Journal of
Low Power Electronics and Applications

Opinion

A Case for Security-Aware Design-Space Exploration of Embedded Systems

Andy D. Pimentel

Parallel Computing Systems Group, University of Amsterdam, Science Park 904, 1098 XH Amsterdam, The Netherlands; a.d.pimentel@uva.nl; Tel.: +31-20-5257578

Received: 22 June 2020; Accepted: 15 July 2020; Published: 17 July 2020

Abstract: As modern embedded systems are becoming more and more ubiquitous and interconnected, they attract a world-wide attention of attackers and the security aspect is more important than ever during the design of those systems. Moreover, given the ever-increasing complexity of the applications that run on these systems, it becomes increasingly difficult to meet all security criteria. While extra-functional design objectives such as performance and power/energy consumption are typically taken into account already during the very early stages of embedded systems design, system security is still mostly considered as an afterthought. That is, security is usually not regarded in the process of (early) design-space exploration of embedded systems, which is the critical process of multi-objective optimization that aims at optimizing the extra-functional behavior of a design. This position paper argues for the development of techniques for quantifying the 'degree of secureness' of embedded system design instances such that these can be incorporated in a multi-objective optimization process. Such technology would allow for the optimization of security aspects of embedded systems during the earliest design phases as well as for studying the trade-offs between security and the other design objectives such as performance, power consumption and cost.

Keywords: embedded computer systems; cyber security; system-level design and design-space exploration; multi-objective optimization; system trade-offs

1. Introduction

Embedded computer systems are ubiquitous and have a major impact on our society. Examples of such systems are close at hand—modern TVs contain one or multiple computer systems to handle functionality such as decoding the input signal, performing various image enhancement techniques as well as displaying and updating live information (e.g., program guide or weather forecast). Smart-phones rely on embedded computer systems to allow users to make phone calls, shoot photos and videos, perform GPS navigation, browse the Internet, execute apps, and so on. The use of embedded computer systems is, however, by no means restricted to consumer electronics: in industrial, medical, automotive, avionic, or defense applications they are equally pervasive.

The complexity of the underlying system architectures of modern embedded systems forces designers to start with modeling and simulating (possible) system components and their interactions in the very early design stages. This is often referred to as system-level design [1]. The system-level models typically represent application workload behavior, characteristics of the underlying computing platform architecture, and the relation (e.g., mapping, hardware-software partitioning) between application workload(s) and platform architecture. These models are applied at a high level of abstraction, thereby minimizing the modeling effort and optimizing the simulation speed. This is especially needed for targeting the early design stages since many design decisions are still open and, therefore, many design alternatives still need to be studied. High-level system modeling allows for the early verification of a design and can provide estimates on the extra-functional properties of a design such as system performance and energy/power consumption. The system-level models are typically accompanied by a methodology for efficient design-space exploration (DSE) [2], which is the process of assessing alternative design instances with respect to (i) the platform architecture that will be deployed (e.g., the number and type of processing elements in the platform, the type of network to interconnect the processors, etc.) and(ii) the mapping of application tasks to the underlying platform components [3]. It is a multi-objective optimization problem that searches through the space of different implementation alternatives to find optimal design instances. Exploration of different design choices, especially during the early design stages where the design space is still at its largest, is of eminent importance. Wrong decisions early in the design can be extremely costly in terms of re-design effort, or even deadly to the product's success. Consequently, considerable research effort in the embedded systems domain has been spent in the last two decades on developing frameworks for system-level modeling and simulation that aim for early design-space exploration.

As embedded systems are becoming more and more ubiquitous and interconnected (illustrated by, e.g., the strong trend towards the Internet of Things), they also attract a world-wide attention of attackers. This makes the security aspect more important than ever during the design of these systems [4]. Moreover, given the ever-increasing complexity of the applications that run on modern embedded systems, it becomes increasingly difficult to meet all security criteria. While design objectives such as performance and power/energy consumption are usually taken into account during the early stages of design (as explained above), system security is still mostly considered as an afterthought. That is, security is typically not regarded in the process of (early) design-space exploration of embedded systems. However, any security measures that may eventually be taken much later in the design process do affect the already established trade-offs with respect to the other extra-functional properties of the system like performance, power/energy consumption, cost, and so forth [4]. Thus, covering the security aspect in the earliest phases of design is necessary to design systems that are, in the end, optimal with regard to all extra-functional objectives. However, this poses great difficulties because unlike the earlier mentioned conventional system objectives, like performance and power consumption, security is hard to quantify—there exists no single metric with which one can measure the degree of secureness of a design.

This position paper argues for the need for security-aware, system-level design-space exploration methods and techniques for embedded systems. To this end, we will discuss a multifaceted, scoring-based methodology for quantifying the degree of secureness of embedded system design instances. This methodology allows for incorporating the secureness quantifications in a multi-objective optimization process and would thus enable optimization of the security aspect during the earliest phases of design. However, we want to emphasize the fact that this is a position paper and therefore does not present an actual implementation of the proposed solution nor any experimental results.

