.58% of the found solutions are regarded as highly risky, coming with an MRV_{avg} of more than 4735\$. The combined approach mixes both risk values in \$ and attack probabilities in %. Hence, a histogram over the solutions found by the combined approach gives no further insight.

We show how we use the framework for designing secure systems. We model the security constraints both with the BAG based and the RISKEE based approach. With the modeled use case, we show how the impact and frequency added by the RISKEE method influences the calculation of the security constraints, defining the attack consequences more precisely. This



(a) BAG based solutions found by the framework categorized according to their average attack success probability. Stepsize of $4.99 * 10^{-5}$



(b) RISKEE based solutions found by the framework categorized according to their average mean risk. Stepsize of 41$

Fig. 5: Solutions found for the secure ranging use case modeled using BAGs or RISKEE.

TABLE VI: Most power efficient and secure solutions found based on AP_{avg}, MRV_{avg}, and power consumption (power cons) normalized to solution with lowest sys_{pwr}

HWC	most power eff. secure (BAG)	most power eff. secure (RISKEE)
AP (node)	HWC, TEE	HWC TEE
AP (anchor)	HWC, TEE	HWC TEE
SE (node)	EAL 4+	EAL 6+
SE (anchor)	EAL 4+	EAL 6+
UR (node)	HWC, TZ	HWC, MS
UR (anchor)	HWC, TZ	HWC, MS
avg ap / rv	0.0013	126.17$
power cons	~1.0415	~1.068

more accurate definition comes, however, with the drawback of higher computation time. Hence, the framework introduces the combined approach of BAG and RISKEE. Section V shows the performance of the different methods.

V. Timing Evaluation

The framework was executed on a system providing 16 GB of RAM and an Intel® Core™ i7-4600U CPU with 2.10 GHz.

TABLE VII: Computational overhead for BAG, RISKEE and combined approach for the different variants.

Variants	BAG	RISKEE	Combo
Var. I	910.253s	1656.605s	1218.024s
Var. II	27.241s	297.445s	262.56s
Var. III	15.357s	112.092s	80.649s

We executed the use case using the BAG, the RISKEE, and the combined approach of BAG and RISKEE (Combo). We compared the execution time with and without using the permutation limitation (pl), as explained in Section III. Table VII and VIII list the execution times (with and without pl) for the use case presented in here. We split the use case into three variants. The first variant (Var. I) is the full execution of the whole use case. The second variant (Var. II) consists of the authentication and session establishing phase, leaving out the ranging phase. The third variant (Var. III) consists only of the authentication phase. As the variants use different phases, they also come with different numbers of found solutions. These solutions also include insecure mappings, as those also influence the computation time of the framework. Var. I consists of 947072 solutions, Var. II of 19728 solutions, and Var. III of 580 solutions. The timings presented in the tables reflect the actual CPU times (spent in user mode and kernel).

TABLE VIII: Computational overhead for BAG, RISKEE and combined approach for the different variants. With permutation limitation (pl)

Variants	BAG (pl)	RISKEE (pl)	Combo (pl)
Var. I	453.326s	1072.002s	744.09s
Var. II	26.165s	214.594s	179.992s
Var. III	13.824s	51.529s	45.966s

The timings listed in Table VII and VIII show how the different approaches scale. The framework's execution time contains a dynamic part which only depends on the computation time of finding the solutions. It further also contains a static performance overhead consisting of a ramp-up and clean up phase. Hence, the execution time spent per solution decreases with the increase in calculated solutions.The framework comes with an worst-case execution time of $27\text{min}37\text{s}$ considering the complete use case solutions.

As can be seen in Table VII, the chosen approach for calculating the security constraints has a great impact on the overall execution time needed for finding the task mappings and system partitioning. The greatest difference between the execution time lies between the BAG and RISKEE based approach. The additional execution time needed for the RISKEE method compared to the BAG approach comes with ~82% for Var. I. For this variant, the combined BAG and RISKEE approach comes with a speedup of 26.47%, compared to the RISKEE based approach. Figure 6 depicts the calculation times. The usage of the break criteria also increases the execution speed of the security constraint calculation. The permutation limitation speeds up the calculation using the

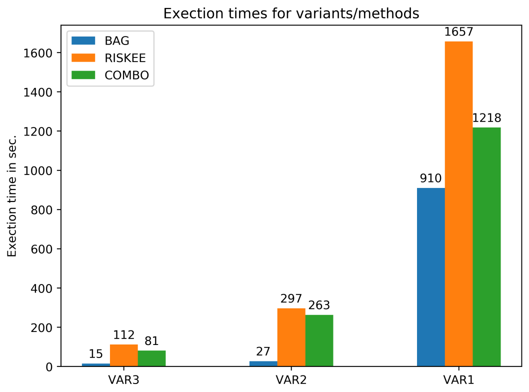


Fig. 6: Execution times for the different methods calculating the use case variants.

BAG method by 50.2%, the RISKEE method by 35.29%, and the combined method by 38.91%, when calculating the security constraints for Var. I. The execution time greatly depends on the attack scenarios modeled for the use case.

VI. Limitations and Conclusion

The main limitation of the approach presented here is its dependency on accurate estimations of potential attacks and the effectiveness of their countermeasures. To estimate attacks accurately, one should follow state-of-the-art estimation techniques, such as [16], [18], [19]. These methods are usable when capturing the attack success probabilities of known systems. Especially the CVSS presents a good base as it covers attack complexity and impact metrics from which one can draw an estimation of the attacker's motivation. Attacks on known systems can be used to estimate the potential of new attacks, as many of them share the same base. However, for completely new attacks the approach presented here is not applicable. Furthermore, considering the estimation for the countermeasures, no method has yet been published on how to rate a system's ability to withstand security attacks. Our assumptions made for the use case stem from CC certifications. Here, we linked the attack's complexity from the CVSS to the mitigation capability of the CC regarding an attacker's strength. Methods on how to estimate a system's vulnerability to completely new attacks is out of this paper's scope.

The main purpose of our approach is to provide a design framework for integrating a system's attack susceptibility with its traditional requirements on performance, power consumption, etc. The framework offers the system designers to model security constraints as BAGS or risk trees, in an early design phase. It automatically calculates the task mappings and system partitioning to arrive at secure system solutions, given its inputs. It is capable of providing the designer with secure solutions adhering to various other system constraints, such as performance, power consumption, etc. Using a secure access system use case, we showed the feasibility and the scalability of our approach. By introducing a combined calculation method, which first calculates attack goals using the BAG approach and then verifies them using the risk trees, we were able to speed up the overall calculation time.

Acknowledgment

Project partners are NXP Semiconductors Austria GmbH and the Technical University of Graz. This work was supported by the Austrian Research Promotion Agency (FFG) within the project UBSmart (project number: 859475).

References

[1] Seokung Yoon et al. Security issues on smarthome in iot environment. In *Computer Science and its Applications*. Springer Berlin Heidelberg, 2015.
[2] Mohammed Nasser et al. Cyber-Security Incidents: A Review Cases in Cyber-Physical Systems. *Int. Journal of Advanced Computer Science and Applications*, 9, 2018.
[3] Ruozhou Yu et al. Deploying robust security in internet of things. *2018 IEEE Conf. on Communications and Network Security, CNS 2018*, 2018.
[4] Kathrin Rosvall et al. Exploring power and throughput for dataflow applications on predictable NoC multiprocessors. *Proc. - 21st Euromicro Conf. on Digital System Design, DSD 2018*, 2018.
[5] Yong Xie et al. Security/Timing-aware Design Space Exploration of CAN FD for Automotive Cyber-Physical Systems. *IEEE Transactions on Industrial Informatics*, 2018.
[6] Ingo Stierand et al. Integrating the security aspect into design space exploration of embedded systems. *Proc. of IEEE 25th Int. Symp. on Software Reliability Engineering Workshops, ISSREW 2014*, 2014.
[7] Bastian Knerr. *Heuristic Optimisation Methods for System Partitioning in HW / SW Co-Design*. PhD thesis, Vienna University of Technology, 2008.
[8] Kathrin Rosvall et al. Throughput Propagation in Constraint-Based Design Space Exploration for Mixed-Criticality Systems. *Proc. of the 9th Workshop on Rapid Simulation and Performance Evaluation: Methods and Tools - RAPIDO '17*, 2017.
[9] Yves Roudier and Ludovic Apvrille. SysML-Sec - A Model Driven Approach for Designing Safe and Secure Systems. *Proc. of the 3rd Int. Conf. on Model-Driven Engineering and Software Development*, 2015.
[10] Bowen Zheng et al. Cross-Layer Codesign for Secure Cyber-Physical Systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2016.
[11] Nan Feng et al. A security risk analysis model for information systems: Causal relationships of risk factors and vulnerability propagation analysis. *Information Sciences*, 2014.
[12] Nayot Poolsappasit et al. Dynamic Security Risk Management Using Bayesian Attack Graphs. *IEEE Transactions on Dependable and Secure Computing*, 2012.
[13] Michael Krisper et al. RISKEE : A Risk-Tree Based Method for Assessing Risk in Cyber Security. In *Proc. of EuroSPI 2019: European System, Software & Service Process Improvement & Innovation*, 2019.
[14] L. Gressl et al. Consideration of Security Attacks in the Design Space Exploration of Embedded Systems. In *2019 22nd Euromicro Conf. on Digital System Design (DSD)*, 2019.
[15] David Heckerman and John S. Breese. Causal independence for probability assessment and inference using Bayesian networks. *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans.*, 1996.
[16] M. Schiffman. Common Vulnerability Scoring System (CVSS). *URL https://www.first.org/cvss/v3.1/specification-document*, 2019.
[17] ISO/IEC. Common Criteria for Information Technology Security Evaluation Part 2. *Security*, (September), 2012.
[18] Kim Fenrich. Securing your control system: the" cia triad" is a widely used benchmark for evaluating information system security effectiveness. *Power Engineering*, 112(2):44–49, 2008.
[19] Shawn Hernan, Scott Lambert, Tomasz Ostwald, and Adam Shostack. Uncover security design flaws using the stride approach (2006). *URL http://msdn. microsoft. com/en-gb/magazine/cc163519. aspx*, 15.

Design Space Exploration for Secure IoT Devices and Cyber-Physical Systems

LUKAS GRESSL, Institute of Technical Informatics, Graz University of Technology
CHRISTIAN STEGER, Institute of Technical Informatics, Graz University of Technology
ULRICH NEFFE, NXP Semiconductors Austria GmbH

With the advent of the Internet of Things (IoT) and Cyber-Physical Systems (CPS), embedded devices have been gaining importance in our daily lives, as well as industrial processes. Independent of their usage, be it within an IoT system or a CPS, embedded devices are always an attractive target for security attacks, mainly due to their continuous network availability and the importance of the data they handle. Thus, the design of such systems requires a thorough consideration of the various security constraints they are liable to. Introducing these security constraints, next to other requirements, such as power consumption, and performance increases the number of design choices a system designer must consider. As the various constraints are often conflicting with each other, designers face the complex task of balancing them. System designers facilitate Design Space Exploration (DSE) tools to support a system designer in this job. However, available DSE tools only offer a limited way of considering security constraints during the design process. In this paper, we introduce a novel DSE framework, which allows the consideration of security constraints, in the form of attack scenarios, and attack mitigations in the form of security tasks. Based on the descriptions of the system's functionality and architecture, possible attacks, and known mitigation techniques, the framework finds the optimal design for a secure IoT device or CPS. Our framework's functionality and its benefits are shown based on the design of a secure sensor system.

CCS Concepts: • **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability.

Additional Key Words and Phrases: datasets, neural networks, gaze detection, text tagging

ACM Reference Format:
Lukas Gressl, Christian Steger, and Ulrich Neffe. 2018. Design Space Exploration for Secure IoT Devices and Cyber-Physical Systems. *J. ACM* 37, 4, Article 111 (August 2018), 24 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Cybersecurity has become an increasingly important factor when considering the utilization of Internet of Things (IoT) devices and Cyber-Physical Systems (CPS) for industries. Numerous large scale attacks on power plants, steel factories, and other industrial facilities have been reported in recent years [2, 18]. Especially the utilization of CPS in the form of smart sensors, connected continuously to control systems, opened new ways of attacking these facilities. Therefore, the design of IoT devices and CPS must consider all possible security attacks targeted on them, especially

Authors' addresses: Lukas Gressl, Institute of Technical Informatics, Graz University of Technology, gressl@tugraz.at; Christian Steger, Institute of Technical Informatics, Graz University of Technology, steger@tugraz.at; Ulrich Neffe, NXP Semiconductors Austria GmbH, ulrich.neffe@nxp.com.

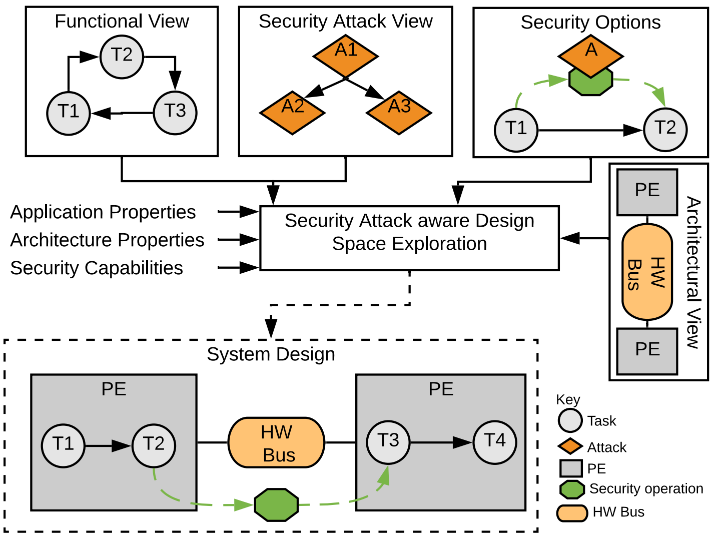


Fig. 1. System design exploration framework overview.

when operating in the industrial environment. As the industrial environment demands fast reaction times and low maintenance costs, the used CPS must guarantee high performance and low power consumption at the same time [25]. The consideration of the various constraints during the design process is a challenging problem for system designers, who must decide which functionality they implement on what device. Finding the best solution among all the alternatives means solving a multi-objective optimization problem, a task too complex to be performed manually.

System designers commonly use Design Space Exploration (DSE) frameworks to support the designer in this task. These frameworks take as an input the functional description of the system, the descriptions of possible system architectures, and application and architecture characteristics. Based on these inputs, such frameworks find the optimal architecture selection and function mapping. Classical DSE frameworks optimize for power or performance [27]. Other frameworks additionally consider security constraints either on an abstract or detailed level [32].

In this paper, we present a novel method for the integration of security constraints and security capabilities into the DSE. We present a framework extending the classical DSE by integrating security constraints described as Bayesian Network Attack Graphs. It is based on the DeSyDe framework [27–29], and supports designers in finding the optimal solution for a device under design using four distinct views. The functional view lets the designer describe the system's behavior in the form of a task graph (i). The architectural view describes the platform options of the device under design (ii). The security attack view describes the view of a potential attacker on the system (iii). The security options describe the various mitigation techniques offered by the system to counter potential attacks (iv). These views serve as inputs to the framework. Based on these inputs, the framework finds the optimal selection of the system components, mapping of tasks to the selected components, and choice on the security mitigation techniques fulfilling the security constraints and optimizing the overall system's performance or power consumption. Figure 1 shows the basic overview of the tool, its distinct views, and the solution produced by the framework.

