A Security-enhanced Design Methodology for Embedded Systems

Alberto Ferrante, Jelena Milosevic and Marija Janjušević
ALaRI, Faculty of Informatics, University of Lugano, via G. Buffi 13, Lugano, Switzerland

Keywords: Security, Design Methodology, Embedded Systems, Security Metric, Metric, AHP.

Abstract: Designing an embedded system is a complex process that involves working both on hardware and on software. Designers often optimize the systems that they design for specific applications; an optimal system is the one that can execute the desired set of applications with the required performances at the lowest possible cost. Cost may be expressed in different ways such as, for example, energy consumption and/or silicon area. Security is being, in the common practice, disregarded during this phase and inserted in later stages of the design process, thus obtaining non optimal and/or non safe systems.
In this paper we propose a design methodology for embedded systems that integrate the choice of suitable design solutions into the early stages of the design process. The main purpose of this methodology is to provide a way to evaluate security as an additional optimization parameter. Along with a description of the methodology, in this paper we also show a case study that explains how the methodology can be applied and that proves its effectiveness.

1 INTRODUCTION

The use of embedded systems has been continuously increasing in the last years both with respect to the number of application areas and to the complexity of platforms designed: embedded systems have become fundamental in ICT. Unfortunately, this increase in their use, even in new, critical, and non-protected environments, has not been accompanied by a proper development of methodologies for making them secure (Viega and Thompson, 2012). Yet, security for these systems proved to be a more difficult and more critical problem than security for general purpose systems: added difficulties are given by the often limited amount of resources available to implement security solutions. Criticality is given both by the kinds of data stored in mobile devices (e.g., phone bookmarks and personal user data) and by the kind of damages that an attack to a mobile system may produce. Though, security is not yet perceived as a significant problem by embedded systems users. Embedded systems (e.g., vehicle infotainment systems, smart house control system, a health monitoring system) may be directly or indirectly (i.e., through other devices) network connected for different purposes and they can be used as a mean for attacking these devices (Seo and Cho, 2012). Attackers may exploit possible security holes to gain access to the control system considered and cause significant damages. In fact, while a proper backup mechanism can, for example, allow the recovery of lost data, a hacked insulin pump may create damages to the wearer that cannot be reverted (Viega and Thompson, 2012). Attacks can be conducted through hardware or software. Security needs to be approached in a new way to cope with the new threats posed by the new capabilities of mobile systems and with the multiple environments in which they are used.

To solve the aforementioned problems not only innovative security techniques for embedded systems need to be developed, but also a comprehensive, yet flexible, approach is required. In this approach security should be targeted at all levels of the embedded system, ranging from hardware to application software. At the software level the goal is not only to rely on code to protect the system but to closely interact with hardware resources to provide more efficient protection techniques matching with embedded systems constraints (memory footprint, power limitation). Considering such an approach will become mandatory to increase the level of security and to better take advantage of the power computation available in the device. At the hardware level designing efficient solutions to protect and monitor the execution of the application is essential to meet application constraints in terms of speed. Bridging the gap between application and hardware resources in the domain of security will provide the guarantee to develop a defense in depth approach. The goal of this approach is not only to provide protection against known attacks, but also to be more resistant to not-known ones. Such a holistic approach will also offer designers a clearer vision about security and it will allow them to evaluate and customize security solutions in their design during the design space exploration (i.e., the evaluation and comparison of different design points to find the optimal one with respect to different metrics) phase, based on threat models foreseen for the device.

Although important, the development of comprehensive security solutions does not solve the problem of designing secure embedded systems. In fact, embedded system designers have often seen security as a desired but not directly feasible and not critical feature. This is due to different reasons, the main one being that the organizations developing embedded software often lack support of security specialists (Koopman, 2004); the introduction of security mechanisms in embedded systems is further complicated by the non existence of a design methodology aimed at including security in system codesign from the early phases of development. Designers miss both security solutions specifically optimized for embedded systems and the information that would allow them to easily integrate those solutions into their designs by evaluating security-cost trade-offs. Presently, most embedded systems are developed by using performance and/or power driven approaches. By these approaches, different design solutions (i.e., different system hardware/software configurations) are evaluated in a procedure called design space exploration (Palermo et al., 2008; Alippi et al., 2004).

In this paper we discuss a design methodology that allows designers to easily include security from the early stages of the design process as opposes to adding it at later stages as it is most often done currently. In this methodology, sets of security solutions need to be identified and labeled with their cost (power consumption, consumption of computing resources, silicon area, memory, ...) and with a measure of their security. While it is known that an absolute security metric is difficult, or even impossible, to obtain, it is possible to measure system security relatively to known attacks for the system considered (Atzeni and Lioy, 2005). In most cases (e.g., different cryptographic algorithms) it is also possible to compare security provided by different solutions.

The remaining part of this paper is organized as follows: Section 2 discusses the related work; Section 3 introduces the design methodology that we have developed; Section 4 discusses the results obtained by applying our methodology to a case study; Section 5 presents a discussion on some key elements of the methodology.

2 RELATED WORK

Security is a new dimension that designers should consider throughout the design process, along with other metrics such as cost, performance, and power (Ravi et al., 2004). For this purpose, a security metric is required. Unfortunately, though, at the moment there is none available (Atzeni and Lioy, 2005): while there is the possibility to compare similar security solutions (e.g., different cryptographic algorithms) from the stand point of security and cost, at present there is no methodology to measure the security of systems. In this paper, among the other things, we propose a security metric that aggregates an evaluation of the security of all different security elements. The evaluation is done by considering the resistance of the security solutions to known attacks as well as their correspondence to security requirements.

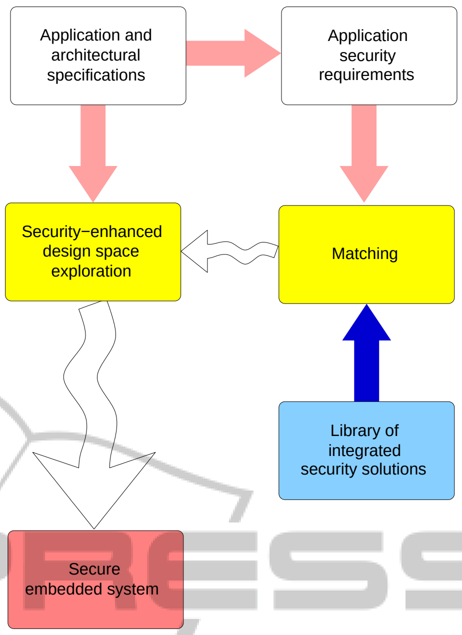
In the last years some effort has been put in developing security standards for systems and applications. The NIST FIPS 140-2 (NIST, 2002) standard categorizes cryptographic systems in four levels depending on the services they offer and on the algorithms they support. The International Organization for Standardization (ISO) has also defined a set of standards for security in ISO/IEC 27000-series. In particular, a standard related to application security (ISO/IEC 27034) is in early stages of its development (ISO/IEC, 2011). In this work we rely on these standards to specify security requirements.