The remainder of this paper is organized as follows. In the next section, we will provide a brief introduction to the concept of design-space exploration. In Section 3, we will describe our proposal for a security-aware DSE approach, focusing on a method to quantify the secureness of embedded system design instances. Section 4 discusses related work, after which Section 5 concludes the paper.

2. Design-Space Exploration

During the design-space exploration (DSE) of embedded systems, multiple optimization *objectives*—such as performance, power/energy consumption, and cost—should be considered simultaneously. This is called multi-objective DSE [2]. Since the objectives are often in conflict, there cannot be a single optimal solution that simultaneously optimizes all objectives. Therefore, optimal decisions need to be taken in the presence of trade-offs between design criteria.

2.1. Multi-Objective Optimization

Given a set of m decision variables, which are the degrees of freedom (e.g., parameters like the number and type of processors in the system, application mapping, etc.) that are explored during DSE, a so-called *fitness function* must optimize the n objective values [2]. The fitness function is defined as:

$$f_i : R^m \rightarrow R^1. \tag{1}$$

A potential solution $x \in R^m$ is an assignment of the m decision variables. The fitness function f_i translates a point in the solution space X into the i-th objective value (where $1 \leq i \leq n$). For example, a particular fitness function f_i could assess the performance or energy efficiency of a certain solution x (representing a specific design instance). The combined fitness function $f(x)$ subsequently translates a point in the solution space into the objective space Y. Formally, a multi-objective optimization problem (MOP) that tries to identify a solution x for the m decision variables that minimizes the n objective values using objective functions f_i with $1 \leq i \leq n$:

$$\begin{aligned} \text{Minimize } y &= f(x) = (f_1(x), f_2(x), ..., f_n(x)) \\ \text{Where } x &= (x_1, x_2, ..., x_m) \in \ X \\ y &= (y_1, y_2, ..., y_n) \in Y. \end{aligned}$$

Here, the decision variables x_i (with $1 \leq i \leq m$) usually are constrained. These constraints make sure that the decision variables refer to valid system configurations (e.g., using not more than the available number of processors, using a valid mapping of application tasks to processing resources, etc.), that is, x_i are part of the so-called feasible set. In the remainder of this section, we assume a minimization procedure, but without loss of generality, this minimization procedure can be converted into a maximization problem by multiplying the fitness values y_i with -1.

With an optimization of a single objective, the comparison of solutions is trivial. A better fitness (i.e., objective value) means a better solution. With multiple objectives, however, the comparison becomes non-trivial. Take, for example, two different embedded system architecture designs: a high-performance system and a slower but much cheaper system. In case there is no preference defined with respect to the objectives and there are also no restrictions for the objectives, one cannot say if the high-performance system is better or the low-cost system. A typical MOP in the context of embedded systems design can have a variety of different objectives, like performance, energy consumption, cost and reliability. To compare different solutions in the case of multiple objectives, the Pareto dominance relation is generally used. Here, a solution $x_a \in X$ is said to dominate solution $x_b \in X$ if and only if $x_a < x_b$:

$$\begin{aligned} x_a < x_b \iff & \forall i \in \{1, 2, ..., n\} : f_i(x_a) \leq f_i(x_b) \wedge \\ & \exists i \in \{1, 2, ..., n\} : f_i(x_a) < f_i(x_b). \end{aligned}$$

Hence, a solution x_a dominates x_b if its objective values are superior to the objective values of x_b. For all of the objectives, x_a must not have a worse objective value than solution x_b. Additionally, there must be at least one objective in which solution x_a is better (otherwise they are equal).

An example of the dominance relation is given in Figure 1, which illustrates a two dimensional MOP. For solution H the dominance relations are shown. Solution H is dominated by solutions B, C and D as all of them have a lower value for both f_1 and f_2. On the other hand, solution H is superior to solutions M, N and O. Finally, some of the solutions are not comparable to H. These solutions are better for one objective but worse for another.

The Pareto dominance relation only provides a partial ordering. For example, the solutions A to F of the example in Figure 1 cannot be ordered using the ordering relation. Since not all solutions $x \in X$

can be ordered, the result of a MOP is not a single solution, but a front of non-dominated solutions, called the *Pareto front*. A set X' is defined to be a Pareto front of the set of solutions X as follows:

$$\{x \in X' \mid \nexists x_a \in X \colon x_a < x\}.$$

The Pareto front of Figure 1 contains six solutions: $A - F$. Each of these solutions does not dominate the other. An improvement on objective f_1 is matched by a worse value for f_2. Generally, it is up to the designer to decide which of the solutions provides the best trade-off.