With the approach described in this paper, we make the following contributions. To the best of the first to explicitly select security operations based on the probability of security attacks (ii). We show that a calculation of the attack probabilities based on the task mappings is feasible and can be optimized (iii).

This work is an extension of the framework presented in [8]. Compared to the already presented work, we completely reworked the use case in Section 5, using a more sophisticated and expanded example to present the functionality and feasibility of our framework. Furthermore, we changed the description of the mapping between the different modeling perspectives to a more generic rule set representation. We emphasize the depiction of the solution space, showing the number of secure solutions as opposed to the number of insecure ones. Finally, we discuss the framework's scalability using the extended use case.

The rest of the paper is structured as follows: in Section 2, various contributions in DSE, security modeling and attack description are discussed; Section 3 describes the modeling approach for designing secure systems; Section 4 explains the framework's implementation; Section 5 shows the framework solving an example use case; Section 6 concludes this paper and gives an outlook on future work.

2 RELATED WORK

Information security modeling and threat analysis have attracted much interest in research and industry in the last decades. To consider security measures in the design phase of products, various modeling tools and languages, such as the Unified Modeling Language, integrate extensions allowing the modeling of security [14]. Such tools support the designers in reflecting, e.g., security protocols within the system's behavior. However, they do not consider the system's hardware architecture, nor the security's performance overhead, nor the attacker's view on the system under design. The framework described in this paper integrates security attacks and the architecture's security capabilities into the design flow, automatically proposing security operations.

Network security analytics use the integration of the attacker's view to assessing the security of networks. Feng et al. [4] and Poolsappasit et al. [26] describe the representation of possible attackers using attack trees. Both authors use Bayesian Networks to assess the way security attacks propagate through the systems under design. These so-called Bayesian Network Attack Graphs (BNAGs) can be used to localize security vulnerabilities in the overall system and assess the efficiency of security measures to reduce the system's information security risk [4]. These BNAGs are described by security experts or can be built from historical security incidents [26]. The framework presented here combines these BNAGs with the functional and architectural description of the overall system. These additional views allow the selection of hardware components fulfilling the security constraints and the selection of feasible security operations. Safety and risk assessment use similar graph-based approaches. Among these methods, especially the attack tree analysis and the fault tree analysis are well established in both the industry and science [3].

Numerous works in literature have described classical DSE tools for embedded systems. These tools consider the power and performance of the individual implementation alternatives of the system's behavior on distinct components. The tools select the optimal system components, functionality implementations, and scheduling based on these power and performance characteristics [16, 27–29].

Knerr [16] investigate different approaches to using heuristic algorithms for solving the multi-objective optimization problem of finding the performance and power optimal system partitioning and task mapping for embedded system designs. Rosvall et al. [16, 27–29] solve this optimization problem using a constraint programming approach. With this approach, they can find the

Several works extended these classical optimization goals by functional safety and cybersecurity. Related projects focusing on the integration of cybersecurity into DSE divide into two categories: (i) works putting a clear focus on one specific problem at hand [10, 13, 15, 23, 35, 36] and (ii) works introducing security aspects into the design space on an abstract level [14, 17, 19–22, 32, 37].

The works in category (i) optimize the throughput of special bus systems [36], seamlessly integrate security surveillance tasks into an already existing task schedule [10], or consider specific network security schemes for system integration [15]. They consider the secure design of CAN bus systems [23] and optimizing the quality of service (QoS), quality of confidentiality (QoC), and the intrusion detection accuracy (IDA) of embedded systems [13].

Xie et al. describe the signal packing problem of CAN-based embedded sensor networks as an optimization problem in their work. Their goal is to find the optimum between message payload size and the size of the message authentication code (MAC). This MAC size heavily influences the communication's security and the packet size, thus decreasing its performance. The authors solve this problem by employing DSE techniques to the available options[35, 36].

Kang [15] describes a framework to support designers regarding the security mitigation techniques used for networked systems. These techniques come in the form of security policies. The framework allows the evaluation of design candidates against a given system description and filtering valid design candidates satisfying the policies. Their work focuses on the validation of design candidates rather than optimizing the system design.

Hasan et al. [10] consider the integration of security surveillance tasks into an already existing task schedule on multicore-systems. These tasks detect the attacker's intrusion into the system. As the intrusion detection is time-consuming, the integration of such tasks can break the overall task schedule and break real-time constraints of distinct real-time tasks. Hence, the integration of security surveillance tasks must be performed so that no timing constraints are broken. The authors perform this integration using a DSE-based approach.

Lukasiewycz et al. [23] consider the security-aware DSE of a CAN-based embedded system. The authors focus on secure embedded systems used as vehicle platforms shared within the whole vehicle fleets. Thereby, they aim at securing the message headers transmitted via the CAN bus system. The header of CAN messages is usually transmitted in plain, as the arbitration method cannot be performed on encrypted headers. The plain headers can be used by attackers to assume the system's internal state. To mitigate such attacks, the authors propose the obfuscation of the identifiers. As this obfuscation decreases the communication performance, the authors apply a DSE of different approaches to find the optimal obfuscation technique.

Jiang et al. [13] focus on finding an optimum considering QoS, QoC, and IDA in embedded systems. The QoC increases with the number of encryption rounds. The QoS increases with the time available for executing general-purpose tasks, and the IDA increases with the number of executions of intrusion detection tasks. Naturally, the increase of the QoC and the IDA decreases the QoS. The authors use DSE to find the optimal solution considering constraints on QoS, QoC, and IDA.

The second category's projects perform a mapping of tasks to hardware platforms based on security requirements and security hardware extensions [32] or integrate security mechanisms into industrial control loops [37]. Other projects focus on the design of secure processor architectures [22], formalize security constraints for the design of embedded systems [19–21], or offer security extension to widespread design tools [14, 17].

Stierand et al. [32] present an approach to integrating security constraints into the DSE of embedded systems. Their approach allows integrating security assets used by the system under

Zheng et al. [37] focus on the control performance and the schedulability of embedded systems while considering security requirements. They assume the attacker learns the states of the internal control tasks of the system under design. The attacker learns the state by eavesdropping on exchanged messages. The more messages the attacker can eavesdrop, the higher the probability that the internal state is disclosed. Encrypting the exchanged messages prohibits their disclosure but decreases the system's performance. The authors propose an approach to find the optimal solution considering the system's performance and the attacker's ability to guess the system's state.

In [22], the authors use a high-level analytical model to design secure processor architectures. They use the model to investigate practical designs considering security requirements and evaluate the trade-offs of different security modules and their performance.

Lin et al. [19] present a DSE method to formalize security constraints for embedded systems and use them during DSE. Their method supports the mapping of system functions to architecture components regarding their security properties and supported security mechanisms. Furthermore, the presented approach considers the system's performance risk values defined by the designers. The usability of the approach is shown based on the security enhancement of a controller are network [21] and the design of a Time Division Multiple Access-based real-time system [20].

Jurjens [14] present a security extension to the well-know Unified Modeling Language, heavily used for designing systems. With this extension, Jurjens supports designers in modeling security protocols and general system security behavior. Furthermore, the framework developed by the author considers the system's architecture and the security's performance.

The SysML-sec project [17] integrates cryptographic mechanisms during hardware/software partitioning based on predefined attack scenarios. Compared to our framework, the SysML-sec project does not allow the integration of the attacker's perspective into the system design, which improves the level of detail with which a system under design can be modeled. Furthermore, our framework also considers the secret keys' placement, which is not supported by SysML-sec.

The works of category (i) deal with particular problems. They do not consider the system's whole design space. In contrast, the framework presented here delivers a holistic approach to considering the system's performance, power consumption, and security under design. Unlike the framework presented here, the contributions in (ii) do not consider the secret keys' placement on the target platform or the probability of successfully executing attacks. These projects consider security requirements and capabilities on an abstract level and do not detail security vulnerabilities or mitigation techniques.

The DSE of safety-critical systems considers a similar problem space when introducing safety-relevant aspects [6, 24, 30, 34]. Like security constraints, safety constraints restrict critical tasks' mapping to especially enhanced components or enforce a redundant implementation of specific components. DSE frameworks in this realm are in high demand for the industry and especially for automotive companies. As security and safety risk mitigation is implemented in different ways, related projects considering systems' safety are not discussed in detail.

The framework presented in this paper introduces security attacks and mitigation techniques on a detailed level, without losing the overall view on the whole system design space. Thereby, it focuses on information security. We see the framework between the related projects focusing on detailed problems and the abstract approaches, attempting to close this gap in the research on security-aware DSE. The framework supports designers in deciding what functionality to implement on which architectural component, considering performance, power consumption, security attacks, and security capabilities.

3 DESIGN VIEWS

The system representation consists of four distinct views: (i) the functional view, (ii) the architectural view, (iii) the security options view, and (iv) the security attack view. These views are used by designers to describe the distinct design options for the system, considering the different perspectives. With these perspectives, a design space is spanned, serving as an input to the framework. In this design space, the framework searches for optimal solutions considering the system's security, performance, and power consumption.

3.1 Functional and Architectural View

Task graphs describe the functionality of the system. A task graph represents the system's tasks as nodes ($t \in T$) and their communication as edges ($\epsilon_{t_{src},t_{dst}} \in E$), connecting two tasks. The task graph is a directed graph, where each edge describes the communication from t_{src} to t_{dst}. This type of functional representation is commonly used for DSE tools [16], as they are also capable of expressing synchronous data flows [28]. The approach presented here demands the task graph to be acyclic. To integrate the security constraints into the DSE, the framework presented here enriches the task graph with data entities ($d \in D$), and operations performed on them by the tasks. These operations (o) can comprise receiving, transmitting, reading, and writing the data or other operations. Each data entity is characterized by its size and a set of security requirements (sr), defining its confidentiality, authenticity, or other security properties. These security properties classify the data entities as security assets that need protection as potential attackers aim at compromising them [7]. Multiple tasks might work on the same piece of data. Hence, this data entity must be passed from task to task, using the edges connecting them. If a data entity d_z is used by both t_{src} and t_{dst}, it is exchanged via the edge $\epsilon_{t_{src},t_{dst}}$.

The architectural view comprises a set of processing elements (PEs) connected by bus systems. A communication bus physically connects two or more PEs. Both PEs and bus systems contain certain modes coming with distinct characteristics. These characteristics describe the chip area, costs, power consumption, and performance. A PE's mode's performance depends on the worst-case execution time (WCET) of the tasks it allocates. The WCET describes the execution time of a task implemented on a PE running in a specific mode. The system designers estimate the WCET for each mapping they want to be explored by the framework. The performance of a communication bus' mode is described by its data rate and the data entities it transports. The framework uses the combination of PEs and their modes to describe distinct hardware components ($pe(m)$). Therefore, the PE in the architectural description can be seen as a placeholder in which hardware components, described by the PE's modes, can be placed. Each PE mode defines a set of security capabilities ($secCap$), describing the PE's capability to encrypt, authenticate, and tamper safe storage. Additionally, the PE modes are annotated with implementation vulnerability risks, defining a potential attacker's probability to find a vulnerability to bypass the security capability. Furthermore, the communication delay depends on the task graph's edges' mapping to the platform's communication buses. What edges map to what buses depends on the mapping of the tasks to the hardware components connected by the distinct communication buses. The buses' communication delay is then calculated by its communication speed and the data entities exchanged by its mapped edges. Figure 2 depicts the components of the functional and architectural view. The task graph and the platform model build on the DeSyDe framework. We extended the task graph with the operations and the data

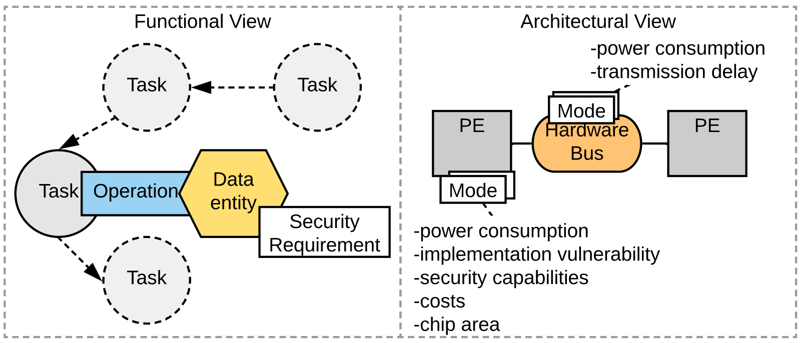


Fig. 2. Functional and platform architecture view.

3.2 Security Option and Attack View

Modeling security threats on systems is widely used for describing the security of network [5, 26, 33]. These models build on the combination of attack graphs and Bayesian networks, forming so-called Bayesian network-based attack graphs (BNAGs) [11, 31]. Each node in the BNAG represents a distinct security attack and knows two states ($state_{no-success}$ and $state_{success}$), defining if the attack has been executed successfully or not. The state transition $state_{no-success} \rightarrow state_{success}$ describes the successful execution of an attack (represented as attack node (an)) with a distinct probability. The BNAG models the dependencies of the security attacks by the directed edges between the nodes and the conditional distribution tables (CDTs). The CDT of each node represents its attack success probabilities considering the states of its parent nodes. The leaves of the BNAG describe attack goals on the overall system. The success probabilities of any attack in the BNAG, including the attack goals, are calculated using the Bayesian chain rule (1). This attack success probability is denoted $P(an)$.

$$P(an_1, ..., an_n) = \prod_{i=1}^{n} P(an_i | Pa[an_i]) \tag{1}$$

Considering the example BNAG depicted in Figure 3, the attack success probability of attack 2 can be calculated as $P(an2) = P(an2_1|P(an1_1)) + P(an2_1|P(an1_0))$, with $an1$ and $an2$ naming the attacks and the subcaptions 1 and 0 representing success and no-success states. Hence, $P(an2) = 0.7 * 0.6 + 0.3 * 0.4$. The system designers assign each attack goal a success probability threshold stating the maximal probability with which the attacker might reach his goal. The attack types used by the framework are derived from the STRIDE threat model [12], focusing on spoofing, tampering, and information disclosure.

We integrate the attack view into the system design by assigning attacks to tasks represented in the task graph. Each attack node in the BNAG attacks the data entities used by the task it aims at. Each attack furthermore has a distinct attack-type (at) defining what security requirement of the handled data it aims to break. The task's data entities being victims of a potential attack is also depicted in Figure 3.

In the security options view, the system designers describe the available security functions, feasible to counter security attacks. If the task uses security functions to protect its data handling, the attacker must first break the security function before being able to attack the data entity itself.

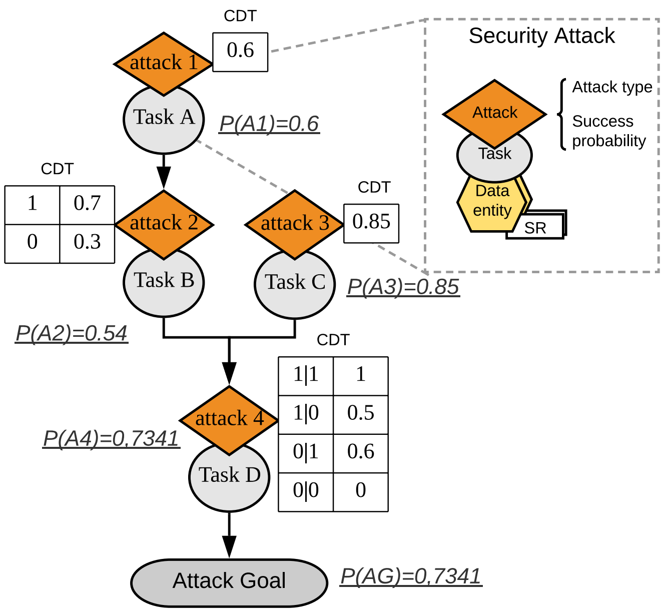


Fig. 3. BNAG example with detailed model of an attack node.

the *secFunc* itself. Modeling a cryptographic function (encryption, authentication, etc.) as a *secFunc* also includes the declaration of the used secret keys. The designer further categorizes these keys by defining their validity time. The validity time allows the designer to add the information if a *secFunc* uses, e.g., a session key, master key, etc. The validity time influences the attacker's motivation to compromise the *secFunc*, as, e.g., compromising a master key valid over the whole product lifetime has a much greater impact on the overall system's security than merely compromising a session key.