In (Ravi et al., 2004; Atzeni and Lioy, 2005) it is stated importance of including security as objective in multi-objective design space exploration for embedded systems design with the purpose of reducing power and processing overhead. In these papers, though, no methodology for including security in design space exploration is proposed. In (Kocher et al., 2004) it is stated that, as opposed to the other design metrics (e.g., area, performance, power), security is currently specified by system architects in a vague and imprecise manner. The main problem is that security experts are the only people in a design team who have a complete understanding of the security requirements, although different aspects of the embedded system design process can affect security. Furthermore, (Kocher et al., 2004) suggests that design methodologies for secure embedded systems should include techniques for specifying security requirements in a way that can be easily communicated to the design team, and evaluated throughout the design cycle, and that any attempt to specify security requirements needs to address the *level of security* desired. During embedded system architecture design, techniques to map security requirements to different alternative solutions, and to explore the associated trade-offs in terms of cost, performance, and power consumption, would be invaluable in helping embedded system architects understand and make better design choices (Kocher et al., 2004). In (Juerjens, 2003) the significance of including security in design phase is emphasized as well. The authors underline that the main reason for not doing that is the lack of properly defined security requirements and they propose a language, named UMLsec, aimed at this purpose. UMLsec is a specification language that extends UML. In our work we use UML to specify security requirements, even though in a different way than the one of UMLsec. In (Bayrak et al., 2011) a methodology for automatic application of security measures at design time is presented; the scope of the paper, though, is limited to countermeasures for power analysis attacks.

In different works related to security the Analytic Hierarchy Process (AHP) is used to evaluate security solutions and to build some sort of security metrics. For example, in (Taddeo and Ferrante, 2009) AHP is used to evaluate the relevance of different security algorithms in the process of selecting the one that best suits with the run-time conditions of the system. In our work we use AHP as a way to sort security requirements based on their priority.

3 DESIGN METHODOLOGY

Most real modern embedded system designs are inherently complex, given that their functional specifications are rich and they must obey multiple other requirements such as, for example, the ones on cost, performance, time, and area. As a result, methodologies and frameworks are needed in order to help designers in the different design phases. Figure 1 shows how we propose to integrate security in the design of embedded systems. Security requirements of the application need to be defined by following a suitable model. Available security solutions are grouped in a library (i.e., an organized collection); this is done to favor reuse of security components in different projects. Application security requirements are matched with security solutions available in the library and, from this operation, multiple sets of possible security solutions - each one of them satisfying the security requirements in different ways - are obtained; these sets are then ordered and rated with respect to their relative security level and their conformity to security requirements. Each set of solutions is evaluated during design space exploration by considering its cost and its security evaluation. This security evaluation is done by means of a *security metric*; in this way, security becomes an additional dimension of the design space.


Application and architectural specifications
Application security requirements
Security−enhanced design space exploration
Matching
Library of integrated security solutions
Secure embedded system
Figure 1: The security-enhanced design methodology.

To summarize, we propose a modified design methodology that includes the following steps:

1. Define the application functional requirements.
2. Define the application security requirements.
3. Match the solutions contained in the library with the security requirements.
4. Define the design space considering both functional requirements and sets of security solutions.
5. Evaluate security solutions by using the security metric.
6. Perform a design space exploration of the whole system, including the security solutions.
7. Evaluate the results of the design space exploration process by using a proper metric that includes security.

Steps 1, 4, 6, and 7 are part of the normal design process of embedded systems (Alberto Ferrante et al., 2005), but some of them need to be modified to include security. Steps 2, 3, and 5 are specific to our security design methodology. In this paper, therefore, we concentrate our effort on defining how security requirements should be specified (step 2), how the security solutions can be matched with requirements (step 3), and how security solutions are evaluated (step 5). We also discuss the modifications that are necessary to the other steps, namely steps 4 and 7.

3.1 Security Requirements

Specifying security requirements is the first and fundamental step of our design methodology. Security requirements are non-functional requirements describing the security features that should be provided by the system. Requirements should be written by considering the possible security attacks that the system should be able to withstand. Ideally, a designer should be able to specify only that he wants a system to be secure with a given *security level*. Unfortunately, though, this is a too generic requirement that, at the moment, cannot be translated into more specific ones. Though, we can foresee the possibility to specify more generic requirements by specifying which are the attacks that the systems should be able to withstand and by developing a method to map security solutions with attacks. A further evolution of this mechanism would be to specify which are the standard security test that the system is supposed to pass: this will lead to a list of attacks and, by using the aforementioned mapping among attacks and security solutions, to a set of security requirements. How to translate generic requirements in more specific ones is outside the scope of this paper.

In this paper we consider security requirements in which security features are specified. For example, we may specify that communication encryption is required. A mechanism used to assign priorities to different mechanisms has also is also proposed. This method is based on Analytical Hierarchy Process (AHP).

3.1.1 Security Requirements Model

Security requirements have been modeled by means of UML; the UML model can be used as a template to define system-specific requirements by specifying the desired security solutions and their characteristics. As shown in Figure 2, different kinds of security requirements are represented by different classes in the diagram; all of them are derived from the superclass *SecurityMechanisms*. Whenever multiple different options, with different characteristics, are available for certain security mechanisms, a superclass is used to represent the generic requirement and subclasses are used to specify the details. In each class different attributes are specified. These attributes can be used to specify the details related to each requested security mechanism. For example, the attributes of the class *CryptographicAlgorithms* can be used to define the desired characteristics of the corresponding cryptographic algorithm (e.g., the key length).

Table 1: Table: Rating scale.

Intensity of Importance	Definition	Explanation
1	Equally important	The two factors contribute equally to the objective
3	Somewhat more important	Experience and judgment slightly favor one over the other
5	More important	Experience and judgment strongly favor one over the other
7	Much more important	Experience and judgment very strongly favor one over the other
9	Absolutely more important	The evidence is favoring one over the other
2, 4, 6, 8	Intermediate values	Importance in-between other levels

To specify the security requirements of a system, the classes corresponding to the desired requirements should be selected and their attributes should be specified. A mandatory attribute of every class is the one that specified the desired level of security; this parameter should conform to the format specified in Section 3.3. Designer has two ways to define requirements inside the class: they can only specify the required security level or they can specify the exact features and options that are required.