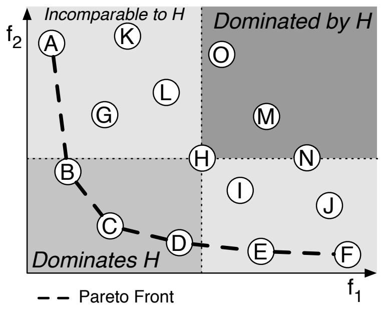

Figure 1. A Pareto front and an example of the dominance relation (taken from Reference [2]).

2.2. Search for Pareto Optimal Solutions

The search for Pareto optimal design points with respect to multiple design criteria entails two distinct elements [5]:

1. The evaluation of a single design point using the fitness function(s) $f(x)$ regarding all the objectives in question like system performance, power/energy consumption and so on. These evaluations are usually based on measurements using real systems or predictions from either analytical models or simulation models [2].
2. The search strategy for navigating through and covering the design space during the DSE process. Such search strategies can be based on exact, but typically unscalable, methods that guarantee finding the optimal solution(s). These exact methods can, for example, be implemented using integer linear programming (ILP) solutions (e.g., References [6,7]) or branch & bound algorithms (e.g., Reference [8]). Alternatively, so-called meta-heurisics, such as genetic algorithms (GA) or simulated annealing, can be used to search the design space for optimal solutions. They only perform a finite number of design point evaluations, and can thus handle larger design spaces. However, there is no guarantee that the global optimum will be found using meta-heuristics, and therefore the result can be a local optimum within the design space. GA-based DSE has been widely studied in the domain of system-level embedded design (e.g., References [9–12]) and has demonstrated to yield good results.

In this paper, we focus on the fitness evaluation aspect of DSE. More specifically, we argue that while there are well-established techniques and metrics for the fitness evaluation of traditional design objectives such as performance, power/energy consumption, cost, and reliability, this is not the case for evaluating the fitness of design instances in terms of how secure they are. This lack of security fitness evaluation methods and metrics inhibits the use of system security as a first-class citizen in the process of early design-space exploration of embedded systems. As was indicated before, such design practice leads to suboptimal products because any security measures that may be taken later in the design process do affect the already established trade-offs with respect to the other extra-functional properties of the system like performance, power/energy consumption, cost, and so forth.

In the next section, we will therefore argue for the development of a security-aware DSE approach, based on a multifaceted, scoring-based security quantification methodology. This methodology allows for quantifying the degree of secureness of design instances such that these can be incorporated in the DSE's multi-objective optimization process. Eventually, once such a security-aware DSE would have been implemented, it would allow for optimization of security aspects of embedded systems in their earliest design phases as well as for studying the trade-offs between security and the other design objectives like performance, power consumption and cost. Evidently, such technology would provide a substantial competitive advantage in the embedded systems industry.

3. Towards Security-Aware, System-Level DSE

The envisioned approach for security-aware system-level design-space exploration, adopting a multifaceted, scoring-based security quantification methodology, is illustrated in Figure 2. Below, we will explain the different components of this approach. The blue parts of Figure 2 refer to the methodology components that only need to be specified or performed once, whereas the red parts refer to components that are dependent on the design-space exploration process and thus must be revisited every time a new design instance is evaluated in terms of extra-functional properties such as performance, power consumption, and of course, in the scope of this paper, also secureness. Before describing our approach in detail, however, we will first discuss several assumptions that delimit our proposed approach.

3.1. Assumptions

We focus on security threats in which the underlying embedded system architecture plays a central role, and do not consider any security flaws that can be exploited purely at the application level. This implies that we restrict ourselves to the following set of attack types:

1. *Side-channel attacks* like power analysis attacks, timing attacks such as the recent Spectre and Meltdown attacks, scan attacks, differential fault analysis attacks and electromagnetic analysis attacks (see References [13,14] for an overview of these side-channel attacks);
2. *Denial of service attacks* [15,16];
3. *Software-based attacks* such as buffer overflows for which protection mechanisms may be available at the system (architecture) level (e.g., Reference [13]);
4. *Attacks directed towards breaking encryption algorithms* [17].

For each of the above attacks, we subsequently consider a range of protection mechanisms—derived from literature—that can be applied to protect specific system components or the entire system against these attacks.

Moreover, we consider system-level DSE in which both the platform architecture (e.g., selection of platform components such processing elements, memories, and networking components) as well as the mapping of application tasks and communications to the selected platform components are optimized for traditional objectives such as performance, power consumption, and cost, but now also for secureness. Such system-level design-space exploration is depicted in the top-middle part of Figure 2 and could, for example, be performed with system-level DSE frameworks such as Sesame [18,19] or a similar environment (e.g, References [20–22]). Important to note here is that the performance, power and cost models used in the DSE also need to account for the effects of deploying specifically selected security protection mechanisms (as discussed above) inside a platform architecture.