Additionally to the attacks on the *secFunc*, each secret key comes with an attack aiming at its disclosure. Also, for those attacks, the key's validity time determines the motivation of the attacker disclosing it. Any key used in the system under design might derive from another key. Experts must estimate the probabilities of the attacks aiming at victim tasks, *secFunc*, and keys. The experts must consider the attacker's capabilities and motivation for each attack. Judging the motivation of the attacker, the experts must take into account the value of the data entities. Additionally, to the attack graph, the used *secFunc* also influence the system's performance, as each *secFunc* comes with additional computational overhead.

3.3 Integration of Distinct Views

Based on the inputs from the functional, architectural, attack, and security options view and the optimization goal, the framework selects the architecture blocks, chooses the *secFunc* including the used key, and maps the tasks to architecture blocks. Thereby, it produces the solution with the lowest success probability goals (security optimal), the solution with the best performance still satisfying all security thresholds (security/performance optimal), or the solution with the best performance, neglecting security.

$pe(m)$) and must be defined to allow a successful task to PE mapping (denoted by $\rho(t, pe(m))$). The task to PE mapping is further restricted by the physical connections of the architecture platform. Given two connected tasks (t_1 and t_2) the mappings $\rho(t_1(pe_x(m_y)))$ and $\rho(t_2(pe'_x(m'_y)))$ are only valid if pe_x and pe'_x are physically connected. The mapping of t_1 and t_2 on the same PE is always valid.

3.3.2 Security Constraints Calculation. Additionally, to the general task mapping constraints, all task PE mappings must fulfill the system security constraints. Two requirements must be satisfied by the task-to-PE-mapping to fulfill these security constraints. First, the framework maps each task only to those $pe(m)$, which support the task's security functions. This mapping restriction ensures the feasibility of the needed security functions. Second, no attack goal's success probability is allowed to exceed its threshold.

The mapping between security functions (*secFunc*), security capabilities (*secCap*), operations (o), security requirements (sr), and attack-types (at_t) is described by a rule set. This rule set can be adapted and extended by the designers. Table 1 describes the framework's basic rule set.

Table 1. Rule set describing the relations between security functions (*secFunc*), security capabilities (*secCap*), operations (o), security requirements (sr), and attack-types (at_t).

secFunc from o and sr	at to sr	*secCap* supports *secFunc*
$sf_{enc} = (r \vee w) \wedge sr_{conf}$	$at_i \mapsto sr_{conf}$	$sc_{enc} = sf_{enc} \wedge ext$
$sf_{auth} = (r \vee w) \wedge sr_{auth}$	$at_s \mapsto sr_{auth}$	$sc_{sign} = sf_{auth} \wedge ext$
$sf_{ss} = st \wedge (sr_{auth} \vee sr_{conf} \vee sr_{int})$	$at_t \mapsto sr_{int}$	$sc_{te} = (sf_{enc} \vee sf_{auth}) \wedge int$
		$sc_{tss} = sf_{ss}$

The *secFunc* are determined by the operations said task performs on it's data entities and their sr [7]. The framework's basic set of operations o comprises reading (r), writing (w), receiving (rx), transmitting (tx) and storing (st). The basic set of sr consists of confidentiality (sr_{conf}), authenticity (sr_{auth}), and integrity (sr_{int}). The basic set of security functions comprises encryption (sf_{enc}), authentication (sf_{auth}), and secure storage (sf_{ss}).

Each PE mode can support a set of *secCap*, consisting of encryption (sc_{enc}), signing (sc_{sign}), task encapsulation (sc_{te}), and tamper-safe storage (sc_{tss})). Based on the *secCap* and the provided rule set, it is determined if the mapping $\rho(t, pe(m))$ supports all *secFunc* using the rules determining what *secFunc* needs what *secCap*. The attributes *int* and *ext* describe if the data entity's sr need to be protected for the internal usage only, or also for exchange between tasks mapped to different PEs. It is checked whether $\rho(t, pe(m))$ fulfills the security mapping constraint. Additionally to the mapping of *secFunc* to *secCap* and the deriving of *secFunc* from o and sr, the rule set also describes what attack types (at) are capable of compromising what sr. The basic set of at consists of information disclosure (at_i), spoofing (at_s), and tampering (at_t). The sets o, sr, at, *secFunc*, and *secCap*, as well as the rules shown in table 1 are extendable by the designers.

The mapping between *secFunc* and a PE mode's *secCap* further determines what kind of *secFunc* should be used by the task to protect its data entities. A *secFunc* must be supported by a *secCap* being integrated into the hardware component represented by the PE's mode ($pe(m)$). Cryptographic functions (e.g. sf_{enc} and sf_{auth}) are further described by the used keys (sf_k). What sf_k can be used on which $pe(m)$ is further restricted by the *secCap* it supports. Secret keys with a validity time covering the whole product's lifetime (e.g. master key, root certificate, etc.) are only allowed to

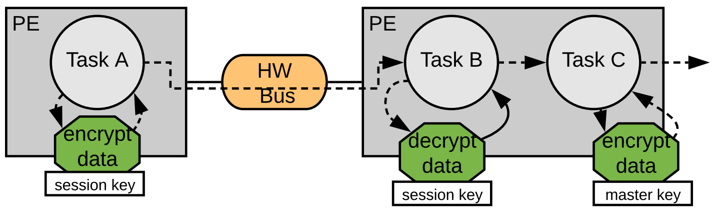


Fig. 4. Example of a connection secured by encryption spanning over multiple tasks mapped to two PEs.

secFunc is restricted by the secure connection $sec_{conn} = (\rho(t_x(pe_y(m_z))), ..., \rho(t_{x'}(pe_{y'}(m_{z'}))))$ of which it is a part of. Considering the task graph, the starting task of sec_{conn} is the first task performing a *secFunc* mapped to a $pe_y(m_z)$. The end task of sec_{conn} is the last task performing the same *secFunc*. The framework determines for each sec_{conn} if all tasks are mapped on the same PE. If so, the sec_{conn} is marked as internal (int), otherwise as external (ext). All sec_{conn} marked as ext must perform cryptographic functions using the supported secret keys sf_k to secure the sr of the used data. All sec_{conn} marked as int may also use *secFunc* internally usable (e.g. sc_{te}) . Special keys, denoted as internal keys, are only usable within the same device. Figure 4 depicts an example secure connection. The Figure shows how the tasks mapped to different $pe(m)$ use cryptographic security functions to secure the sec_{conn} leading over a communication bus.

The framework determines secure connections to calculate the influence of the used *secFunc* on the BNAG. For each $\rho(t(pe(m)))$ in sec_{conn} the attack nodes of the BNAG are determined. These attacks are mitigated by the *secFunc* if the attack's type at maps the sr of the victim task's data entities. Each task securing its data using a *secFunc* forces the attacker to break the *secFunc* enabling the original attack targeting said task. Attacks aiming at the mitigating *secFunc* are, thus, added as parents to the node attacking the task. The framework adds the parent attacks' states to the node's conditional distribution table as an `OR` function. Thus, the breaking of at least one *secFunc* would enable the attacker to perform the original attack on the victim task. Additionally, to the integration of the attacks aiming at the *secFunc* into the BNAG, the framework adds the disclosure attacks on the keys used by the cryptographic operations sf_k. The framework adds the disclosure attack on a key k_x as parent to the attack on each security operation using k_x ($sf_y(k_x)$). The conditional distribution table of the attack aiming at $sf_y(k_x)$ is extended as an `OR` function, as disclosing the k_x renders sf_y insecure. If k_x derives from a parent key $k_{x'}$, the framework adds the disclosure attack on $k_{x'}$ as a parent to the disclosure attack on k_x, again extending its conditional distribution table as an `OR` function. Hence, the framework represents the key derivation hierarchy using the disclosure attacks on the secret keys.

Figure 5 depicts an example integration of an attack aiming at a *secFunc* into an existing BNAG. The original BNAG consists of the two attacks attack 1 and attack 2, and the attack goal. To protect its data entity, task A selects a *secFunc*. With this selection, the attack aiming at the selected *secFunc*, attack a, is added as a parent to attack 1. This is represented by the additional entry into the CDT of attack 1. With this integration, the unconditional attack success probability of the attack goal $P(AG)$ is lowered from 0.75 to 0.435.

Additionally to the inclusion of the *secFunc* attack nodes into the BNAG, each $pe(m)$ comes with a distinct implementation vulnerability risk $impl_{vuln} \in \mathbb{Q} : impl_{vuln} \in [0, 1]$. This risk reduces the attack success probabilities of *secFunc* attack nodes used by the tasks mapped to said PE. This reduction is performed by multiplying the attack success probability with $impl_{vuln}$.

The assignment of the security functions' $impl_{vuln}$ risks must be performed by the security

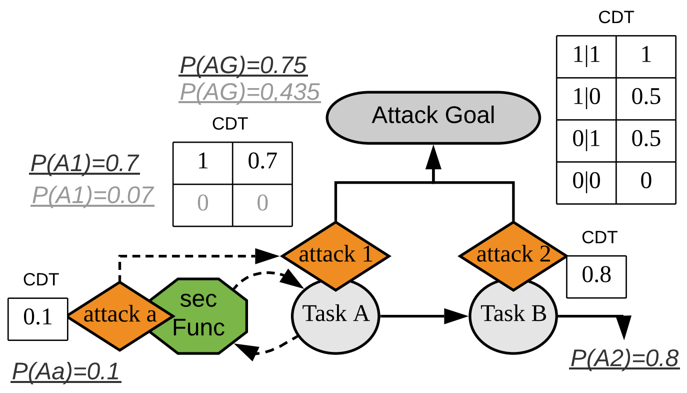


Fig. 5. Integration of attack on *secFunc* into a predefined attack scenario described as a BNAG.

of security products. The CC certification defines evaluation assurance levels (EALs), stating in what extend security experts have tested the security functions for possible vulnerabilities. These EALs range from simple, functional testing (EAL1) to formal verification, including tests (EAL7). Additionally, each product must match a defined protection profile, ensuring that it provides all expected security features [1]. The CC's official web page[1] provides a list of CC certified products and their certification documentation.

Based on the BNAG adapted with the security functions, the framework calculates the attack goals' success probabilities. It calculates the attack goals applying the Bayesian chain rule on the BNAG and merging the marginals for each attack goal. Only if each attack goal's attack probability is below its defined threshold, the overall system functionality mapping to the architecture satisfies the security constraint. The implementation of the security mapping constraints is explained in more detail in Section 4.

Considering Figure 1, one can notice how the task graph, the BNAG-based attack description, the available security options, and the architectural platform description serve as input to the framework. Based on the security-, performance-, and power-consumption-constraints, the attacks, and the security mechanisms provided by the hardware components, the framework performs a selection of the hardware components, the task mapping, and the security function selection to find suitable solutions. These solutions are then searched for the security-, performance-, or power-optimal solution, based on the designer's selected optimization goal.

3.3.3 System Performance and Power Consumption. The framework calculates a solution's delay based on the WCETs of the tasks t mapped on the selected $pe(m)$ $(wcet(\rho(t, pe(m))))$. To these WCETs, the framework adds the communication delay caused by the data entities sent over the communication buses. The data entity's size (d_{len}) is multiplied with the transmission delay (δ_{tx}) of the used bus ($\delta_{comm} = d_{len} * \delta_{tx}$). The solution's security overhead is calculated based on the delays induced by the security capabilities used for securing the data entities. These delays are either given as overall execution time, or as a delay per bit of secured data $\delta_{secCap,d_{len}}$. The hardware component's security overhead is the sum of the delays of all used security capabilities ($\delta_{sec} = \sum \delta_{secCap,d_{len}}$). These security overheads (δ_{sec}) are added to the overall performance delay ($\delta = \sum \delta + \sum \delta + \sum wcet(\rho(t, pe(m)))$). The power consumption is calculated using static

(ω_{stat}) and dynamic power consumption (ω_{dyn}). The ω_{dyn} depends on $wcet(\rho(pe(m), t))$. Again, the power needed by the security capabilities are given as an overall power consumption or per bit of secured data $\omega_{secCap,d_{len}}$. The hardware component's power consumption needed by the used security capabilities is calculated as ($\omega_{sec} = \sum \omega_{secCap,d_{len}}$). Furthermore, the framework adds the power needed by the *secCap* (ω_{sec}), and consumed power for each bus transmission to calculate the overall solution's power consumption ($\omega_{sys} = \sum \omega_{dyn} * wcet(\rho(t, pe(m))) + \sum \omega_{stat} + \sum \omega_{sec}$).

As each chosen hardware component comes with a distinct $impl_{vuln}$ stating its mitigation capability against attacks and comes with distinct security capabilities securing the attacked data entities, the increase of the attack mitigation comes with an increase in the system's performance and power consumption. Based on the given constraints formalized by the designers, the framework's main task is to find solutions satisfying all given constraints. Furthermore, by formalizing optimization goals on one of these constraints makes the framework find the optimal solution, considering the goal and all other given constraints.

4 FRAMEWORK IMPLEMENTATION

The framework is implemented based on the *DeSyDe* framework publicly available at *Github*[2]. The *DeSyDe* framework finds an optimal selection of PE modes and mapping of tasks to PEs considering power consumption and performance. The papers of Rosvall et al. [27, 29] describe the details on how the framework calculates the scheduling, performance, and power consumption. The following paragraphs describe the extensions and changes to the *DeSyDe* framework we implemented to achieve the before described calculation of an optimal and secure solution.

Task and communication bus representation: We extended the task representation by adding the definition of data entities and operations to the tasks. The *DeSyDe* tool's functional system description uses task graphs to represent synchronous data flows. We use this representation to describe the system's control flow, but with the integration of the data entities also its data flow. This data flow representation is further used to describe the communication overhead. Based on the task mapping, the data exchange channels between the tasks are put to different communication buses. Based on the used bus mode's transmission speed, the communication delay for passing the data entities over the bus is calculated. The original communication model of the *DeSyDe* model uses a TDMA-bus model where each PE can communicate with any other PE. By replacing the TDMA-bus model with the dedicated communication buses used in our approach, we not only integrated the communication delay to the performance model but also changed the mapping of tasks to PEs, as explained in Section 3.3.1. The framework calculates the transmission delay of the communication between two tasks over a physical link by multiplying the link's encoding factor with the transmission speed, times the exchanged data size. The data entity exchanged via the physical connection is determined by the communication channel between the tasks mapped to the linked PEs. The framework uses this additional transmission delay for calculating the worst-case communication time propagating into the overall system performance, as described in [27]. The framework integrates the computational overhead induced by the security functions by adding it to the WCET of the respective tasks, which affects the system's performance.