3.1.2 Priorities

It can often happen that not all of the requirements are equally important and, therefore, a mechanism for assigning them priorities may be required. In our methodology, these priorities are expressed through weights. The method used to derive weights is called AHP. AHP is a way to perform decision making that involves structuring multiple choice criteria into a hierarchy, assessing relative importance to these criteria, comparing alternatives for each criterion and determining an overall ranking of the alternatives (Coyle, 2004).

First step in performing AHP is decomposing the considered problem into sub-problems in order to form a multi-level, multi-target, and multi-factors structure called hierarchical structure. The next step is assigning priorities. Let us assume that we need to assign weights to a set of n attributes that, in our case, are the number of requirements (R). An essential part of this step is forming an $R \times R$ matrix, called A, which is expressing the relations among the attributes. Attributes are pair-wise compared with the purpose of deciding their relative importance. Relative importance is defined by using a set of predefined values as shown in Table 1. Each element a_{ij} of the matrix A denotes the relative weight of the i-th attribute with respect to the j-th one. Equation 1 shows the constraints that must be respected in this matrix.

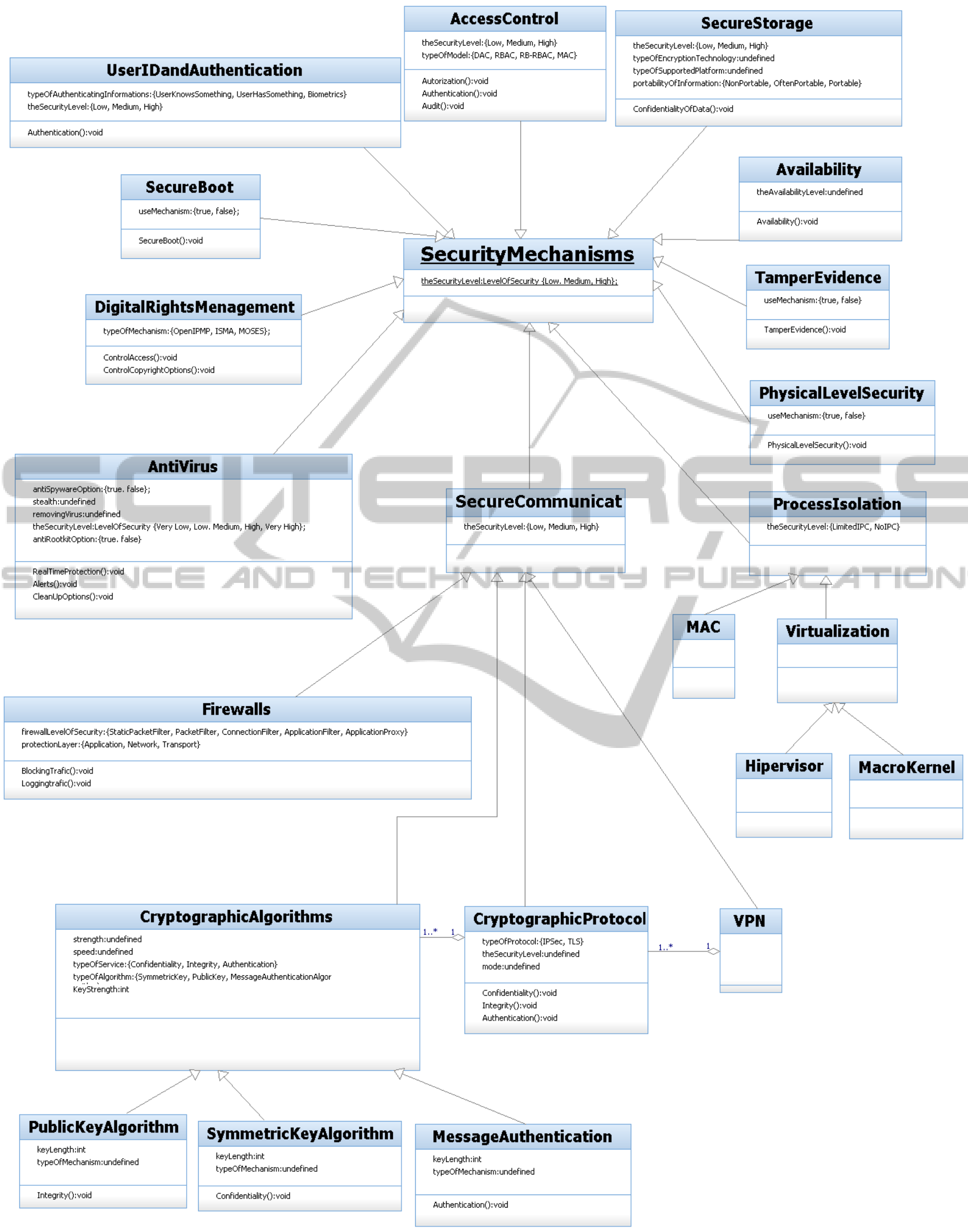

AccessControl
theSecurityLevel:{Low, Medium, High}
typeOfModel:{DAC, RBAC, RB-RBAC, MAC}
Autorization():void
Authentication():void
Audit():void
SecureStorage
theSecurityLevel:{Low, Medium, High}
typeOfEncryptionTechnology:undefined
typeOfSupportedPlatform:undefined
portabilityOfInformation:{NonPortable, OftenPortable, Portable}
ConfidentialityOfData():void
UserIDandAuthentication
typeOfAuthenticatingInformations:{UserKnowsSomething, UserHasSomething, Biometrics}
theSecurityLevel:{Low, Medium, High}
Authentication():void
SecureBoot
useMechanism:{true, false};
SecureBoot():void
Availability
theAvailabilityLevel:undefined
Availability():void
SecurityMechanisms
theSecurityLevel:LevelOfSecurity {Low, Medium, High};
TamperEvidence
useMechanism:{true, false}
TamperEvidence():void
DigitalRightsMenagement
typeOfMechanism:{OpenIPMP, ISMA, MOSES};
ControlAccess():void
ControlCopyrightOptions():void
PhysicalLevelSecurity
useMechanism:{true, false}
PhysicalLevelSecurity():void
AntiVirus
antiSpywareOption:{true. false};
stealth:undefined
removingVirus:undefined
theSecurityLevel:LevelOfSecurity {Very Low, Low. Medium, High, Very High};
antiRootkitOption:{true. false}
RealTimeProtection():void
Alerts():void
CleanUpOptions():void
SecureCommunicat
theSecurityLevel:{Low, Medium, High}
ProcessIsolation
theSecurityLevel:{LimitedIPC, NoIPC}
MAC
Virtualization
Firewalls
firewallLevelOfSecurity:{StaticPacketFilter, PacketFilter, ConnectionFilter, ApplicationFilter, ApplicationProxy}
protectionLayer:{Application, Network, Transport}
BlockingTrafic():void
Loggingtrafic():void
Hipervisor
MacroKernel
CryptographicAlgorithms
strength:undefined
speed:undefined
typeOfService:{Confidentiality, Integrity, Authentication}
typeOfAlgorithm:{SymmetricKey, PublicKey, MessageAuthenticationAlgor
KeyStrength:int
1..*
1
CryptographicProtocol
typeOfProtocol:{IPSec, TLS}
theSecurityLevel:undefined
mode:undefined
Confidentiality():void
Integrity():void
Authentication():void
1..*
1
VPN
PublicKeyAlgorithm
keyLength:int
typeOfMechanism:undefined
Integrity():void
SymmetricKeyAlgorithm
keyLength:int
typeOfMechanism:undefined
Confidentiality():void
MessageAuthentication
keyLength:int
typeOfMechanism:undefined
Authentication():void

