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.

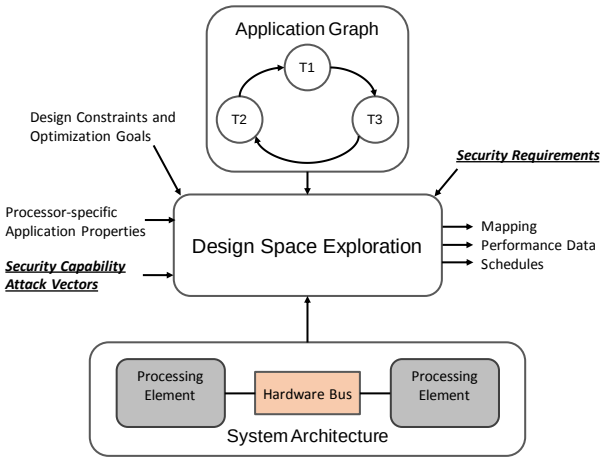

Figure 1. DSE process, based on [1] extended by security requirements, security capabilities, and attack vectors.

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

 ISBN: 978-1-61208-696-5

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.

 ISBN: 978-1-61208-696-5

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.

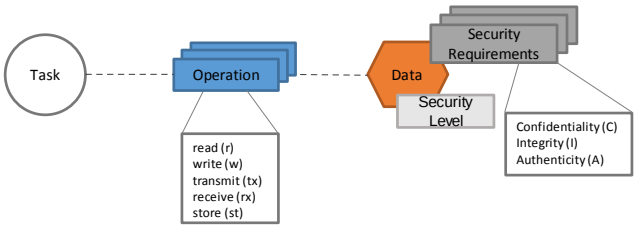

Figure 2. Representation of a process, consisting of a task, its associated data attributed with security requirements, and the task's operation performed on the data.

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.

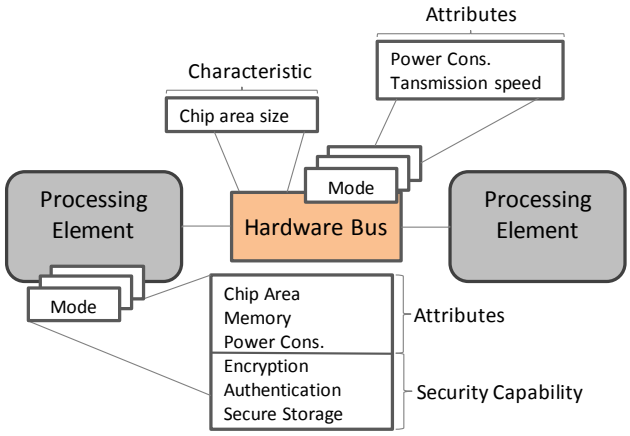

Figure 3. Hardware platform representation, consisting of PEs, connected by a hardware bus.

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.

 ISBN: 978-1-61208-696-5

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.

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

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

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

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.

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

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

 ISBN: 978-1-61208-696-5

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.

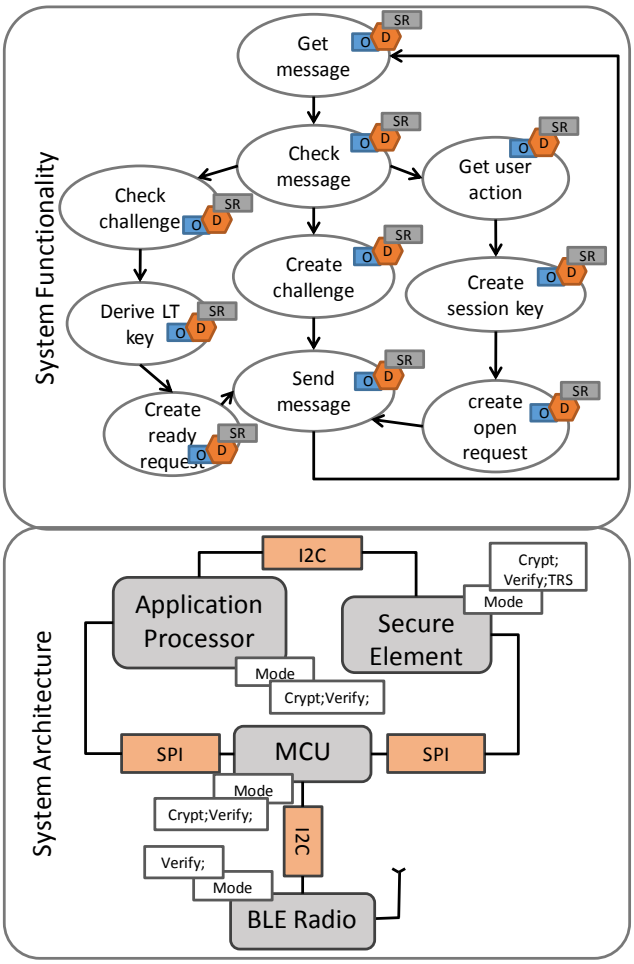

Figure 4. Evaluation example. Keyless entry system's simplified functionality which is to be mapped to a hardware platform. The PEs are connected via bus systems

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

 ISBN: 978-1-61208-696-5

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.”

 ISBN: 978-1-61208-696-5