Security constraints: The calculation of the security constraints builds upon the knowledge of security capabilities, security requirements, attack goals, and available security functions. The mapping between the security capabilities and tasks' security functions is realized in a straight forward check, coming with a computational overhead of $O(N_T^{N_{PE_M}})$, with N_T being the number of tasks in the application graph, and N_{PE_M} being the number of all modes of all PEs. The framework determines the set of all secure connections *sec* denoted *SEC* = (*sec* *sec*), with

ϕ being the maximum number of all found sec_{conn}. The calculation of the attack goals bases on the BNAG and, thus, is a computationally hard problem. The framework must solve a Boolean satisfiability problem, checking if all mappings satisfy the security mapping constraints and the attack goal thresholds are not exceeded. This computational overhead even increases by the fact that each task to PE mode mapping $\rho(t, pe(m))$ changes the BNAG by adding the respective secret key-based security function sf_k attacks with the supported keys. Hence, the recalculation of the attack goals is necessary for each task to PE mapping. Approximate inferences through sampling are capable of reducing the NP-hardness of the exact probabilistic inference [9].

Reducing the number of necessary attack goal calculations induced by the different mapping possibilities, the framework searches the BNAG for all attacks influencing the attack goals, disregarding all other attacks. It orders all PEs according to the implementation vulnerability $impl_{vuln}$ of their respective modes. The PE modes' $impl_{vuln}$ are denoted by the function $\theta(pe(m))$. The ordered set $\Theta = (\theta_1(pe_x(m_y)) \leq, ..., \leq \theta_n(pe_{x'}(m_{y'}))$ is then used for calculating the attack goals for each possible mapping $\rho(t, pe(m))$. The framework permutes all feasible $\rho(t, pe(m))$ using Θ and calculates the attack goals for each permutation described as $P = (\rho_1(t_1, pe_a(m_b)), ..., \rho_{N_T}(t_{N_T}, pe_{N_{PE}}(m_{N_m})))$, with N_{PE} being the number of all pe and N_m the number of modes of the current pe. Considering the permutations, not all of them represent feasible task mappings, as a system mapping $\rho(t_x, pe_a(m_b))$ cannot map $\rho(t_y, pe_a(m_c))$. The framework eliminates these infeasible combinations before starting the attack goal calculation. For each P the possible key sets of the used *secFunc* are determined by traversing all sec_{conn} in SEC_{conn} and checking the *secCap* of each $\rho(t, pe(m))$ of sec_{conn}. Based on *secCap* the set of possible keys K for all involved *secFunc* is determined (sf_k). In each P, the combinations of all sf_K are iterated. In each iteration, the framework forms the respective BNAG to calculate the attack goals (ag) of said iteration ($ag(i)$). At the end of each iteration, $ag(i)$ is added to the set $ag(P)$. The framework checks this set $ag(P)$ at the end of said permutation P. As the framework ordered Θ descending, it can halt the calculation of $ag(P)$ when it found a P_{insec}, not containing any $ag(i)$ satisfying the defined attack thresholds. All following P can be rendered insecure without further attack goal calculation, until a P is reached, in which at least one $\rho'(t_x, pe_{a'}(m_{b'}))$ has a $\theta'(pe_{a'}(m_{b'}))$ smaller than $\theta(pe_a(m_b))$ of $\rho(t_x, pe_a(m_b))$ in P_{insec}. After that, the algorithm continues calculating the security goals for all following P until reaching the next P_{insec}.The algorithm's computational worst case is not finding any mapping dissatisfying the security goals' thresholds. In all other cases, the algorithm will reduce the number of necessary inference calculations in the BNAG.

5 EVALUATION AND RESULTS

We used the framework presented here to design a secure indoor localization system with the support of our industry partners. The secure localization system consists of nodes and anchors. The anchors have static positions within a room. Nodes and anchors determine their distances from each other using an ultra-wideband (UWB) based ranging approach. A localization system consisting of a single or more anchors aims at localizing all nodes in its vicinity. To perform this localization, nodes and anchors exchange ranging packets. Taking the timestamps when sending and receiving these packets allows the calculation of the packet's time of flight. This time of flight then allows the calculation of the distance between the communicating devices. A system that knows the distances between a node and multiple (at least three) anchors can calculate the node's position based on triangulation.

Such systems are not only used for device and asset tracking, but also for realizing key-less entry systems. In such systems, the node functions as a key, the anchor as a lock. Lock and key perform a

Table 2. PEs with security options, containing their implementation vulnerability $impl_{vuln}$, performance (Perf) given in μs, and power consumption (PWC) in mW. The performance and power consumption for encryption sc_{enc} and authentication sc_{auth} are given for symmetric (sym) and asymmetric (asym) cryptography.

HWC	Sec. Feat.	$impl_{vuln}$	Perf/PWC		
			sc_{enc},sc_{auth}	sc_{tss}	sc_{te}
AP	HWC, HWF	0.4	sym(40/50),asym(100/90)	-/-	10/10
	SWC scp, HWF	0.5	sym(70/60)	-/-	10/10
	SWC scp, TEE	0.6	sym(70/60)	-/-	5/5
	SWC f., TEE	0.8	sym(45/40)	-/-	5/5
SE	EAL 6+	0.05	sym(110/70),asym(160/130)	50/20	20/15
	EAL 5+	0.1	sym(100/60),asym(140/120)	30/10	15/10
	EAL 4+	0.2	sym(100/50),asym(140/100)	20/10	10/10
UR	HWC, MS	0.25	sym(60/40),asym(110/100)	-/-	20/20
	HWC, HWF	0.35	sym(50/30),asym(100/90)	-/-	5/10
	HWC, TZ	0.5	sym(50/30),asym(100/90)	-/-	10/5
	SWC scp, TZ	0.6	sym(80/40)	-/-	10/5
	SWC f., TZ	0.2	sym(50/35)	-/-	5/5

to the location it secures. Secure designs of such systems are of great importance, as the lack of security considerations can lead to inadvertent incidents, such as reported here[3] [4].

The task was to design a secure localization system, given a set of alternative PE coming with different security capabilities. Each node and anchor device consists of an application processor (AP), a UWB radio (UR), and an optional secure element (SE). Each hardware component listed in table 2 is represented by a distinct PE, coming with the described modes. Furthermore, each device additionally comes with a Bluetooth Low Energy (BLE) radio, serving the message exchange of information not needed for the ranging process. The AP types come with hardware-based cryptography (HWC) or with software-based cryptography (SWC). The SWC can either be functional only (f.) or side-channel protected (scp). The task encapsulation is either supported by a hardware firewall (HWF) or by a Trusted Execution Environment (TEE). For the SE, we provided the framework with various types, coming with different Evaluation Assurance Levels (EAL). The UR can also support HWC or SWC, and come with an HWF or a Trust-Zone (TZ). Furthermore, one possible solution for the UR consists of two separate micro-controllers, where one serves distinctly as a secure zone.

Each PE comes with a distinct $impl_{vuln}$ and different supported *secCap*. Security experts rated their $impl_{vuln}$. Where possible, we used Common Criteria Certifications[5], performing the estimation on the distinct (EAL). The *seCap* comes with distinct security overhead and power consumption characteristics. We estimated these characteristics during early system design. Based on the chosen type, the distinct *secCap* comes with different security overheads and power consumption. Table 2 lists the alternative PEs, including their supported *secCap*, $impl_{vuln}$, performance and power characteristics. The functionality of the system consists of three consecutive phases:

- **Authentication phase:** In the authentication phase, both node and anchor device exchange pre-configured challenges. These challenges allow a mutual authentication between the node and anchor. Based on successful authentication, a shared session secret is derived.

[3]https://www.esat.kuleuven.be/cosic/fast-furious-and-insecure-passive-keyless-entry-and-start-in-modern-supercars/

Table 3. Security relevant tasks with their *sr*, and their WCETs given in µs.

Device	Task Name	*sr*	AP	SE	UR
Node & Anchor	create challenge (t1, t7)	*conf,auth*	80	100	90
Node & Anchor	check challenge (t6, t11)	*conf,auth*	70	150	100
Node & Anchor	create session secret (t13, t18)	*conf,auth,int*	80	140	100
Node & Anchor	check session secret (t17, t23)	*conf,auth*	60	120	80
Node	create config (t24)	*conf,auth*	50	150	70
Node	check config (t30)	*conf,auth*	30	100	40
Anchor	create status msg (t31)	*conf,auth*	50	150	100
Node	create rng poll msg (t36)	*conf,auth*	40	150	80
Anchor	create rsp msg (t42)	*conf,auth*	40	130	70
Node	create final msg (t48)	*conf,auth*	20	180	50
Node & Anchor	generate ts (t38, t44, t50)	*conf,auth*	10	50	20
Anchor	extract ts (t53)	*conf,auth*	40	90	70
Anchor	calculate distance (t54)	*conf,auth*	30	80	50

- **Session and Configuration phase:** The node and anchor device exchange their session secrets derived on the previously exchanged challenges. After establishing the session, node and anchor devices exchange their configurations. Based on these configurations, they set up a ranging session.
- **Ranging phase:** Based on the exchanged configuration, the node and the anchor device perform the ranging. The ranging phase starts with the node sending a poll-packet to the anchor. The anchor responds to this packet sending a response on the node. The node sends a final packet to the anchor. This final packet contains all sending and receiving timestamps (ts) stored by the node. The anchor device uses these ts to calculate the packets' time of flight, and, hence, the distance to the node. Based on this distance and the authorization performed earlier, the anchor grants access to the node.

The task graph depicted in Figure 6 represents the system's functionality. It includes the data entities used by each task, and the task's operations performed on these entities. The overall functionality consists of an authentication phase, a session and configuration phase, and a ranging phase. In the authentication phase, node and anchor exchange and verify their respective challenges. In the session and configuration phase, the node device requests a new session and, upon receiving a positive response, sends its configuration parameters to the anchor device. The anchor device applies the received configuration. In the ranging phase, the node and anchor device exchange ranging packets. At the end of this exchange, the anchor collects the packets' timestamps and calculates the distance between the itself and the node. If the distance drops below a certain threshold, the anchor opens the door it secures. Overall, the task graph consists of 59 nodes describing the secure ranging system's functionality. Table 3 lists the estimated WCETs of the security-relevant tasks when mapped to the distinct PEs. Furthermore, it lists the security requirements *sr* of the distinct tasks, based on the data entities they operate on.

The system's functionality must resist potential cybersecurity attacks. Figure 7 shows detailed attack scenarios, aiming at distinct phases. To model these attack scenarios, and, further, let the framework derive the system's security constraints, we made the following assumptions regarding potential attackers. The attacker is capable of sniffing both the traffic on the BLE and the UWB

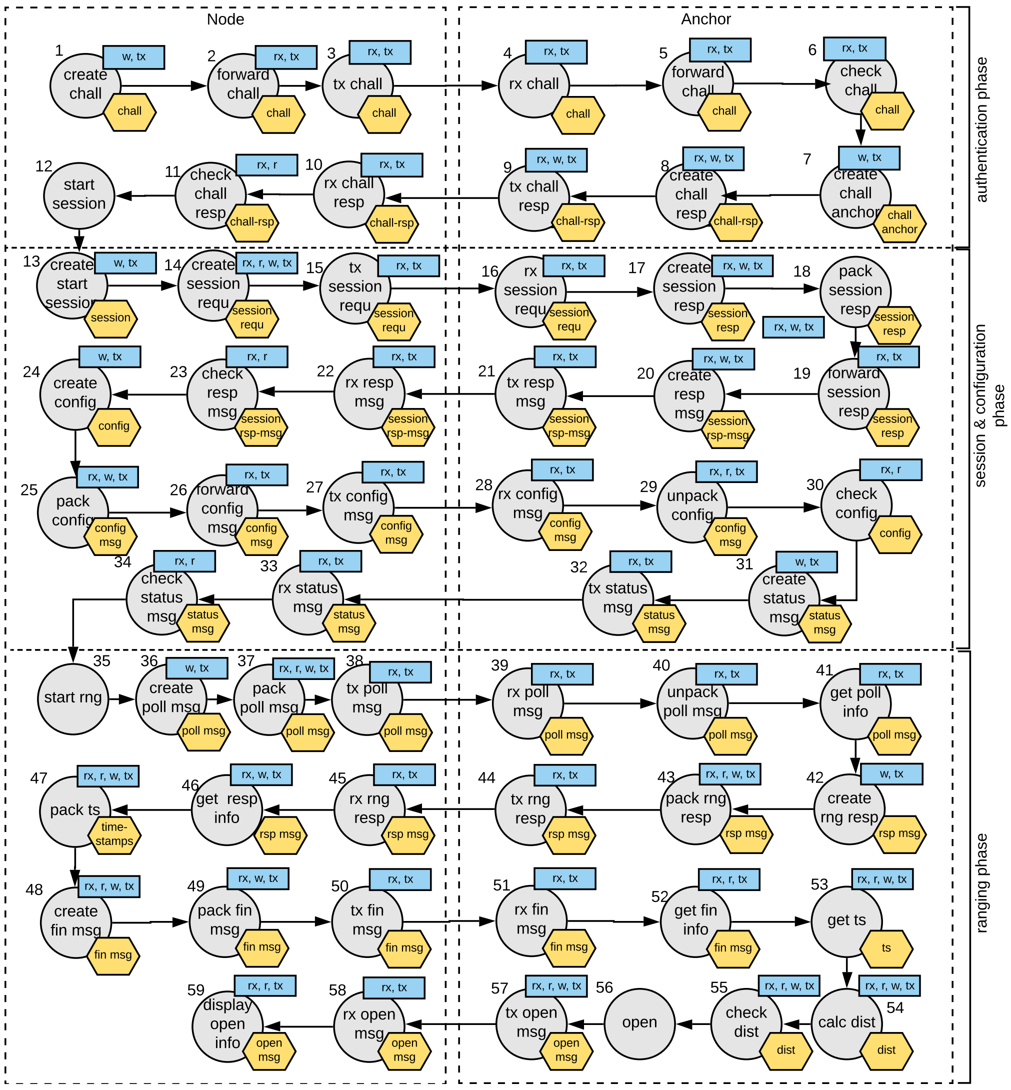

Node
Anchor
authentication phase
session & configuration phase
ranging phase
1 create chall w, tx chall
2 forward chall rx, tx chall
3 tx chall rx, tx chall
4 rx chall rx, tx chall
5 forward chall rx, tx chall
6 check chall rx, tx chall
7 create chall anchor w, tx chall anchor
8 create chall resp rx, w, tx chall-rsp
9 tx chall resp rx, w, tx chall-rsp
10 rx chall resp rx, tx chall-rsp
11 check chall resp rx, r chall-rsp
12 start session
13 create start session w, tx session
14 create session requ rx, r, w, tx session requ
15 tx session requ rx, tx session requ
16 rx session requ rx, tx session requ
17 create session resp rx, w, tx session resp
18 pack session resp rx, w, tx session resp
19 forward session resp rx, tx session resp
20 create resp msg rx, w, tx session rsp-msg
21 tx resp msg rx, tx session rsp-msg
22 rx resp msg rx, tx session rsp-msg
23 check resp msg rx, r session rsp-msg
24 create config w, tx config
25 pack config rx, w, tx config msg
26 forward config msg rx, tx config msg
27 tx config msg rx, tx config msg
28 rx config msg rx, tx config msg
29 unpack config rx, r, tx config msg
30 check config rx, r config
31 create status msg w, tx status msg
32 tx status msg rx, tx status msg
33 rx status msg rx, tx status msg
34 check status msg rx, r status msg
35 start rng
36 create poll msg w, tx poll msg
37 pack poll msg rx, r, w, tx poll msg
38 tx poll msg rx, tx poll msg
39 rx poll msg rx, tx poll msg
40 unpack poll msg rx, tx poll msg
41 get poll info rx, tx poll msg
42 create rng resp w, tx rsp msg
43 pack rng resp rx, r, w, tx rsp msg
44 tx rng resp rx, tx rsp msg
45 rx rng resp rx, tx rsp msg
46 get resp info rx, w, tx rsp msg
47 pack ts rx, r, w, tx time-stamps
48 create fin msg rx, r, w, tx fin msg
49 pack fin msg rx, w, tx fin msg
50 tx fin msg rx, tx fin msg
51 rx fin msg rx, tx fin msg
52 get fin info rx, r, tx fin msg
53 get ts rx, r, w, tx ts
54 calc dist rx, r, w, tx dist
55 check dist rx, r, w, tx dist
56 open
57 tx open msg rx, r, w, tx open msg
58 rx open msg rx, tx open msg
59 display open info rx, r, tx open msg

Fig. 6. Functional description of the secure ranging use case showing the tasks for the node and the anchor device. The numbers denote the task numbers.

the devices logically, exploiting maintenance interfaces. Based on these assumptions, the attacks in the distinct phases comprise:

- **Authentication attacks:** Sniffing the authentication exchange, extracting, faking, and injecting the faked challenges into the communication between node and anchor. Disclosing and manipulating the challenge within the devices, before their exchange.
- **Session and Configuration attacks:** Sniffing the session exchange, extracting, manipulat-

Table 4. Security functions and keys used by the system, including their attack success probabilities (*asp*).