Figure 2: General security requirements in embedded systems.

$$a_{ij} = \begin{cases} \frac{1}{a_{ji}}, & i \neq j \\ 1, & i = j \end{cases} \qquad (1)$$

Furthermore, matrix A is subject to a consistency analysis. When obtained from measured data, weights are said to be consistent if they are transitive, i.e., $a_{ik} = a_{ij} \cdot a_{jk}$ for all i, j, k. To verify this condition we need to find a vector z of order R such that $Az = \lambda z$; where z is an eigenvector of order R and λ is eigenvalue of the matrix A. For a consistent matrix the condition $\lambda = R$ need to apply. Nevertheless, in our case the elements of matrix A are based on human judgment and some amount of inconsistency may exist. Therefore, the previous condition is relaxed in such a way that the vector z satisfies the conditions $Az = \lambda_{max} z$ and $\lambda_{max} > R$, where λ_{max} is maximal eigenvector. Any difference between λ_{max} and R, measured by the Consistency Index shown in Equation 2, is an indication of inconsistency of judgments in matrix A.

$$CI = \frac{\lambda_{max} - R}{R - 1} \tag{2}$$

The Consistency Index is then used to compute the Consistency Ratio as shown in Equation 3.

$$CR = \frac{CI}{RI} \tag{3}$$

where RI stands for Random Index for consistency that is given in (Coyle, 2004) for matrices of different orders. To guarantee consistency of the matrix A, the Consistency Ratio should not be higher than 0.1.

Once the matrix A is well defined, the *Relative Value Vector* (RVV) that contains the weights assigned to the R requirements can be defined as shown in Equation 4.

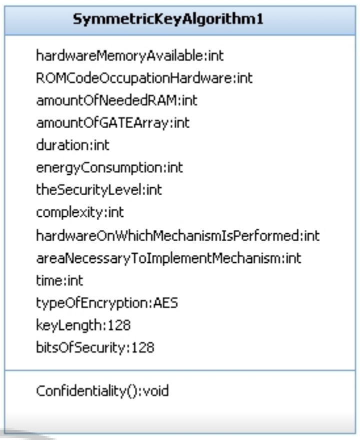
$$RVV = (w_1, w_2, ..., w_n). \tag{4}$$

As shown in Equation 5, the weights w_i are the eigenvectors of the matrix A.

$$w_i = \frac{\sum_{j=1}^{R} a_{ij}}{R \sum_{k=1}^{R} \sum_{j=1}^{R} a_{kj}}, \ i = \{1, R\} \tag{5}$$

3.2 Matching Requirements with Solutions

Requirements should be matched with the solutions contained in the library. Elements of the library are security solutions that have been characterized with respect to performances and security features. An example of a library element is shown in Figure 3. Based on given requirements, the desired security level of all classes and sub classes has to be calculated. Depending on the kind of requirement considered, different methods can be used for this purpose: some security solutions allow for a quantitative evaluation of the security level; some others just for a qualitative one. The quantitative approach can be used for all the solutions in which security can be defined quantitatively such as, for example, in the case of cryptographic algorithms where the resistance to brute force attacks of the considered algorithm can be used as a quantitative measure of security. In Table 2 the measure of security of different algorithms provided as bits of security in (Barker et al., 2012) was used to compute the security level of each algorithm.



SymmetricKeyAlgorithm1

```
hardwareMemoryAvailable:int
ROMCodeOccupationHardware:int
amountOfNeededRAM:int
amountOfGATEArray:int
duration:int
energyConsumption:int
theSecurityLevel:int
complexity:int
hardwareOnWhichMechanismIsPerformed:int
areaNecessaryToImplementMechanism:int
time:int
typeOfEncryption:AES
keyLength:128
bitsOfSecurity:128

Confidentiality():void
```

Figure 3: Representation of the library element.

Table 2: Quantitative approach for assignment of security levels.

[redacted]	Bits Of Security	Symmetric Key Algorithms	Hash	Authentication
0.2	80	2TDEA	SHA-1 SHA-224 SHA-256 SHA-384 SHA-512	SHA-1 SHA-224 SHA-256 SHA-384 SHA-512
0.4	112	3TDEA	SHA-224 SHA-256 SHA-384 SHA-512	SHA-1 SHA-224 SHA-256 SHA-384 SHA-512
0.6	128	AES-128	SHA-256 SHA-384 SHA-512	SHA-1 SHA-224 SHA-256 SHA-384 SHA-512
0.8	192	AES-192	SHA-384 SHA-512	SHA-224 SHA-256 SHA-384 SHA-512
1	256	AES-256	SHA-256 SHA-384 SHA-512	SHA-256 SHA-384 SHA-512

The mapping has been done by mapping the bits of security to a value ranging from 0 to 1, 1 being the highest possible security level.

For other classes, where the quantitative method cannot be applied, the qualitative one is used. As shown Equation 6, for these solutions the security level is calculated by considering the number of security features supported for the corresponding security solution. Features are the ones specified in Figure 2 and they are weighted, based on their importance, according to the weights provided by the designer.

$$L_i = \sum_{j=1}^{R} g_j \cdot v_{ji}, \ i = 1, 2, ..., S \tag{6}$$

In this equation R is the number of requirements, S is the number of solutions, g_j refers to the weight of the requirement r_j and v_{ji} shows if requirement r_j is fulfilled in Solution s_i ($v_{ji} = 1$ if requirement is fulfilled, or $v_{ji} = 0$ in the opposite case). The overall security level required, is calculated by using AHP to combine the security levels of the considered security solutions.

Solutions that are in the library are already described with their own security level. Two additional parameters are calculated in order to perform the matching. These two parameters are the level of missing features (F_i^-) and the level of additional features (F_i^+). These two values show how close the considered solution is to the requirements; F_i^- shows the number of features that are specified in the requirements but that are not supported by the considered solution; F_i^+ represents the number of features that are not specified in the requirements but that are present in the considered solution.