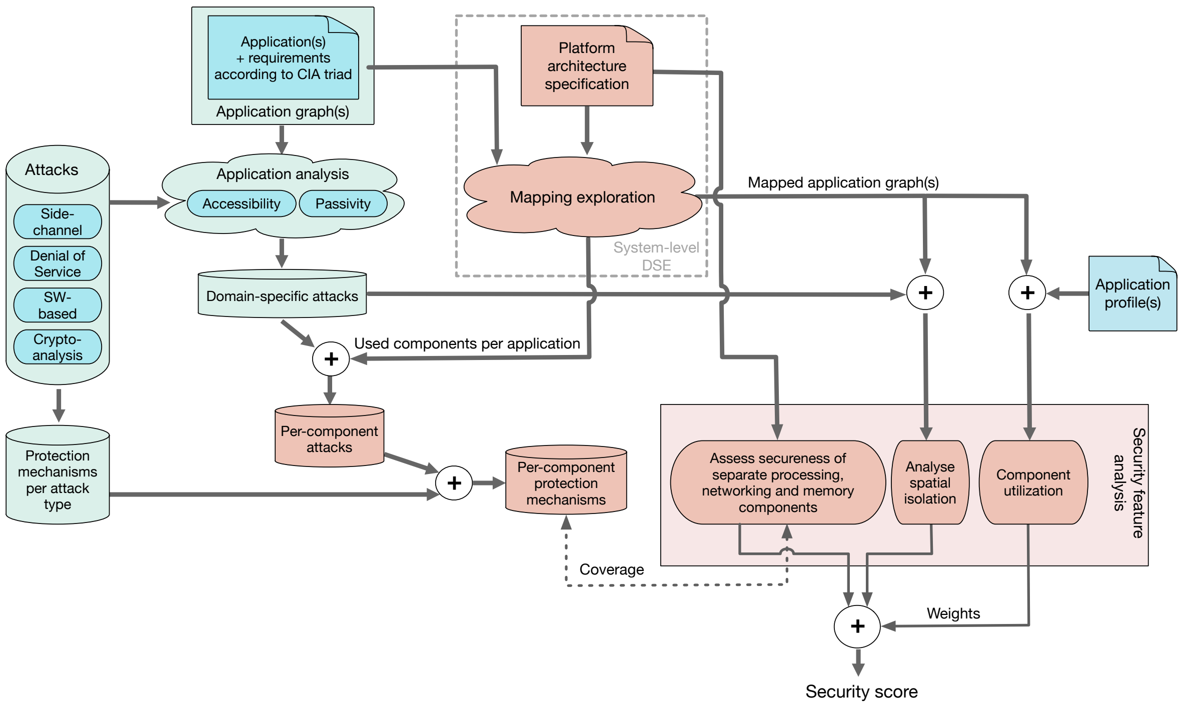
Application(s) + requirements according to CIA triad
Application graph(s)
Platform architecture specification
Mapping exploration
System-level DSE
Attacks
Side-channel
Denial of Service
SW-based
Crypto-analysis
Application analysis
Accessibility
Passivity
Domain-specific attacks
Mapped application graph(s)
Application profile(s)
Used components per application
Per-component attacks
Protection mechanisms per attack type
Per-component protection mechanisms
Assess secureness of separate processing, networking and memory components
Analyse spatial isolation
Component utilization
Security feature analysis
Coverage
Weights
Security score

Figure 2. Proposed approach for security-aware system-level design-space exploration (DSE) using a multifaceted, scoring-based security quantification methodology.

3.2. A Multifaceted, Scoring-Based Methodology for Secureness Quantification

As a first step in our methodology, shown in the top left of Figure 2, the applications that need to execute on the target embedded system together with their extra-functional requirements are identified and specified. The specified extra-functional requirements include the traditional ones such as performance and real-time requirements, power/energy consumption budgets, and so forth, but also requirements in terms of secureness. Regarding the latter, one can use the well-known CIA triad to indicate the needs with respect to the security aspects **C**onfidentiality (preventing sensitive information from reaching the wrong people), **I**ntegrity (maintaining consistency, accuracy, and trustworthiness of data) and **A**vailability (ensuring timely and reliable access to, and use of, information) [23]. Here, a domain-specific language (DSL) could be developed to specify these extra-functional requirements. The application workloads themselves can be specified using task or process graphs, explicitly describing application tasks and their interactions (communications).