Security Functions	Key	Key-lifetime	*secFunc asp*	Key disclosure *asp*
Cryptography	Master Key	long-term	0.4	0.5
	Internal Key	long-term	0.25	0.4
	Session Key	short-term	0.15	0.2
	Certificate	long-term	0.1	0.2
Task Encapsulation	/	/	0.25	/
Secure Storage	/	/	0.25	/

parameters in the same way, thereby manipulating the configuration of the subsequent ranging process.

- **Ranging attacks:** The attacker is also capable of sniffing the ranging process, extracting, manipulating, and injecting faked ranging packets. Furthermore, the attacker can manipulate the timestamps within the device. These attacks allow the attacker to lead the anchor into wrong distance calculations.

The overall attack scenario consists of 43 attack nodes split into four attack graphs. These graphs contain seven attack goals. None of these attack goals are allowed to exceed their distinct thresholds.

We provided the framework with a set of security functions *secFunc* the system can use to mitigate these attacks, protecting the system against them. As explained in Section 3, each *secFunc* comes with an attack stating the probability of an attacker breaking it. Furthermore, *secFunc* use keys, which further influence the attack probability of the attack targeting the *secFunc*. Table 4 lists the *secFunc* the framework can select from, including their attack probabilities of being broken by the potential attacker. For cryptographic security functions, the attack probabilities also reflect the used key's lifetime and the prospect of successfully breaking the operation.

Furthermore, it contains the attack probabilities of the distinct keys, which also depend on their lifetimes. The session key derives from either the master key or the certificate. The system uses the internal key only for communication within the same device. Although it is a long-term key, the internal key can be used by all $pe(m)$ supporting task encapsulation sc_{te}. The system uses the master, internal, and session keys for symmetric cryptography only. The certificate for asymmetric cryptography.

We provided the inputs described in here to the framework and let it compute the security-feasible solutions, considering their task mappings and system partitioning. Figure 8 shows this solution space, depicting each solution based on its overall system delay, power consumption, and the number of broken security goals. We normalized both the performance and power consumption to the solution with the best performance and lowest power consumption, respectively. The solution space regards all security-feasible solutions (solutions which map tasks in such a way that their *secFunct* are supported by the PE's *secCap*). The whole solution space consists of 543808 solutions. From these solutions, only 29232 fulfill the security constraints (have no broken security goals). Hence, the framework discarded 94, 62% of the whole solution space. Figure 9 shows the solutions ordered according to their average attack success probability (ap_{avg}). The figure shows that the ap_{avg} of the found solutions ranges from 0.0008 to 0.00823. Most of the solutions (> 55%) have an ap_{avg} of less than 0.0033. All these solutions are at least 59.9% less likely to be successfully attacked than the most insecure solution found by the framework.

Tables 5 and 6 show the fastest, the most secure, the fastest secure, and the most power-efficient, but secure solution. Each solution states the average attack probability at the attack goals ($ap_{avg} = \frac{\sum_{i=0}^{G} ap_{avg}(i)}{G}$), where G is the number of all attack goals. It states the normalized δ_{sys} and the

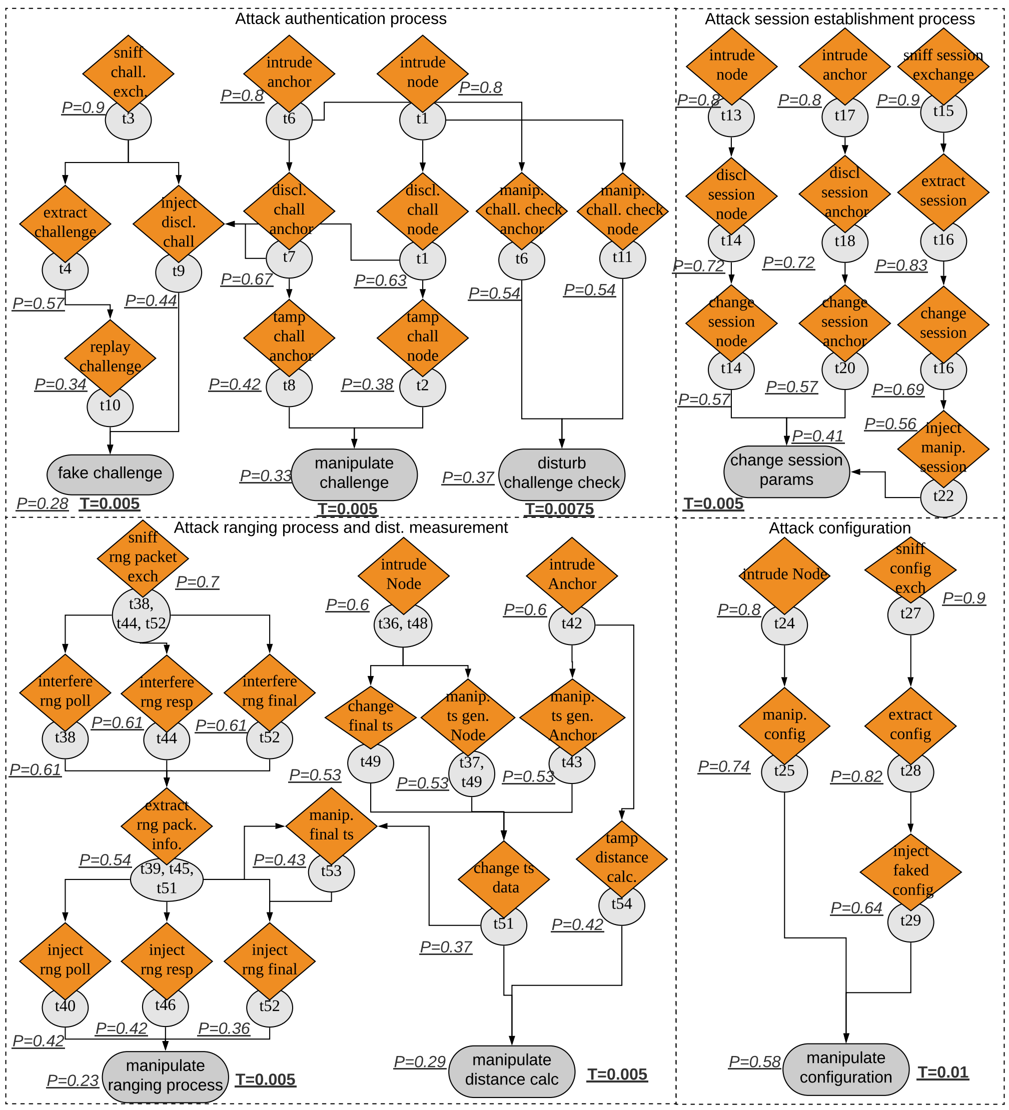


Fig. 7. Security attack scenarios described as BNAGs. The BNAGs show the unconditional probabilities of each attack node. The attack goals come with distinct thresholds.

fastest solution, it places the tasks to the PEs where they have the lowest estimated WCET. Therefore, it only puts tasks to the SEs of the anchor and node device, if there is no other option (feasibility of *secFunc* must be guaranteed). It also tries to put as many tasks to the same PE as possible, as this reduces the communication time induced by the bus system. Looking at the cryptographic *secCap*, one can notice that the fastest solution aims at using symmetric cryptography only. For the most

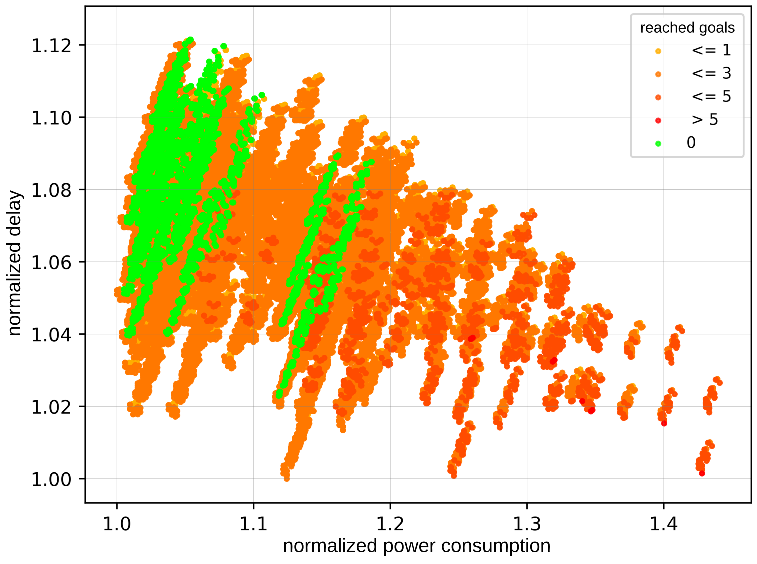


Fig. 8. Solution space identified by the framework. Solutions are ordered by their system delay δ_{sys} normalize to the fastest solutions and the system power consumption ω_{sys} normalize to the most power efficient solutions. The solutions are colored based on the number of broken security goals

aims at using the cryptographic *secFunc* with the session key as much as possible, as it comes with the lowest *asp*. For this solution, the session key derives from the certificate, as the certificate comes with a lower *asp* than the master key. Where possible, the solution uses asymmetric cryptography using the certificate. Considering secure but performance optimal/power consumption optimal solutions, the framework only selects modes for both the UR and the AP coming with HWC. Neither the performance optimal and secure nor the most power-efficient and secure solution comes with a SE with an assurance level lower than EAL5+, as this would be considered insecure. Both solutions use the session key for performing the cryptographic *secFunc*, derived from the master key.

Table 7 lists the mappings of the security-relevant tasks to the PEs in the fastest, most secure, fastest secure, and most power-efficient and secure solution. For the fastest solution, the framework maps all tasks to the AP, if possible. For the most secure solution, the framework puts all possible tasks to the SE, as it offers the lowest $impl_{vuln}$. The task mappings for the fastest secure and the most power-efficient and secure solution are equal. The different modes selected for the PEs of these solutions alone cause the differences between the δ_{sys} and ω_{sys}. In all solutions, the framework places at least t18 and t13 on the SE of the node and anchor device, respectively. It must perform this mapping as both tasks store the session secret exchanged between the devices. As this secret must be kept safe from tampering, only the SE can store it in tamper-safe storage.

5.1 Execution Times and Scalability

We executed the framework with the experimental use case described in here on a system providing 16 GB of RAM and an Intel® Core™ i7-4600U CPU with 2.10 GHz. To examine the runtime and the

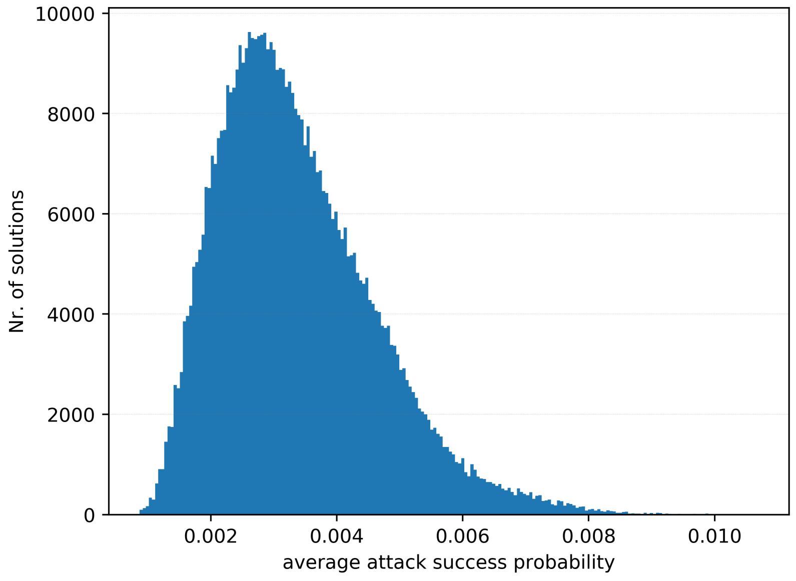


Fig. 9. Solutions found by the framework categorized according to their ap_{avg}. Stepsize of $4.9095 * 10^{-5}$

Table 5. Fastest and most secure solutions found based on ap_{avg} and δ_{sys} normalized to the system with the lowest δ_{sys}.

Device	HWC	Fastest solution	Most secure solution
Anchor	UR	SWC, f., TZ	HWC, MS
	SE	EAL 4+	EAL 6+
	AP	SWC, f., TEE	HWC, HWF
Node	UR	SWC, f., TZ	HWC, MS
	SE	EAL 4+	EAL 6+
	AP	SWC, f., TEE	HWC, HWF
ap_{avg}		0.0047	0.00087
norm δ_{sys}		~1.0	~1.09
norm ω_{sys}		~1.12	~1.03
Key Placement			
Anchor	UR	*ssk*, *sbk*	*ssk*, *sbk*
	SE	*sbk*, *smk*,	*sbk*, *ssk*
	AP	*sbk*, *smk*,	*sbk*, *cert*
Node	UR	*ssk*, *sbk*	*ssk*, *sbk*
	SE	*sbk*, *smk*,	*sbk*, *cert*
	AP	*sbk*, *smk*,	*sbk*, *ssk*

Table 6. Fastest secure and most power efficient (MPE) secure solution found based on ap_{avg}, δ_{sys} normalized to the system with the lowest δ_{sys}, and ω_{sys} normalized to the system with the lowest ω_{sys}.

Device	HWC	Fastest secure solution	MPE secure solution
Anchor	UR	HWC, HWF	HWC, TZ
	SE	EAL 5+	EAL 5+
	AP	HWC, HWF	HWC, HWF
Node	UR	HWC, HWF	HWC, TZ
	SE	EAL 6+	EAL 5+
	AP	HWC, HWF	HWC, HWF
ap_{avg}		0.0047	0.00087
norm δ_{sys}		~1.023	~1.051
norm ω_{sys}		~1.112	~1.053
Key Placement			
Anchor	UR	ssk, sbk	ssk, sbk
	SE	sbk, smk	sbk, smk
	AP	sbk, smk	sbk, smk
Node	UR	ssk, sbk	ssk, sbk
	SE	sbk, smk	sbk, smk
	AP	sbk, smk	sbk, smk

Table 7. Fastest and most secure solutions found based on ap_{avg} and δ_{sys} normalized to the system with the lowest δ_{sys}.