The required security level and the security level provided with the considered security mechanism is compared. If the quantitative approach is used, when the security level is higher than the one of the solution, F_i^- is calculated as a difference between these two numbers, and F_i^+ is 0. Otherwise, F_i^+ is computed as difference and F_i^- is 0. For the qualitative approach a comparison between the required security level and the one provided by the considered solution is done and F_i^- is calculated as a weighted sum of missing features; F_i^+ is computed as a weighted sum of added features. Thresholds can be defined for these values so that only suitable solutions are selected. By assigning this tree numbers (L_i, F_i^-, F_i^+) to all solutions in the library, we are able to order security solutions from the security stand point and to label them with their costs by using AHP.

3.3 Security Metric

A widely accepted management principle is that an activity cannot be managed if it cannot be measured. Security falls under this rubric. Metrics can be effective tools for security managers to discern the effectiveness of various components of their security programs, the security of a specific system, product or process, and the ability of staff or departments within an organization to address security issues for which they are responsible. Unfortunately, though, no suitable security metric was available in the literature (Ravi et al., 2004; Atzeni and Lioy, 2005). Therefore, we developed our own metric based on what we defined as *Security level* of the different security solutions as well as a rating on how well each solution fit with the corresponding security requirements.

For example, the security level of the *Secure Communication* class can be calculated by using Equation 7, where L_{ska}, L_{pka}, L_{ma} are the security levels of its subclasses. Weights w_{ska}, w_{pka}, w_{ma} are the weights of the symmetric key algorithm, the public key algorithm and the message authentication algorithm, respectively; these weights are determined by using AHP.

$$L_{sc} = w_{ska} * L_{ska} + w_{pka} * L_{pka} + w_{ma} * L_{ma} \tag{7}$$

AHP is applied again for all classes to compute the overall security level. This is done by using Equation 8, where w_{av}, w_{fw}, w_{sc} are weights, determined by using AHP, of the anti-virus, firewall and secure communication classes, respectively. L_{av}, L_{fw}, L_{sc} are the security levels associated to the aforementioned classes.

$$SL_{overall} = w_{av} * L_{av} + w_{fw} * L_{fw} + w_{drm} * L_{drm} + w_{sc} * L_{sc} \tag{8}$$

By applying AHP we can obtain the list of all possible sets of solutions with their security levels, levels of missing features and levels of additional features. With this list we are able to decide what are the most appropriate ones from the security stand point. Three thresholds based on the security level, the number of missing features, and the number of added features are used to provide the ability to filter the sets of solutions that should be considered in the design exploration phase. This provides designers the flexibility to define suitable constraints for the solutions to be considered. For example, designers may use the filter on the number of missing feature to restrict the design space only to the solutions that support all the required features.

Although having three numbers to characterize all solutions is much more descriptive and helpful in choosing the appropriate ones, in order to evaluate the solutions during the design exploration phase, these three numbers should be summarized in a single one that represents our security metric. If $S_i^- = 0$, the security metric is simply represented by L_i. If $S_i^- < 0$ (i.e., some desired requirements are not fulfilled) L_i is scaled by the number of non-satisfied requirements. Scaling is done in such a way that the higher the number of missing features, the lower number that represents whole solution are. Security metric is the number that represents the security level of one solution comparing to previously defined requirements of the system. The security metric number is obtained by aggregation of the three numbers that represent every solution. In case that the security metric is equal or greater then requested one, it means that all requirements are fulfilled with the same or even higher security level. In case that security metric number is lower than requested, it means that some of requirements are not fulfilled.

The security metric discussed above can be used alone to sort the security solutions. Though, this metric do not include any other system parameter and do not allow to perform any trade-off among security solutions and their costs. Metrics currently used in design of embedded systems include performance as well as energy consumption and/or area occupation. Energy consumption and area are considered as costs and the system is optimized to obtain the best possible trade-off among performances and cost. For example, the metric of Equation 9 is widely used to obtain the optimize the energy-performance trade-off.

$$OverallMetric = Time \cdot Power \tag{9}$$

Other metrics that attribute different weights to the different parameters (e.g., by considering the power n of one or more parameters) may also be used. The choice of a suitable metric depends on the kind system considered and on the goals of designers. Additionally to computing the values of the metric by using Equation 9, the values of the different parameters can be plotted in a diagram and the optimal point can be chosen among the ones that reside on the Pareto curve.

Security can be integrated in these kinds of metric as a parameter that should be maximized as opposed to time and energy that need to be minimized. Different sets of security solutions can, therefore, be evaluated by using a metric such as the one shown in Equation 10.

$$OverallMetric_s = \frac{Time \cdot Power}{SecurityMetric} \tag{10}$$

This equation can be modified with different parameters as explained for Equation 9. The same metric can be used for optimizing the entire system by also including security solutions. In this case the overall energy consumption and performances will be considered instead of the ones associated only to security solutions.

4 EXPERIMENTAL RESULTS

In this section we discuss a case study that is used both to show how our design methodology can be applied in practice and its effectiveness in designing an optimal system. The case study has been chosen to be simple and with a limited number of solutions to analyze. Furthermore, without leading to the generality of our approach and with the purpose of showing how the optimization process is effective on security solutions, we kept the base architecture of the system fixed (i.e., we have not changed the architectural parameters such as, for example, the number of functional units in the processor and the memory architecture). If the architectural parameters were changed, the performance figures of each security solution would have required to be evaluated for each new architecture.

The case study of choice is a Voice over IP (VoIP) phone. The phone is an embedded device based on a microprocessor that runs the VoIP application. The system may also include some hardware accelerators for executing specific functions. The phone is based on the Session Initiation Protocol (SIP) (Rosenberg et al., 2002); SIP is a network communications protocol commonly employed for VoIP.

In this case study we used an optimization metric that considers performances, power consumption, area and security. The metric is shown in Equation 11.

$$OverallMetric_s = \frac{Time \cdot Power \cdot Area}{SecurityMetric} \tag{11}$$

This equation is an extension of Equation 10 as discussed in Section 3.3. We added *Area* as a parameter to also account for the additional area that is used by possible hardware implementations of the security solutions. Considering only performances and power consumption is suitable for systems based on software-only implementations of the applications and of the security solutions; though, it may be convenient when also hardware solutions are considered. In fact, a metric based only on these two parameters does not consider the cost of the additional silicon area required by hardware solutions. Therefore, by using such a metric, these solutions would always been favored by their often better performances and power consumption.