Given the attack types we consider in our proposed approach, as discussed in Section 3.1 and shown at the left in Figure 2, only those attacks that are relevant for the embedded system under design need to be identified, which we refer to as the so-called domain-specific attacks. To this end, we need to consider the security requirements of the target embedded system as specified using the CIA triad as well as the characteristics of the specific attack types in terms of, for example, passivity and accessibility. Here, passivity refers to what extent an attack manipulates the target system, either as a means or a goal of the attack. For example, a denial of service attack clearly is an active attack as its sole aim is to manipulate the system, whereas a side-channel attack based on power analysis is a passive attack. Accessibility refers to the access level that is required for an attack to be performed. Revisiting the example of a side-channel attack via power analysis, such an attack obviously requires physical access to the embedded system, whereas for example, a software-based attack does not require this. If we now consider, for example, an anti-lock braking system, then confidentiality is not a major concern as such a system does not process sensitive information. This makes passive attacks such as side-channel attacks not relevant and can therefore be excluded from the set of domain-specific attacks. However, the braking system may not be disrupted or manipulated (i.e., integrity and availability are crucial CIA elements) thereby making active attacks highly relevant. Table 1 provides an overview of the required access level and passivity of the attack types we consider in this paper. Here, virtual access level attacks require access to one process that runs within an application on the embedded system in question. Cryptanalysis attacks often do not require access to the system. For example, in the case of public key cryptography, the public key is distributed to other systems and therefore freely available.

Table 1. Required access level and passivity of various types of attacks on embedded systems.

Attack	Sub-Type	Access Level	Passivity
Side-channel	Power analysis	Physical	Passive
	Timing attack	Virtual	Passive
	Scan attack	Physical	Active
	Fault Analysis	Physical & Virtual	Passive & Active
	Electromagnetic Analysis	Physical	Passive
Denial of Service		Virtual	Active
Software	Buffer overflow	Virtual	Active
Cryptanalysis		None	Passive

Once the set of domain-specific attacks has been determined, those attacks that are relevant to the different components in the underlying platform architecture can be determined: for example, a networking component is not susceptible to a software-based buffer overflow attack, whereas a microprocessor component is. To do so, we also need the mapping information (i.e., which application tasks and communications are mapped onto what platform components) of the design instance(s) that are currently being explored by the system-level DSE process. Subsequently, for each component in the platform architecture, we can now determine the set of possible security protection mechanisms that can be deployed to effectively increase its secureness ('Per-component protection mechanisms' in Figure 2). These sets of possible per-component protection mechanisms are an important ingredient of our envisioned scoring-based security quantification methodology: they allow for determining the coverage with respect to the protection mechanisms that are actually deployed in the design instances being explored by the system-level DSE. To achieve this, a scoring technique would be needed that can capture binary coverage relationships (i.e., a certain protection mechanism is available or not) as well as numerical coverage relationships. The latter applies in cases where, for example, a certain amount of random noise is added to a system component to disguise real power behavior in order to complicate or even prevent side-channel attacks based on power analysis [24] (here, the amount of noise forms a power/security trade-off) or when the strength of a cryptographic processor is identified as a function of the key-size it uses.

Besides the coverage of protection mechanisms deployed in the platform architecture components, one can take two other facets into account to determine the security score of a particular design instance. First, the spatial isolation realized in design instances can be considered. That is, reducing the amount of resource sharing between applications or even between tasks from a single application will increase the secureness of the system, as this will complicate certain types of attacks such as side-channel attacks. Therefore, a proper technique for quantifying the spatial isolation (using the mapping information from the system-level DSE) would be required such that it can be used for security scoring purposes. As a final ingredient of our anticipated security score, the platform component utilization can be used. The rationale behind this is that higher utilized components typically are more prone to certain attacks. Moreover, higher utilized components possibly also play a more important role in achieving the CIA-triad system requirements. To include the platform component utilization, we need to profile the application(s) to measure the activity of application tasks and communications, that is, the degree to which they utilize the underlying resources. Hereafter, this information is related to the mapping of these application tasks and communications onto the platform architecture. The resulting component utilization can then be used to weight the protection mechanism coverage and spatial isolation of design instances in the final security scoring (as shown at the bottom of Figure 2).

3.3. Scoring the Security of Design Instances

Above, we described the ingredients of our envisioned security scoring methodology. We do realize that we have not provided any details on how such security scoring could actually be implemented. Actually, this remains a topic for future research, which will hopefully also be picked up by the community. Nevertheless, in this section, we would like to provide a rough sketch of a fairly simple approach to do such scoring.

Given a mapping of a (set of) application(s) to the underlying resources of a possible platform architecture, which includes the mapping of application tasks to computational resources as well as the mapping of inter-task communications to network and memory resources. Then, for each utilized component in the platform, we could calculate a security score along the following lines. Let AT_x be the set of Attack Types (see e.g., the second column of Table 1) to which component x is susceptible:

$$AT_x = \{ \bigcup_{\forall t_i | c_i \text{ mapped on x}} \text{Attacks}_{t_i|c_i} \}.$$

For AT_x, we only consider the application tasks t_i or inter-task communications c_i (dependent on whether x is a processing or communication component) that are mapped to component x. $\text{Attacks}_{t_i|c_i}$ refers to the possible attacks for task t_i or communication c_i, taking into account its security requirements according to the CIA triad as well as taking into account the access level and passivity of the various attacks (see Table 1). To determine a security score S_x for a component x, one could then perform the following calculation:

$$S_x = \frac{\sum_{p \in P_{AT_x}} \text{Protection-level}(p, x)}{|AT_x|} \cdot \frac{1}{\text{Utilization}_x}.$$