Device	HWC	Fastest	Most secure	Fastest secure	MPE secure
Anchor	UR	t31, t42, t44	t44	t6, t31, t42, t44, t53, t54	t6, t31, t42, t44, t53, t54
	SE	t18	t18, t31, t42, t6, t7, t17, t53, t54	t18	t18
	AP	t6, t7, t17, t53, t54	-	t7, t17	t7, t17
Node	UR	t36, t48, t38, t50	t38, t50	t11, t24, t30, t36, t38, t48, t50	t11, t24, t30, t36, t38, t48, t50
	SE	t13	t1, t11, t13, t23, t24, t30, t36, t48	t13	t13
	AP	t1, t11, t23, t24, t30	-	t1, t23	t1, t23

system partitioning. Table 8 shows the different variants (variant III consists of the whole use case) with their distinct execution times, with and without reducing the number of necessary attack goal calculations by using the break criteria described in Section 4. Variant I comes with $6,359$, variant II comes with $51,932$, and variant III comes with $4,661,264$ security-feasible combinations. Based amount of reduced time highly depends on the attack scenarios. The sooner the calculation reaches the break criteria, the more the system can reduce the execution time. The calculation time per solution is ~0.013sec for variant I, for variant III the calculation time per solution is ~0.00095sec. This difference in the calculation time per solution is due to a static ramp up overhead caused by the framework. Hence, the bigger the use case becomes, the less time the framework needs for the calculation of each solution.

With the secure ranging system use case, we show the feasibility and scalability of our security-aware DSE framework. We show how the system designer can use the framework to define security constraints using attack models and how they influence the system partitioning and task mapping of the distinct solutions. The use case shows the approach's capability of reducing the solution space by introducing these additional security constraints and the influence of the chosen security capabilities on the solution's overall performance and power consumption.

Table 8. Execution times for different variants with and without break criteria.

Mode	Variant I	Variant II	Variant III
No break crit.	1min 21.816sec	26min 6.864sec	1h 13min 57.367sec
Break crit	51.239sec	22min 34.498sec	41min 34.199sec

6 CONCLUSION AND FUTURE WORK

In this paper, we describe a novel approach for introducing security attack scenarios and security functionality selection into a system design process of embedded systems. The introduced framework allows a designer to specify the system's functionality, the possible hardware architecture components running the defined functionality, model-based attack scenarios, and a range of security functions usable by the system. The framework uses these descriptions to suggest an optimal hardware component selection, the task to component mapping, and optimal usage of security functions with respective key material. We described the functionality of the framework during design time based on a secure ranging system use case. Furthermore, we showed its feasibility for finding not only the most secure but also the, e.g., best performing and still secure solution.

The framework presented here constitutes an important step for introducing detailed security requirements into the design space exploration of embedded systems based on attack probabilities. Looking into related work security constraints and usable security functions can be formalized more specifically. We imagine the framework utilized as a tool to simplify the design process of secure embedded systems and emphasize the importance of information security at an early stage of the system design process. In this sense, the framework introduces the experience and knowledge of security experts in the design flow in an easier way. Thus, the framework supports system designers not experienced in information security during the first phases of system design. The security expert knowledge is, however, also the limiting factor in this process. The information about the assurance level of a security operation's implementation must be provided by the security experts a priori. The CC certification can only give guidance about the security level of certain secure devices. However, this guidance is not directly usable for judging what vulnerability can be found by attackers considering the security functions. As these security functions secure the tasks running on the platform, this information is critical to the overall system design. Hence, a standardized method for judging these security level of such security functions would be needed. Based on this method, the framework could store this information in a repository. Future designs

For future work, this work will be extended to support not only security attack scenarios but a complete security risk estimation. This extension would add critical and important information to the DSE process and allow the framework to consider additional constraints, such as expected financial loss caused by successfully executed security attacks, etc. Furthermore, we want to show how the framework can also be used on a large scale system, covering not only the embedded world but also higher layers, such as web servers and applications, etc.

ACKNOWLEDGMENT

Project partners are NXP Semiconductors Austria GmbH and the Technical University of Graz. This work was supported by the Austrian Research Promotion Agency (FFG) within the project UBSmart (project number: 859475).

REFERENCES

[1] 2012. Common Criteria for Information Technology Security Evaluation Part 2. (2012). https://doi.org/10.1016/S0168-3659(03)00201-3

[2] Mohammed Nasser Al-mhiqani, Rabiah Ahmad, Warusia Yassin, Aslinda Hassan, Zaheera Zainal Abidin, Nabeel Salih Ali, and Karrar Hameed Abdulkareem. 2018. Cyber-Security Incidents : A Review Cases in Cyber-Physical Systems. (2018).

[3] Paul Ammann, Duminda Wijesekera, and Saket Kaushik. 2002. Scalable, Graph-based Network Vulnerability Analysis. In *Proceedings of the 9th ACM Conference on Computer and Communications Security* (Washington, DC, USA). ACM, New York, NY, USA, 217–224. https://doi.org/10.1145/586110.586140

[4] Nan Feng, Harry Jiannan Wang, and Minqiang Li. 2014. A security risk analysis model for information systems: Causal relationships of risk factors and vulnerability propagation analysis. *Information Sciences* (2014). https://doi.org/10.1016/j.ins.2013.02.036

[5] Marcel Frigault and Lingyu Wang. 2008. Measuring Network Security Using Bayesian Network-Based Attack Graphs. (2008). https://doi.org/10.1109/COMPSAC.2008.88

[6] Sebastian Graf, Michael Glaß, Jürgen Teich, and Christoph Lauer. 2014. Multi-variant-based design space exploration for automotive embedded systems. In *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 1–6.

[7] Lukas Gressl, Christian Steger, and Ulrich Neffe. 2019. A Security Aware Design Space Exploration Framework. In *Proceedings of the Fourteenth International Conference on Systems ICONS 2019*. ThinkMind(TM) Digital Library, Valencia, Spain. http://www.thinkmind.org/index.php?view=article{&}articleid=icons{_}2019{_}2{_}20{_}40042

[8] Lukas Gressl, Christian Steger, and Ulrich Neffe. 2019. Security Driven Design Space Exploration for Embedded Systems. In *2019 Forum for Specification and Design Languages (FDL)*. IEEE, 1–8.

[9] Haipeng Guo and William Hsu. 2002. A Survey of Algorithms for Real-Time Bayesian Network Inference. *Papers from the Workshop on Real-Time Decision Support and Diagnosis Systems* 1 (2002).

[10] Monowar Hasan, Sibin Mohan, Rodolfo Pellizzoni, and Rakesh B. Bobba. 2018. A design-space exploration for allocating security tasks in multicore real-Time systems. *Proceedings of the 2018 Design, Automation and Test in Europe Conference and Exhibition, DATE 2018* (2018). https://doi.org/10.23919/DATE.2018.8342007

[11] David Heckerman and John S. Breese. 1996. Causal independence for probability assessment and inference using Bayesian networks. *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans.* 26, 6 (1996). https://doi.org/10.1109/3468.541341

[12] Shawn Hernan, Scott Lambert, Tomasz Ostwald, and Adam Shostack. 2006. Threat modeling-uncover security design flaws using the stride approach. *MSDN Magazine-Louisville* (2006).

[13] Ke Jiang, Petru Eles, and Zebo Peng. 2013. Optimization of secure embedded systems with dynamic task sets. *Proceedings -Design, Automation and Test in Europe, DATE* (2013), 1765–1770. https://doi.org/10.7873/date.2013.355

[14] Jan Jürjens. 2005. Sound methods and effective tools for model-based security engineering with UML. *Proceedings. 27th International Conference on Software Engineering, ICSE* (2005). https://doi.org/10.1145/1062455.1062519

[15] Eunusk Kang. 2016. Design Space Exploration for Security. *IEEE Cybersecurity Development Design* (2016). https://doi.org/10.1109/SecDev.2016.22

[16] Bastian Knerr. 2008. *Heuristic Optimisation Methods for System Partitioning in HW / SW Co-Design.* Ph.D. Dissertation. Vienna University of Technology.

[17] Letitia W. Li, Florian Lugou, and Ludovic Apvrille. 2017. Security-aware Modeling and Analysis for HW/SW Partitioning

[18] Chih Ta Lin, Sung Lin Wu, and Mei Lin Lee. 2017. Cyber attack and defense on industry control systems. *2017 IEEE Conference on Dependable and Secure Computing* (2017). https://doi.org/10.1109/DESEC.2017.8073874

[19] Chumg-Wei Lin and Alberto Sangiovanni-Vincentelli. 2017. *Security-Aware Design for Cyber-Physical Systems*. Springer.

[20] Chung Wei Lin, Qi Zhu, and Alberto Sangiovanni-Vincentelli. 2015. Security-Aware mapping for TDMA-based real-Time distributed systems. *IEEE/ACM International Conference on Computer-Aided Design, Digest of Technical Papers, ICCAD* 2015-January, January (2015), 24–31. https://doi.org/10.1109/ICCAD.2014.7001325

[21] Chung Wei Lin, Qi Zhu, and Alberto Sangiovanni-Vincentelli. 2015. Security-Aware Modeling and Efficient Mapping for CAN-Based Real-Time Distributed Automotive Systems. *IEEE Embedded Systems Letters* 7, 1 (2015), 11–14. https://doi.org/10.1109/LES.2014.2354011

[22] Yung Chia Lin, Chung Wen Huang, and Jenq Kuen Lee. 2005. System-level design space exploration for security processor prototyping in analytical approaches. *Proceedings of the Asia and South Pacific Design Automation Conference, ASP-DAC* 1 (2005), 376–380. https://doi.org/10.1145/1120725.1120873

[23] Martin Lukasiewycz, Philipp Mundhenk, and Sebastian Steinhorst. 2016. Security-aware obfuscated priority assignment for automotive CAN platforms. *ACM Transactions on Design Automation of Electronic Systems* 21, 2 (2016), 1–27. https://doi.org/10.1145/2831232

[24] Giovanni Mariani, Prabhat Avasare, Geert Vanmeerbeeck, Chantal Ykman-Couvreur, Gianluca Palermo, Cristina Silvano, and Vittorio Zaccaria. 2010. An industrial design space exploration framework for supporting run-time resource management on multi-core systems. In *2010 Design, Automation & Test in Europe Conference & Exhibition (DATE 2010)*. IEEE, 196–201.

[25] Lorenzo Pagliari, Raffaela Mirandola, and Catia Trubiani. 2018. Multi-modeling Approach to Performance Engineering of Cyber-Physical Systems Design. *Proceedings of the IEEE International Conference on Engineering of Complex Computer Systems, ICECCS* (2018). https://doi.org/10.1109/ICECCS.2017.22

[26] Nayot Poolsappasit, Rinku Dewri, and Indrajit Ray. 2012. Dynamic Security Risk Management Using Bayesian Attack Graphs. *IEEE Transactions on Dependable and Secure Computing* 9, 1 (2012). https://doi.org/10.1109/TDSC.2011.34

[27] Kathrin Rosvall, Nima Khalilzad, George Ungureanu, and Ingo Sander. 2017. Throughput Propagation in Constraint-Based Design Space Exploration for Mixed-Criticality Systems. *Proceedings of the 9th Workshop on Rapid Simulation and Performance Evaluation: Methods and Tools - RAPIDO '17* (2017). https://doi.org/10.1145/3023973.3023977

[28] Kathrin Rosvall, Tage Mohammadat, George Ungureanu, Johnny Oberg, and Ingo Sander. 2018. Exploring power and throughput for dataflow applications on predictable NoC multiprocessors. *Proceedings - 21st Euromicro Conference on Digital System Design, DSD 2018* (2018). https://doi.org/10.1109/DSD.2018.00011

[29] Kathrin Rosvall and Ingo Sander. 2014. A Constraint-based Design Space Exploration Framework for Real-time Applications on MPSoCs. *Proceedings of the Conference on Design, Automation & Test in Europe* (2014). https://doi.org/10.7873/DATE.2014.339

[30] Bernhard Schätz, Sebastian Voss, and Sergey Zverlov. 2015. Automating Design-Space Exploration: Optimal Deployment of Automotive SW-Components in an ISO26262 Context. In *Proceedings of the 52nd Annual Design Automation Conference* (San Francisco, California) *(DAC '15)*. Association for Computing Machinery, New York, NY, USA, Article Article 99, 6 pages. https://doi.org/10.1145/2744769.2747912

[31] Vivek Shandilya, Chris B. Simmons, and Sajjan Shiva. 2014. Use of attack graphs in security systems. *Journal of Computer Networks and Communications* 2014 (2014). https://doi.org/10.1155/2014/818957

[32] Ingo Stierand, Sunil Malipatlolla, Sibylle Froschle, Alexander Stuhring, and Stefan Henkler. 2014. Integrating the security aspect into design space exploration of embedded systems. *Proceedings - IEEE 25th International Symposium on Software Reliability Engineering Workshops, ISSREW 2014* (2014). https://doi.org/10.1109/ISSREW.2014.29

[33] Xiaoyan Sun, Jun Dai, Peng Liu, Anoop Singhal, and John Yen. 2018. Using Bayesian Networks for Probabilistic Identification of Zero-Day Attack Paths. *IEEE Transactions on Information Forensics and Security* 13, 10 (2018). https://doi.org/10.1109/TIFS.2018.2821095

[34] Sebastian Voss, Johannes Eder, and Florian Hölzl. 2014. Design Space Exploration and its Visualization in AUTOFOCUS3.. In *Software Engineering (Workshops)*. 57–66.

[35] Yong Xie, Liangjiao Liu, Renfa Li, Jianqiang Hu, Yong Han, and Xin Peng. 2015. Security-aware signal packing algorithm for CAN-based automotive cyber-physical systems. *IEEE/CAA Journal of Automatica Sinica* 2, 4 (2015), 422–430.

[36] Yong Xie, Gang Zeng, Ryo Kurachi, Hiroaki Takada, and Guoqi Xie. 2018. Security/Timing-aware Design Space Exploration of CAN FD for Automotive Cyber-Physical Systems. *IEEE Transactions on Industrial Informatics* (2018). https://doi.org/10.1109/TII.2018.2851939

[37] Bowen Zheng, Peng Deng, Rajasekhar Anguluri, Qi Zhu, and Fabio Pasqualetti. 2016. Cross-Layer Codesign for Secure Cyber-Physical Systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 35, 5 (2016). https://doi.org/10.1109/TCAD.2016.2523937

Bibliography

[1] J. Gan, “Tradeoff analysis for Dependable Real-Time Embedded Systems during the Early Design Phases,” Technical University of Denmark (DTU), 2014.

[2] L. Gressl, M. Krisper, C. Steger, and U. Neffe, “Towards Security Attack and Risk Assessment during Early System Design,” in *2020 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*, pp. 1–8, 2020.

[3] L. Gressl, U. Neffe, and C. Steger, “Design and Implementation of an HCI based Peer to Peer APDU Protocol,” in *2018 21st Euromicro Conference on Digital System Design (DSD)*, pp. 159–162, IEEE, 2018.

[4] D. Evans, “The internet of things - how the next evolution of the internet is chaging everything,” tech. rep., Cisco Internet Business Solutions Group (IBSG), 2011.

[5] P. Middleton, P. Kjeldsen, and J. Tully, “Forecast: The internet of things, worldwide, 2013,” 2013. Available at `https://www.juniperresearch.com/press/press-releases/iot-connected-devices-to-triple-to-38-bn-by-2020`.

[6] Statista, “Iot: number of connected devices worldwide 2012-2025,” 2020. Available at `https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide`.