For evaluating performances, we considered a 10-minute conversation in which we assumed that for 50% of the total time the user of the phone is speaking and for the other 50% he is listening. Considering an 8-bit mono 44.1kHz PCM encoding of voice, we used a data size of 105.84Mbit and 7.2Mbit both for transmission and reception, respectively. We assumed to have a phone book of 200kbit and user ID and authentication performed by using one 64-bit data block.

Parts of the system were simulated by using the Wattch (Brooks et al., 2000) simulator to obtain the data to be used in the optimization process. Wattch is an architectural simulator that provides the ability to estimate performances (execution time) and power consumption. In particular, we simulated the encoding and decoding parts of the SIP client. Encoding and decoding are, in fact, the most demanding parts of any SIP client; given the limitations of the Wattch simulator, we could not run a full implementation of a SIP client, but we used stand-alone encoders and

Table 3: Populated library of secure elements.

Name of the Mechanism	Energy (nJ)	The Security Level	Time (ns)	Hardware
AES128 Encryption SW	3283.69	0.6	2961.19	MIPS
AES128 Decryption SW	4053.84	0.6	3786.37	MIPS
AES128 Encryption HW 1	0.04609	0.6	8.17	HW Accelerator
AES128 Decryption HW 1	0.04609	0.6	8.17	HW Accelerator
AES128 Encryption HW 2	0.03750	0.6	8.17	HW Accelerator
AES128 Decryption HW 2	0.03750	0.6	8.17	HW Accelerator
AES128 Encryption HW 3	0.07734	0.6	8.17	HW Accelerator
AES128 Decryption HW 3	0.07734	0.6	8.17	HW Accelerator
AES256 Encryption	4654.18	1	4263.22	MIPS
AES256 Decryption	5830.10	1	5387.84	MIPS
DES Encryption	304.11	0.2	263.81	MIPS
DES Decryption	304.07	0.2	263.83	MIPS
3DES Encryption	829.64	0.4	712.76	MIPS
3DES Decryption	827.13	0.4	712.76	MIPS
HMAC-SHA1	4373.84	0.6	6232.15	MIPS
HMAC-SHA2	8135.28	1	12480.13	MIPS

decoders for G723 (Sun Microsystems, 1992), one of the encodings specified in the SIP protocol. For evaluating area we considered a reference area for the microprocessor (Wattch simulates a MIPS architecture) of 360,000 gates (Margarita Esponda, 1991). When hardware accelerators were adopted, their area was added to the one of the microprocessor for computing the total silicon area.

The library elements considered during this case study are the ones listed in Table 3. Three hardware solutions (Hamalainen et al., 2006) have been used; all of them are implementations of the AES cryptographic algorithm with a 128-bit key size, optimized for area, power, and speed, respectively.

The first step of the methodology is devoted to defining security requirements. We defined these requirements by considering the possible attacks that the system should withstand. The main ones are the following:

- Eavesdropping of the conversations.
- Modification of the conversations.
- Unauthorized use of the phone.
- Unauthorized access to the phone contacts.

Eavesdropping of conversations can be counteracted by encrypting conversations; modifications of the conversations can be detected by using authentication; unauthorized access to the phone can be prevented by using user authentication (e.g., a PIN); unauthorized access to the phone contacts can be prevented by encrypting the phone contacts (i.e., the contacts are stored in encrypted form and decrypted by using the user PIN as a key). These generic requirements have been transformed into the $R = 3$ requirements that are shown in Figure 4. w_1, w_2, and

Table 4: AHP matrix comparison.

	Secure Storage	User ID and Authentication	Secure Communication
Secure Storage	1	2	4
User ID and Authentication	1/2	1	3
Secure Communication	1/4	1/3	1

w_3 are the weights associated with *Secure Storage*, *User ID and Authentication*, and *Secure Communication*, respectively. According to our pairwise preferences among requirements specified in Table 4 and to Equation 5, weights are computed to be $w_1 = 0.121$, $w_2 = 0.344$ and $w_3 = 0.535$. In this case study we assigned priorities to different security requirements in such a way that the requirements are sorted, from the most important to the least one, as follows: *Secure Communication*, *User ID and Authentication*, *Secure Storage*.

The next step is to match the security solutions available in the library with security requirements. By considering the aforementioned requirements and the library of Table 3, 196 different sets of possible solutions are determined. Some of these sets of solutions are shown in Table 5.

For each set of solution the security level (L_i) the level of missing requirements (S_i^-) and the level of added requirements (S_i^+) need to be computed as explained in Section 3.2. Based on Equation 7 and Equation 8, the security levels of the *Secure Communication* class solutions and of the overall systems solutions are determined, together with the required overall security level. Once these parameters are computed, the security metric of each set of solutions can also be computed as described in Section 3.3. By using the security metric, the general metric can be computed and the optimal solution chosen.

In Table 5 we show the solutions that scored a lower overall metric. These solutions are to be considered, by following the evaluation criteria given by the chosen metric, as the best ones. The solution with the lowest value of the metric is the optimal one. This solution proposes the usage of the software implementation of AES 256 for Secure Storage; a software implementation of SHA-2 has been selected for for User ID and Authentication; a hardware implementation optimized for area of AES 128 has been selected as Symmetric Key Algorithm; a software implementation of the HMAC-SHA2 algorithm has been selected for Message Authentication. The required security level is 0.64; the security level obtained with the selected solution is is 0.755 and all requirements are fulfilled. The metric used allowed to choose the solution that gives the best balance among security, performances, energy, and area. An hardware implementation of AES 128 optimized for area has been