Here, p is a particular protection mechanism and is part of the set P_{AT_x} that consists of the possible protection mechanisms for the attacks listed in set AT_x for component x. The function Protection-level(p, x) returns a value that indicates the extent to which component x implements protection mechanism p. This function could, for example, return a value between 0 and 1: The value 0 would mean that the component does not implement the protection mechanism, implying that component x would be fully susceptible to the associated attack type. The value 1, on the other hand, would refer to an available implementation of protection mechanism p such that component x is fully protected against attacks of the associated type. Evidently, the returned value may also be in between 0 and 1, indicating partial protection. For example, in the case protection mechanism p adds a certain amount of random noise to a component to disguise real power behavior in order to prevent side-channel attacks based on power analysis [24], the level of added noise (which is a trade-off between power consumption and security) would determine the returned value of the function Protection-level(p, x). To calculate S_x, we also consider the reciprocal of the utilization of component x. Here, the utilization refers to the fraction the computing/communication component x contributes to the overall computing or communication time of an application. The rationale behind this is that one could argue that those components (both processing and communication components) that are less active over time will be less susceptible to certain types of attacks, like side-channel attacks. To determine the overall security score of a specific design instance (i.e., a particular application mapping to a selected platform architecture), one could simply accumulate the S_x scores for all components x used in the design instance.

The above scoring example is by no means meant to be a complete and fully-fledged solution to the security scoring problem. It is merely meant to act as an illustration for the direction of thought as presented in this paper. Moreover, in the above scoring example, we also do not consider spatial isolation as part of the security score. One direction to accomplish this, would for example, to penalize the mapping of multiple application tasks or communications to a single platform component.

The multifaceted, scoring-based security quantification methodology as outlined in this section could provide a real innovation to system-level embedded system design as it would facilitate designers to study the trade-offs between the performance, power consumption, cost, and secureness of design instances during the early stages of design.

4. Related Work

The need for recognizing security as a first-class citizen, next to traditional design objectives such as performance, cost and power consumption, in the design of embedded systems is not new. For example, quoting from Reference [4], "However, security is often misconstrued by embedded system designers as the addition of features, such as specific cryptographic algorithms and security protocols, to the system. In reality, it is a new dimension that designers should consider throughout the design process, along with other metrics such as cost, performance, and power." Nevertheless, the integration of security aspects in the process of system-level design-space exploration of embedded systems has never really got off the ground and is still a largely uncharted research ground. Only a few efforts exist that address this problem but, at best, most of them provide partial solutions or solutions to very specific security problems. For example, in Reference [25], the evaluation of security protocols is integrated in the design process. For instance, it rates the security of a system based on the probability of a hash collision. However, it does not cover other types of attacks, such as timing attacks and power analysis. The authors of Reference [26] try to neutralize several types of side-channel attacks by means of spatial isolation in a DSE setting but, again, they do not consider any other types of attacks/protection mechanisms. In Reference [27], a small number of attacker capabilities and corresponding requirements that refer to the CIA triad are defined in the context of DSE. The problem of this approach is that it is not trivial to relate types of attacks to those capabilities and requirements. The work of Reference [28] incorporates security in system-level DSE by first generating potential architecture configurations, after which an automated security analysis is performed to check the generated configurations against designer-specified security constraints.

In all the above works of References [26–28] security is modelled as a requirement in the DSE process, which does not allow for studying actual trade-offs between performance, power consumption or cost in relationship to secureness of a design.

An alternative approach for quantifying security is by means of a security risk assessment using a specific attack model [29]. For example, Reference [30], proposes an attack tree model to evaluate the user's privacy risks associated with an Internet-of-Things eco system. They evaluate the potential risks based on varying attack attributes, the probable considerations or preferences of an adversary, and the varying computational resources available on a device. Research efforts like this are, however, typically not focused on the process of (early) DSE.

To the best of our knowledge, only the works of References [31,32] and Reference [33] are similar to what we propose in terms of aiming at incorporating security as an objective that can be traded off with other objectives in the process of early DSE. In References [31,32], the authors introduce an UML-based approach in which application security requirements can be described together with security 'capabilities'—in addition to other extra-functional aspects such as performance and power consumption—of system components stored in a library. This then allows for a DSE process during which the application requirements are matched with the component capabilities. The very recent work of Reference [33] introduces a novel DSE framework that allows for considering 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 tries to find the optimal design for an embedded system.

5. Conclusions

As embedded systems are becoming increasingly ubiquitous and interconnected, they attract a world-wide attention of attackers. This makes the security aspect during the design of these systems more important than ever. However, state-of-the-art design tools and methodologies for embedded systems do not consider system security as a primary design objective. This is especially true for the early design phases in which the process of design-space exploration is of eminent importance for performing trade-off analysis. Any security measures that may eventually be taken much later in the design process will then affect the already established design trade-offs with respect to the other, and more traditional, design objectives like system performance, power consumption and cost. It goes without saying that such a design practice leads to suboptimal products.