[7] L. Atzori, A. Iera, and G. Morabito, “The internet of things: A survey,” *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[8] D. INFSO, “Networked Enterprise & RFID INFSO G. 2 Micro & Nanosystems,” *Co-operation with the Working Group RFID of the ETP EPOSS, Internet of Things in*, 2020.

[9] J. Wurm, K. Hoang, O. Arias, A. Sadeghi, and Y. Jin, “Security analysis on consumer and industrial IoT devices,” in *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 519–524, Jan 2016.

[10] W. Wahlster, “From industry 1.0 to industry 4.0: towards the 4th industrial revolution (forum business meets research),” *3rd European Summit on Future Internet Towards Future Internet International Collaborations Espoo, Finland*, vol. 31, 2012.

[11] E. A. Lee, “Cyber Physical Systems: Design Challenges,” Tech. Rep. UCB/EECS-2008-8, EECS Department, University of California, Berkeley, Jan 2008.

[12] N. Jazdi, “Cyber physical systems in the context of Industry 4.0,” in *2014 IEEE International Conference on Automation, Quality and Testing, Robotics*, pp. 1–4, May 2014.

[13] S. Yoon, H. Park, and H. S. Yoo, “Security issues on smarthome in iot environment,” in *Computer science and its applications*, pp. 691–696, Springer, 2015.

[14] M. Nasser, R. Ahmad, W. Yassin, A. Hassan, Z. Zainal, N. Salih, and K. Hameed, “Cyber-Security Incidents: A Review Cases in Cyber-Physical Systems,” *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 1, 2018.

[15] S. D. Applegate, “The dawn of Kinetic Cyber,” in *2013 5th International Conference on Cyber Conflict (CYCON 2013)*, pp. 1–15, June 2013.

[16] J. Slay and M. Miller, "Lessons learned from the maroochy water breach," in *Critical Infrastructure Protection*, 2007.

[17] A. Humayed, J. Lin, F. Li, and B. Luo, "Cyber-Physical Systems Security—A Survey," *IEEE Internet of Things Journal*, vol. 4, pp. 1802–1831, Dec 2017.

[18] S. Rehman and K. Mustafa, "Research on Software Design Level Security Vulnerabilities," *SIGSOFT Softw. Eng. Notes*, vol. 34, p. 1–5, Dec. 2009.

[19] B. Knerr and M. Holzer, *Heuristic optimisation methods for system partitioning in HW/SW Co-Design*. Citeseer, 2008.

[20] NoMagic, "SysML Plugin 19.0 LTR Documentation." Available at `https://docs.nomagic.com/display/SYSMLP190/Requirement`.

[21] NoMagic, "MagicDraw 19.0 LTR Documentation." Available at `https://docs.nomagic.com/display/MD190/Constraint`.

[22] IBM Knowledge Center, "Security concepts and mechanisms." Available at `https://www.ibm.com/support/knowledgecenter/SSFKSJ_7.5.0/com.ibm.mq.sec.doc/q009730_.htm`.

[23] D. M. Buede and W. D. Miller, *The engineering design of systems: models and methods*. John Wiley & Sons, 2016.

[24] A. K. Mandal, C. Parakash, and A. Tiwari, "Performance evaluation of cryptographic algorithms: DES and AES," in *2012 IEEE Students' Conference on Electrical, Electronics and Computer Science*, pp. 1–5, IEEE, 2012.

[25] D. S. A. Minaam, H. M. Abdual-Kader, and M. M. Hadhoud, "Evaluating the Effects of Symmetric Cryptography Algorithms on Power Consumption for Different Data Types.," *IJ Network Security*, vol. 11, no. 2, pp. 78–87, 2010.

[26] D. W. Hubbard and R. Seiersen, "How to measure anything in cybersecurity risk," 2016.

[27] S. Neema, J. Sztipanovits, G. Karsai, and K. Butts, "Constraint-based design-space exploration and model synthesis," in *International Workshop on Embedded Software*, pp. 290–305, Springer, 2003.

[28] A. Roy, D. S. Kim, and K. S. Trivedi, "Cyber security analysis using attack countermeasure trees," in *Proceedings of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research*, pp. 1–4, 2010.

[29] I. Stierand, S. Malipatlolla, S. Fröschle, A. Stühring, and S. Henkler, "Integrating the Security Aspect into Design Space Exploration of Embedded Systems," in *2014 IEEE International Symposium on Software Reliability Engineering Workshops*, pp. 371–376, Nov 2014.

[30] Y. Xie, G. Zeng, R. Kurachi, H. Takada, and G. Xie, "Security/Timing-aware Design Space Exploration of CAN FD for Automotive Cyber-Physical Systems," *IEEE Transactions on Industrial Informatics*, no. 8, 2018.

[31] L. Gressl, C. Steger, and U. Neffe, "A Security Aware Design Space Exploration Framework," in *ICONS 2019 The Fourteenth International Conference on Systems*, 2019.

[32] L. Gressl, C. Steger, and U. Neffe, "Consideration of Security Attacks in the Design Space Exploration of Embedded Systems," in *2019 22nd Euromicro Conference on Digital System*

Design (DSD), pp. 530–537, IEEE, 2019.

[33] L. Gressl, M. Krisper, C. Steger, and U. Neffe, "Towards an automated exploration of secure iot/cps design-variants," in *International Conference on Computer Safety, Reliability, and Security*, pp. 372–386, Springer, 2020.

[34] L. Gressl, C. Steger, and U. Neffe, "Security Driven Design Space Exploration for Embedded Systems," in *2019 Forum for Specification and Design Languages (FDL)*, pp. 1–8, IEEE, 2019.

[35] L. Gressl, A. Rech, C. Steger, A. Sinnhofer, and R. Weissnegger, "Security based design space exploration for cps," in *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, SAC '20, (New York, NY, USA), p. 593–595, Association for Computing Machinery, 2020.

[36] L. Gressl, A. Rech, C. Steger, A. Sinnhofer, and R. Weissnegger, "A Design Exploration Framework for Secure IoT-Systems," in *2020 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA)*, pp. 1–8, 2020.

[37] L. Gressl, C. Steger, and U. Neffe, "Design Space Exploration for Secure IoT Devices and Cyber-Physical Systems," 2020. accepted at the *ACM Transactions on Embedded Computing Systems*.

[38] L. Gressl, C. Steger, and U. Neffe, "Message Encapsulation Pattern," in *Proceedings of the 23rd European Conference on Pattern Languages of Programs*, pp. 1–6, 2018.

[39] F. Balarin, P. Giusto, A. Jurecska, M. Chiodo, C. Passerone, E. Sentovich, H. Hsieh, L. Lavagno, B. Tabbara, A. Sangiovanni-Vincentelli, *et al.*, *Hardware-software co-design of embedded systems: the POLIS approach.* Springer Science & Business Media, 1997.

[40] T. Basten, E. Van Benthum, M. Geilen, M. Hendriks, F. Houben, G. Igna, F. Reckers, S. De Smet, L. Somers, E. Teeselink, *et al.*, "Model-Driven Design-Space Exploration for Embedded Systems: The Octopus Toolset," in *International Symposium On Leveraging Applications of Formal Methods, Verification and Validation*, pp. 90–105, Springer, 2010.

[41] F. Herrera and I. Sander, "Combining analytical and simulation-based design space exploration for time-critical systems," in *Proceedings of the 2013 Forum on specification and Design Languages (FDL)*, pp. 1–8, Sep. 2013.

[42] T. Kempf, G. Ascheid, and R. Leupers, *Multiprocessor Systems on Chip: Design Space Exploration.* Springer Science & Business Media, 2011.

[43] A. Brekling, M. R. Hansen, and J. Madsen, "Models and formal verification of multiprocessor system-on-chips," *The Journal of Logic and Algebraic Programming*, vol. 77, no. 1-2, pp. 1–19, 2008.

[44] A. Davare, D. Densmore, T. Meyerowitz, A. Pinto, A. Sangiovanni-Vincentelli, G. Yang, H. Zeng, and Q. Zhu, "A next-generation design framework for platform-based design," in *Conference on using hardware design and verification languages (DVCon)*, vol. 152, 2007.

[45] I. Sander and A. Jantsch, "System modeling and transformational design refinement in forsyde [formal system design]," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 23, no. 1, pp. 17–32, 2004.

[46] B. D. Theelen, O. Florescu, M. Geilen, J. Huang, P. Van Der Putten, and J. P. Voeten, "Software/hardware engineering with the parallel object-oriented specification language,"

in *2007 5th IEEE/ACM International Conference on Formal Methods and Models for Codesign (MEMOCODE 2007)*, pp. 139–148, IEEE, 2007.

[47] P. Marwedel, *Embedded System Hardware*, pp. 119–175. Dordrecht: Springer Netherlands, 2011.

[48] K. Rosvall and I. Sander, "A constraint-based design space exploration framework for real-time applications on mpsocs," in *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1–6, IEEE, 2014.

[49] K. Rosvall, N. Khalilzad, G. Ungureanu, and I. Sander, "Throughput propagation in constraint-based design space exploration for mixed-criticality systems," in *Proceedings of the 9th Workshop on Rapid Simulation and Performance Evaluation: Methods and Tools*, pp. 1–8, 2017.

[50] N. Khalilzad, K. Rosvall, and I. Sander, "A modular design space exploration framework for multiprocessor real-time systems," in *2016 Forum on Specification and Design Languages (FDL)*, pp. 1–7, IEEE, 2016.

[51] K. Rosvall, T. Mohammadat, G. Ungureanu, J. Öberg, and I. Sander, "Exploring power and throughput for dataflow applications on predictable noc multiprocessors," in *2018 21st Euromicro Conference on Digital System Design (DSD)*, pp. 719–726, IEEE, 2018.

[52] A. D. Pimentel, C. Erbas, and S. Polstra, "A systematic approach to exploring embedded system architectures at multiple abstraction levels," *IEEE Transactions on Computers*, vol. 55, no. 2, pp. 99–112, 2006.

[53] M. Thompson, H. Nikolov, T. Stefanov, A. D. Pimentel, C. Erbas, S. Polstra, and E. F. Deprettere, "A framework for rapid system-level exploration, synthesis, and programming of multimedia mp-socs," in *Proceedings of the 5th IEEE/ACM international conference on Hardware/software codesign and system synthesis*, pp. 9–14, 2007.

[54] C. Silvano, W. Fornaciari, G. Palermo, V. Zaccaria, F. Castro, M. Martinez, S. Bocchio, R. Zafalon, P. Avasare, G. Vanmeerbeeck, *et al.*, "Multicube: Multi-objective design space exploration of multi-core architectures," in *VLSI 2010 Annual Symposium*, pp. 47–63, Springer, 2011.

[55] D. C. Black, J. Donovan, B. Bunton, and A. Keist, *SystemC: From the ground up*, vol. 71. Springer Science & Business Media.

[56] S. Sutherland, S. Davidmann, and P. Flake, *SystemVerilog for Design Second Edition: A Guide to Using SystemVerilog for Hardware Design and Modeling*. Springer Science & Business Media, 2006.

[57] L. G. Murillo, M. Mura, and M. Prevostini, "Semi-automated hw/sw co-design for embedded systems: from marte models to systemc simulators," in *2009 Forum on Specification & Design Languages (FDL)*, pp. 1–6, IEEE, 2009.

[58] S. Kunzli, F. Poletti, L. Benini, and L. Thiele, "Combining simulation and formal methods for system-level performance analysis," in *Proceedings of the Design Automation & Test in Europe Conference*, vol. 1, pp. 1–6, IEEE, 2006.

[59] K. Lahiri, A. Raghunathan, and S. Dey, "Performance analysis of systems with multi-channel communication architectures," in *VLSI Design 2000. Wireless and Digital Imaging in the Millennium. Proceedings of 13th International Conference on VLSI Design*, pp. 530–

537, IEEE, 2000.

[60] SECTOR, STANDARDIZATION and ITU, OF, “ITU-Tx. 1205,” *Interfaces*, vol. 10, no. 20-X, p. 49.

[61] R. Von Solms and J. Van Niekerk, “From information security to cyber security,” *computers & security*, vol. 38, pp. 97–102, 2013.

[62] A. Alshboul, “Information systems security measures and countermeasures: Protecting organizational assets from malicious attacks,” *Communications of the IBIMA*, 2010.

[63] C. Schou and D. P. Shoemaker, *Information assurance for the enterprise: A roadmap to information security*. McGraw-Hill, Inc., 2006.

[64] S. Samonas and D. Coss, “The CIA strikes back: Redefining Confidentiality, Integrity and Availability in Security,” *Journal of Information System Security*, vol. 10, no. 3, 2014.

[65] N. Ferguson and B. Schneier, *Practical cryptography*, vol. 141. Wiley New York, 2003.

[66] H. G. Brauch, “Concepts of security threats, challenges, vulnerabilities and risks,” in *Coping with global environmental change, disasters and security*, pp. 61–106, Springer, 2011.

[67] M. Jouini, L. B. A. Rabai, and A. B. Aissa, “Classification of security threats in information systems.,” *ANT/SEIT*, vol. 32, pp. 489–496, 2014.

[68] Microsoft Docs, “The STRIDE Threat Model.” Available at `https://docs.microsoft.com/en-us/previous-versions/commerce-server/ee823878%28v%3dcs.20%29`.

[69] G. Dhillon, *Principles of information systems security: Texts and cases*. John Wiley & Sons Incorporated, 2007.

[70] CNSSI, No, “CNSSI 4009-2015,” *National Information Assurance (IA) Glossary*, vol. 26, 2015.

[71] NIST, SP, “NIST Publication 800-32,” *Introduction to Public Key Technology and the Federal PKI Infrastructure*, 2001.

[72] E. McCallister, T. Grance, and K. Scarfone, “NIST Special Publication 800-122: Guide to Protecting the Confidentiality of Personally Identifiable Information (PII),” *Gaithersburg, MD:[US] National Institute of Standards and Technology, US Department of Commerce*, 2010.

[73] M. G. Solomon and M. Chapple, *Information Security Illuminated*. Jones & Bartlett Publishers, 2009.

[74] M. Abomhara *et al.*, “Cyber Security and the Internet of Things: Vulnerabilities, Threats, Intruders and Attacks,” *Journal of Cyber Security and Mobility*, vol. 4, no. 1, pp. 65–88, 2015.

[75] D. Papp, Z. Ma, and L. Buttyan, “Embedded Systems Security: Threats, Vulnerabilities, and Attack Taxonomy,” in *2015 13th Annual Conference on Privacy, Security and Trust (PST)*, pp. 145–152, IEEE, 2015.

[76] B. A. Forouzan, *Cryptography & network security*. McGraw-Hill, Inc., 2007.

[77] N. Fips, “46-3: The official Document describing the DES Standard,” tech. rep., Technical report, Technical report, NIST, 1999.

[78] J. Daemen and V. Rijmen, *The Design of Rijndael AES - The Advanced Encryption Standard.* Springer Science & Business Media, 2013.

[79] ISO/IEC 9797-2:2011, "Information technology — Security techniques — Message Authentication Codes (MACs) — Part 2: Mechanisms using a dedicated hash-function," Standard, International Organization for Standardization, Geneva, CH, 2011.

[80] M. Bellare, R. Canetti, and H. Krawczyk, "Message authentication using hash functions: The hmac construction," *RSA Laboratories' CryptoBytes*, vol. 2, no. 1, pp. 12–15, 1996.

[81] J. Jonsson, "On the Security of CTR+ CBC-MAC," in *International Workshop on Selected Areas in Cryptography*, pp. 76–93, Springer, 2002.