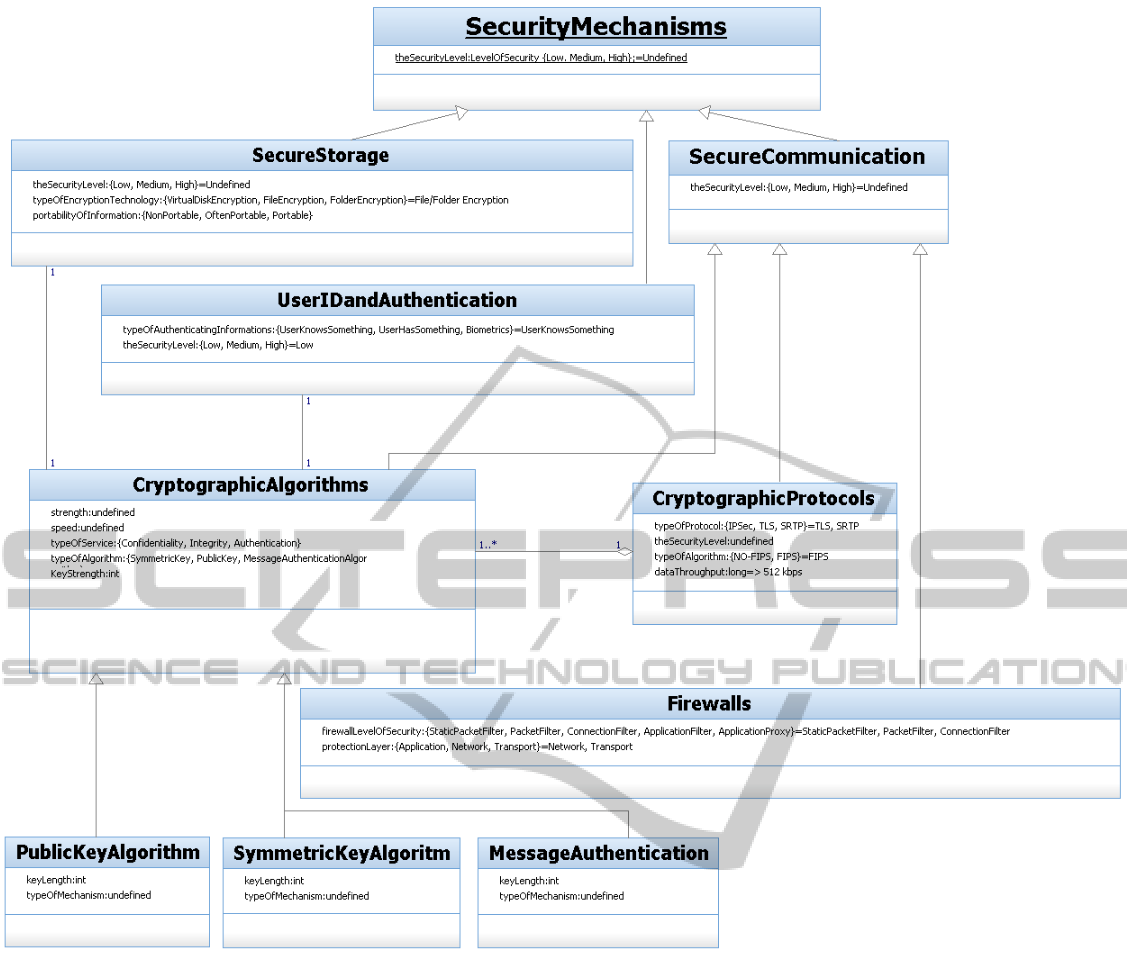

SecurityMechanisms
theSecurityLevel:LevelOfSecurity {Low, Medium, High};=Undefined
SecureStorage
theSecurityLevel:{Low, Medium, High}=Undefined
typeOfEncryptionTechnology:{VirtualDiskEncryption, FileEncryption, FolderEncryption}=File/Folder Encryption
portabilityOfInformation:{NonPortable, OftenPortable, Portable}
SecureCommunication
theSecurityLevel:{Low, Medium, High}=Undefined
UserIDandAuthentication
typeOfAuthenticatingInformations:{UserKnowsSomething, UserHasSomething, Biometrics}=UserKnowsSomething
theSecurityLevel:{Low, Medium, High}=Low
1
1
1
1
CryptographicAlgorithms
strength:undefined
speed:undefined
typeOfService:{Confidentiality, Integrity, Authentication}
typeOfAlgorithm:{SymmetricKey, PublicKey, MessageAuthenticationAlgor
KeyStrength:int
1..*
1
CryptographicProtocols
typeOfProtocol:{IPSec, TLS, SRTP}=TLS, SRTP
theSecurityLevel:undefined
typeOfAlgorithm:{NO-FIPS, FIPS}=FIPS
dataThroughput:long=> 512 kbps
Firewalls
firewallLevelOfSecurity:{StaticPacketFilter, PacketFilter, ConnectionFilter, ApplicationFilter, ApplicationProxy}=StaticPacketFilter, PacketFilter, ConnectionFilter
protectionLayer:{Application, Network, Transport}=Network, Transport
PublicKeyAlgorithm
keyLength:int
typeOfMechanism:undefined
SymmetricKeyAlgoritm
keyLength:int
typeOfMechanism:undefined
MessageAuthentication
keyLength:int
typeOfMechanism:undefined

Figure 4: UML representation of security requirements used in the Case Study.

chosen for encryption/decryption of communication. This allows the system to provide the required level of performances, yet saving energy, and by using a small additional silicon area. For secure storage (used for the phone book) the more secure AES 256 encryption algorithm is used. Given the limited amount of data that needs to be processed, a software implementation is evaluated to provide the best overall results. A software implementation of HMAC-SHA-2 have been chosen for message authentication as no hardware implementations was available in the library.

5 DISCUSSION

As shown in Section 4, the methodology presented in this paper can be used effectively to design optimized embedded systems in which security is included since the early stages of the design process. In this section we discuss some parts of the methodology that needs to be further studied and refined. These parts, in fact, showed some practical limitations during the development of the case study. These limitations, although important for the usability of the methodology, imply nothing on the effectiveness of the methodology itself. Furthermore, elements of the methodology (e.g., the metric) can be changed with a limited impact on the design flow presented in this paper.

As far as specifying security requirements is concerned, the method that we propose can help designers in defining security requirements. Though, this method is not ideal as it requires knowledge of security mechanisms. As previously mentioned, the ideal way of expressing security requirements for the designer would be specifying which are the security standards that the system should follow and/or the security tests that the system should be able to withstand. In this way the designer would be able to specify the requirements without a specific knowledge of the security mechanisms that are going to be used. This would require, though, a way to automatically map security solutions with requirements specified in

Table 5: Sets of security solutions with the lowest metric values.

[redacted] Storage	User Auth.	Connection Encryption	Connection Auth.	Security Level	Security Metric	Energy (J)	Time (s)	Area (kgates)	Overall Metric
AES-256	sha2	AES-256	HMAC-sha2	0.862	0.862	1873.185	2,272.048	0	4,935,026.588
AES-128	sha2	AES-256	HMAC-sha2	0.814	0.814	1872.556	2,271.468	0	5,225,368.086
AES-128 HS1	sha2	AES-256	HMAC-sha2	0.814	0.814	1871.088	2,270.121	6200	5,308,047.17
AES-128 HS2	sha2	AES-256	HMAC-sha2	0.814	0.814	1871.088	2,270.121	6400	5,310,946.15
AES-128 HS3	sha2	AES-256	HMAC-sha2	0.814	0.814	1871.088	2,270.121	7800	5,331,239.11
AES-256	sha2	AES-128	HMAC-sha2	0.755	0.755	1715.343	2,122.711	0	4,820,198.771
AES-256	sha2	AES-128 HS1	HMAC-sha2	0.755	0.755	1338.615	1,782.960	6200	3,213,929.63
AES-256	sha2	AES-128 HS2	HMAC-sha2	0.755	0.755	1338.614	1,782.960	6400	3,215,682.58
AES-256	sha2	AES-128 HS3	HMAC-sha2	0.755	0.755	1338.619	1,782.960	7800	3,227,980.44
AES-128	sha2	AES-128	HMAC-sha2	0.707	0.707	1714.714	2,122.130	0	5,146,883.294

this way; this mapping may be complex and security solutions to be adopted may depend on the kind of system considered.