In this position paper, we therefore argued for security-aware design methods for embedded systems that will allow for the optimization of security aspects of embedded systems in their earliest design phases as well as for studying the trade-offs between security and the other design objectives such as performance, power consumption and cost. To this end, we proposed a multifaceted, scoring-based methodology for quantifying the degree of secureness of embedded system design instances, which would allow for incorporating these secureness quantifications in early design-space exploration of embedded systems. The proposed methodology has not yet been implemented, and would require further research to do so. We do hope, however, that this position paper will be a trigger for more wide-spread research on techniques that allow for incorporating security as a first-class citizen in the process of early design-space exploration of embedded systems.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Keutzer, K.; Newton, A.; Rabaey, J.; Sangiovanni-Vincentelli, A. System-level design: Orthogonalization of concerns and platform-based design. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2000**, *19*, 1523–1543. [CrossRef]
2. Pimentel, A.D. Exploring Exploration: A Tutorial Introduction to Embedded Systems Design Space Exploration. *IEEE Des. Test* **2017**, *34*, 77–90. [CrossRef]
3. Singh, A.K.; Shafique, M.; Kumar, A.; Henkel, J. Mapping on multi/many-core systems: survey of current and emerging trends. In Proceedings of the 2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC), Austin, TX, USA, 29 May–7 June 2013; pp. 1–10.
4. Ravi, S.; Raghunathan, A.; Kocher, P.; Hattangady, S. Security in embedded systems: Design challenges. *ACM Trans. Embed. Comput. Syst. (TECS)* **2004**, *4*, 461–491. [CrossRef]
5. Gries, M. Methods for evaluating and covering the design space during early design development. *Integration* **2004**, *38*, 131–183. [CrossRef]
6. Niemann, R.; Marwedel, P. An Algorithm for Hardware/Software Partitioning Using Mixed Integer Linear Programming. *Des. Autom. Embed. Syst.* **1997**, *2*, 165–193. [CrossRef]
7. Lukasiewycz, M.; Glass, M.; Haubelt, C.; Teich, J. Efficient symbolic multi-objective design space exploration. In Proceedings of the 2008 Asia and South Pacific Design Automation Conference, Seoul, Korea, 21–24 March 2008; pp. 691–696.
8. Padmanabhan, S.; Chen, Y.; Chamberlain, R.D. Optimal design-space exploration of streaming applications. In Proceedings of the ASAP 2011-22nd IEEE International Conference on Application-specific Systems, Architectures and Processors, Santa Monica, CA, USA, 11–14 September 2011; pp. 227–230.
9. Palesi, M.; Givargis, T. Multi-objective Design Space Exploration Using Genetic Algorithms. In Proceedings of the CODES02: 10th International Symposium on Hardware/Software Codesign, Estes Park, CO, USA, 6–8 May 2002; pp. 67–72.