[82] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[83] D. Hankerson, A. J. Menezes, and S. Vanstone, *Guide to Elliptic Curve Cryptography.* Springer Science & Business Media, 2006.

[84] A. J. Menezes, J. Katz, P. C. Van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography.* CRC press, 1996.

[85] D. Johnson, A. Menezes, and S. Vanstone, "The Elliptic Curve Digital Signature Algorithm (ECDSA)," *International journal of information security*, vol. 1, no. 1, pp. 36–63, 2001.

[86] T. Eisenbarth, S. Kumar, C. Paar, A. Poschmann, and L. Uhsadel, "A Survey of LightweightCryptography Implementations," *IEEE Design & Test of Computers*, vol. 24, no. 6, pp. 522–533, 2007.

[87] B. W. Lampson, "A Note on the Confinement Problem," *Communications of the ACM*, vol. 16, no. 10, pp. 613–615, 1973.

[88] B. Schneier, "Cryptanalysis of md5 and sha: Time for a new standard," *Computerworld*, vol. 19, 2004.

[89] H. Krawczyk, M. Bellare, and R. Canetti, "Hmac: Keyed-hashing for message authentication," 1997.

[90] M. M. N. Biasizzo, M. Mali, and F. Novak, "Hardware implementation of AES algorithm," *Journal of Electrical Engineering*, vol. 56, no. 9-10, pp. 265–269, 2005.

[91] M. Shand and J. Vuillemin, "Fast implementations of RSA cryptography," in *Proceedings of IEEE 11th Symposium on Computer Arithmetic*, pp. 252–259, IEEE, 1993.

[92] S. Ravi, A. Raghunathan, and S. Chakradhar, "Tamper resistance mechanisms for secure embedded systems," in *17th International Conference on VLSI Design. Proceedings.*, pp. 605–611, IEEE, 2004.

[93] S. Mao and T. Wolf, "Hardware Support for Secure Processing in Embedded Systems," *IEEE Transactions on Computers*, vol. 59, no. 6, pp. 847–854, 2010.

[94] X. Yan-ling, P. Wei, and Z. Xin-guo, "Design and Implementation of Secure Embedded Systems Based on Trustzone," in *2008 International Conference on Embedded Software and Systems*, pp. 136–141, July 2008.

[95] S. Babar, A. Stango, N. Prasad, J. Sen, and R. Prasad, "Proposed Embedded Security Framework for Internet of Things (IoT)," in *2011 2nd International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic*

Systems Technology (Wireless VITAE), pp. 1–5, IEEE, 2011.

[96] S. Levy, *Crypto: How the Code Rebels Beat the Government–Saving Privacy in the Digital Age*. Penguin, 2001.

[97] H. Krawczyk, "Cryptographic extraction and key derivation: The hkdf scheme," in *Annual Cryptology Conference*, pp. 631–648, Springer, 2010.

[98] J. Camenisch, S. Fischer-Hübner, and K. Rannenberg, *Privacy and identity management for life*. Springer Science & Business Media, 2011.

[99] M. Schiffman, "Common Vulnerability Scoring System (CVSS)," *URL https://www.first.org/cvss/v3.1/specification-document*, 2019.

[100] NIST, "National Vulnerability Database." Available at `https://nvd.nist.gov/general`.

[101] B. Schneier, "Attack trees," *Dr. Dobb's journal*, vol. 24, no. 12, pp. 21–29, 1999.

[102] L. Wang, T. Islam, T. Long, A. Singhal, and S. Jajodia, "An attack graph-based probabilistic security metric," in *IFIP Annual Conference on Data and Applications Security and Privacy*, pp. 283–296, Springer, 2008.

[103] L. Wang, A. Singhal, and S. Jajodia, "Measuring the overall security of network configurations using attack graphs," in *IFIP Annual Conference on Data and Applications Security and Privacy*, pp. 98–112, Springer, 2007.

[104] P. Xie, J. H. Li, X. Ou, P. Liu, and R. Levy, "Using bayesian networks for cyber security analysis," in *2010 IEEE/IFIP International Conference on Dependable Systems & Networks (DSN)*, pp. 211–220, IEEE, 2010.

[105] H.-K. Kong, M. K. Hong, and T.-S. Kim, "Security risk assessment framework for smart car using the attack tree analysis," *Journal of Ambient Intelligence and Humanized Computing*, vol. 9, no. 3, pp. 531–551, 2018.

[106] D. W. Hubbard and R. Seiersen, *How to measure anything in cybersecurity risk*. John Wiley & Sons, 2016.

[107] M. Krisper, J. Dobaj, G. Macher, and C. Schmittner, "RISKEE : A Risk-Tree Based Method for Assessing Risk in Cyber Security," in *Proceedings - EuroSPI 2019: European System, Software & Service Process Improvement & Innovation*, 2019.

[108] Common Criteria, "Common Criteria for Information Technology Security Evaluation Part 1 - 3. Version 3.1 Revision 5," *CC*, April 2018.

[109] B. Schätz, S. Voss, and S. Zverlov, "Automating design-space exploration: Optimal deployment of automotive sw-components in an iso26262 context," in *Proceedings of the 52nd Annual Design Automation Conference*, DAC '15, (New York, NY, USA), Association for Computing Machinery, 2015.

[110] S. Voss, J. Eder, and F. Hölzl, "Design space exploration and its visualization in autofocus3.," in *Software Engineering (Workshops)*, pp. 57–66, 2014.

[111] S. Graf, M. Glaß, J. Teich, and C. Lauer, "Multi-variant-based design space exploration for automotive embedded systems," in *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1–6, IEEE, 2014.

[112] G. Mariani, P. Avasare, G. Vanmeerbeeck, C. Ykman-Couvreur, G. Palermo, C. Silvano,

and V. Zaccaria, “An industrial design space exploration framework for supporting run-time resource management on multi-core systems,” in *2010 Design, Automation & Test in Europe Conference & Exhibition (DATE 2010)*, pp. 196–201, IEEE, 2010.

[113] Y. C. Lin, C. W. Huang, and J. K. Lee, “System-level design space exploration for security processor prototyping in analytical approaches,” *Proceedings of the Asia and South Pacific Design Automation Conference, ASP-DAC*, vol. 1, pp. 376–380, 2005.

[114] B. Zheng, P. Deng, R. Anguluri, Q. Zhu, and F. Pasqualetti, “Cross-Layer Codesign for Secure Cyber-Physical Systems,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 5, 2016.

[115] C.-W. Lin and A. Sangiovanni-Vincentelli, *Security-Aware Design for Cyber-Physical Systems*. Springer, 2017.

[116] C. W. Lin, Q. Zhu, and A. Sangiovanni-Vincentelli, “Security-Aware Modeling and Efficient Mapping for CAN-Based Real-Time Distributed Automotive Systems,” *IEEE Embedded Systems Letters*, vol. 7, no. 1, pp. 11–14, 2015.

[117] C. W. Lin, Q. Zhu, and A. Sangiovanni-Vincentelli, “Security-Aware mapping for TDMA-based real-Time distributed systems,” *IEEE/ACM International Conference on Computer-Aided Design, Digest of Technical Papers, ICCAD*, vol. 2015-January, no. January, pp. 24–31, 2015.

[118] X. Zhang, J. Zhan, W. Jiang, Y. Ma, and K. Jiang, “Design Optimization of Security-Sensitive Mixed-Criticality Real-Time Embedded Systems,” *Proc. ReTiMiCS, RTCSA*, no. Cc, 2013.

[119] J. Jürjens, “Sound methods and effective tools for model-based security engineering with UML,” *Proceedings. 27th International Conference on Software Engineering, ICSE*, 2005.

[120] L. W. Li, F. Lugou, and L. Apvrille, “Security-aware Modeling and Analysis for HW/SW Partitioning,” *Proceedings of the 5th International Conference on Model-Driven Engineering and Software Development*, 2017.

[121] Y. Xie, L. Liu, R. Li, J. Hu, Y. Han, and X. Peng, “Security-aware signal packing algorithm for can-based automotive cyber-physical systems,” *IEEE/CAA Journal of Automatica Sinica*, vol. 2, no. 4, pp. 422–430, 2015.

[122] M. Lukasiewycz, P. Mundhenk, and S. Steinhorst, “Security-aware obfuscated priority assignment for automotive CAN platforms,” *ACM Transactions on Design Automation of Electronic Systems*, vol. 21, no. 2, pp. 1–27, 2016.

[123] K. Jiang, P. Eles, and Z. Peng, “Optimization of secure embedded systems with dynamic task sets,” *Proceedings -Design, Automation and Test in Europe, DATE*, pp. 1765–1770, 2013.

[124] M. Hasan, S. Mohan, R. Pellizzoni, and R. B. Bobba, “A design-space exploration for allocating security tasks in multicore real-Time systems,” *Proceedings of the 2018 Design, Automation and Test in Europe Conference and Exhibition, DATE 2018*, 2018.

[125] E. Kang, “Design Space Exploration for Security,” *IEEE Cybersecurity Development Design*, 2016.

[126] C. J. Anderson, N. Foster, D. Kozen, and D. Walker, “Net KAT : Semantic Foundations for Networks,” pp. 113–126, 2014.

[127] T. Nelson, D. J. Dougherty, C. Barratt, and K. Fisler, "The Margrave Tool for Firewall Analysis," *Proceedings of the 24th USENIX Large Installation System Administration Conference LISA 2010*, pp. 1–8, 2010.

[128] A. Khurshid, X. Zou, W. Zhou, M. Caesar, and P. B. Godfrey, "Veriflow: Verifying network-wide invariants in real time," in *Presented as part of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*, pp. 15–27, 2013.

[129] N. Poolsappasit, R. Dewri, and I. Ray, "Dynamic Security Risk Management Using Bayesian Attack Graphs," *IEEE Transactions on Dependable and Secure Computing*, vol. 9, no. 1, 2012.

[130] N. Feng, H. J. Wang, and M. Li, "A security risk analysis model for information systems: Causal relationships of risk factors and vulnerability propagation analysis," *Information Sciences*, 2014.

[131] M. Frigault and L. Wang, "Measuring network security using bayesian network-based attack graphs," in *2008 32nd Annual IEEE International Computer Software and Applications Conference*, pp. 698–703, IEEE, 2008.

[132] X. Sun, J. Dai, P. Liu, A. Singhal, and J. Yen, "Using bayesian networks for probabilistic identification of zero-day attack paths," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 10, pp. 2506–2521, 2018.

[133] P. Ammann, D. Wijesekera, and S. Kaushik, "Scalable, graph-based network vulnerability analysis," in *Proceedings of the 9th ACM Conference on Computer and Communications Security*, (New York, NY, USA), pp. 217–224, ACM, 2002.

[134] I. Ray and N. Poolsapassit, "Using attack trees to identify malicious attacks from authorized insiders," in *European Symposium on Research in Computer Security*, pp. 231–246, Springer, 2005.

[135] C. Phillips and L. P. Swiler, "A graph-based system for network-vulnerability analysis," in *Proceedings of the 1998 workshop on New security paradigms*, pp. 71–79, 1998.

[136] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J. M. Wing, "Automated generation and analysis of attack graphs," in *Proceedings 2002 IEEE Symposium on Security and Privacy*, pp. 273–284, IEEE, 2002.

[137] W. Fornaciari, D. Sciuto, C. Silvano, and V. Zaccaria, "A design framework to efficiently explore energy-delay tradeoffs," in *Proceedings of the ninth international symposium on Hardware/software codesign*, pp. 260–265, 2001.

[138] T. Kempf, S. Wallentowitz, G. Ascheid, R. Leupers, and H. Meyr, "Analytical and simulation-based design space exploration of software defined radios," *International journal of parallel programming*, vol. 38, no. 3-4, pp. 303–321, 2010.

[139] N. Poolsappasit, R. Dewri, and I. Ray, "Dynamic security risk management using bayesian attack graphs," *IEEE Transactions on Dependable and Secure Computing*, vol. 9, no. 1, pp. 61–74, 2011.

[140] C. Schulte, G. Tack, and M. Z. Lagerkvist, *Modeling and Programming with Gecode.* May 2019. Available at `https://www.gecode.org/doc-latest/MPG.pdf`.

[141] E. A. Lee and D. G. Messerschmitt, "Static scheduling of synchronous data flow programs for digital signal processing," *IEEE Transactions on computers*, vol. 100, no. 1, pp. 24–35,

1987.

[142] K. Ito and K. K. Parhi, “Determining the minimum iteration period of an algorithm,” *Journal of VLSI signal processing systems for signal, image and video technology*, vol. 11, no. 3, pp. 229–244, 1995.

[143] K. Mikhaylov and J. Tervonen, “Evaluation of power efficiency for digital serial interfaces of microcontrollers,” in *2012 5th International Conference on New Technologies, Mobility and Security (NTMS)*, pp. 1–5, IEEE, 2012.

[144] M. Schober, “Implementation and design of a uwb-based network simulation platform for indoor localization systems,” Master’s thesis, Graz University of Technology, April 2019.

[145] D. Veit, M. Gadringer, and E. Leitgeb, “A simulation environment for uwb hardware development and protocol design,” in *2019 15th International Conference on Telecommunications (ConTEL)*, pp. 1–6, IEEE, 2019.

[146] M. Spörk, C. A. Boano, M. Zimmerling, and K. Römer, “Bleach: Exploiting the full potential of ipv6 over ble in constrained embedded iot devices,” in *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*, pp. 1–14, 2017.

[147] Common Criteria Working Group *et al.*, “Common methodology for information technology security evaluation,” tech. rep., Technical report, Common Criteria Interpretation Management Board, 2017.

[148] T. Schläpfer and A. Rüst, “Security on iot devices with secure elements,” in *Embedded World Conference, Nürnberg, 26.-28. Februar 2019*, WEKA, 2019.

[149] S. Levy, “Performance and security of ecdsa,” 2015.

[150] L. M. Raju and M. Sumathi, “Secured high throughput of 128-bit aes algorithm based on interleaving technique,”

[151] A. Rech, L. Gressl, F. Basic, C. Seifert, C. Steger, and A. Sinnhofer, “Multi-Layered IoT System Design Towards Secure End-to-End Communication,” 2020. presented at the 46th Annual Conference of the IEEE Industrial Electronics Society (IES).

[152] P. Koopman and T. Chakravarty, “Cyclic redundancy code (crc) polynomial selection for embedded networks,” in *International Conference on Dependable Systems and Networks, 2004*, pp. 145–154, IEEE, 2004.

[153] D. Neirynck, M. O’Duinn, and C. McElroy, “Characterisation of the nlos performance of an ieee 802.15. 4a receiver,” in *12th Workshop on Navigation, Positioning and Communications (WPNC)*, 2015.

[154] ISO IEC, “ISOIEC 7816-3,” no. 4, p. 27, 2006.

[155] ISO IEC, “ISOIEC FDIS 7816-4,” p. 162, 2014.

[156] ETSI, “TS 102 622 - V12.1.0 - Smart Cards; UICC - Contactless Front-end (CLF) Interface; Host Controller Interface (HCI),” 2014.

[157] N. Feng, H. J. Wang, and M. Li, “A security risk analysis model for information systems: Causal relationships of risk factors and vulnerability propagation analysis,” *Information sciences*, vol. 256, pp. 57–73, 2014.

[158] Z. Zhao, V. Tsoutsouras, D. Soudris, and A. Gerstlauer, “Network/system co-simulation for design space exploration of iot applications,” in *2017 International Conference on Em-*

bedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS), pp. 46–53, IEEE, 2017.