Assigning priorities to different security requirements is now done by using AHP. This provides flexibility, but it also introduces a further level of complexity for the developers: computing the weights values that can satisfy the consistency check may be difficult and should be simplified to make the methodology usable.

The library of security elements has been made for favoring reuse of solutions previously developed. The profiling of these solutions should be done for the architectures considered and, if possible, being updated when changes to this architecture (e.g., a change in the number of functional units) are done. Furthermore, hardware elements may be implemented by using different technologies and this may impact performances and energy consumption. The library description should be modified to account all of these possible variations. In particular, the description of the library elements should allow for the addition of multiple performance and energy figures for different architectures and technologies. A part from this, we believe that, in most of the design environments, where design solutions are often kept within certain bounds (e.g., the same reference architecture and/or technology is always reused), the characterization of library elements should not impose any dramatic overhead.

The fact that UML, that is the language that we used to model both the security requirements and the library of security solutions, is a general purpose notation limits its suitability for modeling some particular domains. However, UML provides a set of extension mechanisms allowing the customization and extension of its own syntax and semantics in order to adapt to certain application domains; our plan is to improve modeling of security requirements and the library of solutions by using these mechanisms. UML profiles and ontologies will be considered for improving the models.

6 CONCLUSIONS AND FUTURE WORK

In this paper we presented a security-enhanced design methodology for embedded systems. This methodology, built over the classic one, provides the ability to specify security requirements of the applications and to evaluate different security solutions with the purpose of obtaining a secure and optimized the system. The case study that we have discussed in the paper shows how the methodology can be applied in practice and how effective it is both in including security in the design process and in optimizing it.

Our plans for the future include improving the methodology and its steps by solving the problems discussed in Section 5. Furthermore, we are currently working on integrating our methodology into a multi-objective architectural optimization tool named Multicube Explorer (Zaccaria et al., 2010). In fact, we believe that integrating our design methodology into existing design tools is a fundamental step for fostering its adoption.

ACKNOWLEDGEMENTS

This work was partially supported and funded by the Hasler Foundation under the Project SDES (Grant No. 12001). The paper reflects only the view of the authors; the Hasler Foundation is not liable for any use that may be made of the information contained herein.

REFERENCES

Alberto Ferrante, Giuseppe Piscopo, and Stefano Scaldaferri (2005). Application-driven Optimization of VLIW Architectures: a Hardware-Software Approach. In *Real-Time and Embedded Technology Applications*, pages 128–137, San Francisco, CA, USA. IEEE Computer Society.

Alippi, C., Piuri, V., and Scotti, F. (2004). *High-level Design of Composite Systems*. Springer, Berlin.

Atzeni, A. and Lioy, A. (2005). Security metrics. First Workshop on Quality Protection, Mina.

Barker, E., Barker, W., Burr, W., Polk, W., Smid, M., Gallagher, P. D., and For, U. S. (2012). *NIST Special Publication 800-57 Recommendation for Key Management Part 1: General*, chapter 5, pages 62–66. National Institute of Standards and Technologies.

Bayrak, A., Regazzoni, F., Brisk, P., Standaert, O.-X., and Ienne, P. (2011). A first step towards automatic application of power analysis countermeasures. In *Design Automation Conference (DAC), 2011 48th ACM/EDAC/IEEE*, pages 230–235.

Brooks, D., Tiwari, V., and Martonosi, M. (2000). Wattch: a framework for architectural-level power analysis and optimizations. In *Proceedings of the 27th annual international symposium on Computer architecture (ISCA'00)*, pages 83–94.

Coyle, G. (2004). Analytic Hierarchy Process. www.booksites.net/download/coyle/student_files/AHP_Technique.pdf.

Hamalainen, P., Alho, T., Hannikainen, M., and Hamalainen, T. (2006). Design and implementation of low-area and low-power aes encryption hardware core. In *Digital System Design: Architectures, Methods and Tools, 2006. DSD 2006. 9th EUROMICRO Conference on*, pages 577–583.

ISO/IEC (2011). *ISO/IEC 27034 Guidelines for application security*.

Juerjens, J. (2003). *Secure Systems Development with UML*. SpringerVerlag.

Kocher, P., Lee, R., McGraw, G., and Raghunathan, A. (2004). Security as a new dimension in embedded system design. In *Proceedings of the 41st annual Design Automation Conference*, DAC '04, pages 753–760, New York, NY, USA. ACM. Moderator-Ravi, Srivaths.

Koopman, P. (2004). Embedded system security. *Computer*, 37:95–97.

Margarita Esponda, R. R. (1991). *The RISC Concept - A Survey of Implementations*.

NIST (2002). *Security Requirements for Cryptographic Modules, FIPS 140-2*. National Institute of Standards and Technology, Information Technology Laboratory.

Palermo, G., Silvano, C., and Zaccaria, V. (2008). An efficient design space exploration methodology for on-chip multiprocessors subject to application-specific constraints. In *Application Specific Processors, 2008. SASP 2008. Symposium on*, pages 75 –82.

Ravi, S., Raghunathan, A., Kocher, P., and Hattangady, S. (2004). Security in embedded systems: Design challenges. *ACM Trans. Embed. Comput. Syst.*, 3:461–491.

Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and Schooler, E. (2002). Request for Comments: 3261 - SIP: Session Initiation Protocol. http://tools.ietf.org/html/rfc3261.

Seo, S.-H. and Cho, T. (2012). An access control mechanism for remote control of home security system. In *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on*, pages 93 –98.

Sun Microsystems, I. (1992). G723 source code. http://www.codeforge.com/article/874.

Taddeo, A. V. and Ferrante, A. (2009). Run-time selection of security algorithms for networked devices. In *Proceedings of the 5th ACM symposium on QoS and security for wireless and mobile networks*, Q2SWinet '09, pages 92–96, New York, NY, USA. ACM.

Viega, J. and Thompson, H. (2012). The state of embedded-device security (spoiler alert: It's bad). *Security Privacy, IEEE*, 10(5):68 –70.

Zaccaria, V., Palermo, G., Castro, F., Silvano, C., and Mariani, G. (2010). Multicube explorer: An open source framework for design space exploration of chip multi-processors. In *Architecture of Computing Systems (ARCS), 2010 23rd International Conference on*, pages 1–7.