10. Erbas, C.; Cerav-Erbas, S.; Pimentel, A.D. Multiobjective Optimization and Evolutionary Algorithms for the Application Mapping Problem in Multiprocessor System-on-Chip Design. *IEEE Trans. Evol. Comput.* **2006**, *10*, 358–374. [CrossRef]
11. Thompson, M.; Pimentel, A.D. Exploiting Domain Knowledge in System-level MPSoC Design Space Exploration. *J. Syst. Archit.* **2013**, *59*, 351–360. [CrossRef]
12. Quan, W.; Pimentel, A.D. Towards Exploring Vast MPSoC Mapping Design Spaces using a Bias-Elitist Evolutionary Approach. In Proceedings of the 2014 17th Euromicro Conference on Digital System Design, Verona, Italy, 27–29 August 2014; pp. 655–658.
13. Ravi, S.; Raghunathan, A.; Chakradhar, S. Tamper resistance mechanisms for secure embedded systems. In Proceedings of the 17th International Conference on VLSI Design. Proceedings, Mumbai, India, 9 January 2004; pp. 605–611. [CrossRef]
14. Yang, B.; Wu, K.; Karri, R. Scan based side channel attack on dedicated hardware implementations of Data Encryption Standard. In Proceedings of the 2004 International Conferece on Test, Charlotte, NC, USA, 26–28 October 2004; pp. 339–344. [CrossRef]
15. Papp, D.; Ma, Z.; Buttyan, L. Embedded systems security: Threats, vulnerabilities, and attack taxonomy. In Proceedings of the 2015 13th Annual Conference on Privacy, Security and Trust (PST), Izmir, Turkey, 21–23 July 2015; pp. 145–152. [CrossRef]
16. Koopman, P. Embedded systems security. *IEEE Comput.* **2004**, *37*, 145–152. [CrossRef]
17. Swenson, C. *Modern Cryptanalysis: Techniques for Advanced Code Breaking*; Wiley Publishing: Hoboken, NJ, USA, 2008.
18. Pimentel, A.D.; Erbas, C.; Polstra, S. A Systematic Approach to Exploring Embedded System Architectures at Multiple Abstraction Levels. *IEEE Trans. Comput.* **2006**, *55*, 99–112. [CrossRef]
19. Erbas, C.; Pimentel, A.D.; Thompson, M.; Polstra, S. A Framework for System-level Modeling and Simulation of Embedded Systems Architectures. *EURASIP J. Embed. Syst.* **2007**, *2007*, 082123. [CrossRef]
20. Mohanty, S.; Prasanna, V.K.; Neema, S.; Davis, J. Rapid Design Space Exploration of Heterogeneous Embedded Systems Using Symbolic Search and Multi-granular Simulation. *ACM SIGPLAN Not.* **2002**, *37*, 18–27. [CrossRef]
21. Mariani, G.; Brankovic, A.; Palermo, G.; Jovic, J.; Zaccaria, V.; Silvano, C. A correlation-based design space exploration methodology for multi-processor systems-on-chip. In Proceedings of the Proc. of the Design Automation Conference (DAC), Anaheim, CA, USA, 13–18 June 2010; pp. 120–125.
22. Jia, Z.J.; Bautista, T.; Núñez, A.; Thompson, M.; Pimentel, A.D. A System-level Infrastructure for Multi-dimensional MP-SoC Design Space Co-exploration. *ACM Trans. Embed. Comput. Syst.* **2013**, *13*, 1–26. [CrossRef]
23. Whitman, M.E.; Mattord, H.J. *Principles of Information Security*, 4th ed.; Course Technology Press: Boston, MA, USA, 2011.
24. Benini, L.; Macii, A.; Macii, E.; Omerbegovic, E.; Pro, F.; Poncino, M. Energy-Aware Design Techniques for Differential Power Analysis Protection. In Proceedings of the 40th Annual Design Automation Conference (DAC), Anaheim, CA, USA, 2–6 June 2003; pp. 36–41.
25. Lin, C.W.; Zheng, B.; Zhu, Q.; Sangiovanni-Vincentelli, A. Security-Aware Design Methodology and Optimization for Automotive Systems. *ACM Trans. Des. Autom. Electron. Syst.* **2015**, *21*, 1–26. [CrossRef]
26. Weichslgartner, A.; Wildermann, S.; Götzfried, J.; Freiling, F.; Glaundefined, M.; Teich, J. Design-Time/Run-Time Mapping of Security-Critical Applications in Heterogeneous MPSoCs. In Proceedings of the 19th International Workshop on Software and Compilers for Embedded Systems (SCOPES), Sankt Goar, Germany, 23–25 May 2016; pp. 153–162.
27. Stierand, I.; Malipatlolla, S.; Fröschle, S.; Stühring, A.; Henkler, S. Integrating the Security Aspect into Design Space Exploration of Embedded Systems. In Proceedings of the IEEE International Symposium on Software Reliability Engineering Workshops, Naples, Italy, 3–6 November 2014; pp. 371–376.
28. Tan, B.; Biglari-Abhari, M.; Salcic, Z. An Automated Security-Aware Approach for Design of Embedded Systems on MPSoC. *ACM Trans. Embed. Comput. Syst.* **2017**, *16*, 1–20. [CrossRef]
29. Rocchetto, M.; Ferrari, A.; Senni, V. Challenges and Opportunities for Model-Based Security Risk Assessment of Cyber-Physical Systems. In *Resilience of Cyber-Physical Systems. Advanced Sciences and Technologies for Security Applications*; Flammini, F., Ed.; Springer: Berlin, Germany, 2019.

30. Asif, W.; Ray, I.G.; Rajarajan, M. An Attack Tree Based Risk Evaluation Approach for the Internet of Things. In Proceedings of the 8th International Conference on the Internet of Things, Santa Barbara, CA, USA, 15–18 October 2018.
31. Ferrante, A.; Milosevic, J.; Janjšević, M. A security-enhanced design methodology for embedded systems. In Proceedings of the 2013 International Conference on Security and Cryptography (SECRYPT), Reykjavik, Iceland, 29–31 July 2013; pp. 1–12.
32. Ferrante, A.; Kaitovic, I.; Milosevic, J. Modelling Requirements for Security-Enhanced Design of Embedded Systems. In Proceedings of the International Conference on Security and Cryptography (SECRYPT), Vienna, Austria, 28–30 August 2014; pp. 315–320.
33. Gressl, L.; Steger, C.; Neffe, U. Security Driven Design Space Exploration for Embedded Systems. In Proceedings of the 2019 Forum for Specification and Design Languages (FDL), Southampton, UK, 2–4 September 2019; pp. 1–8.