On Generating Security Implementations from Models of Embedded Systems

Mehrdad Saadatmand, Antonio Cicchetti, Mikael Sjödin
Mälardalen Real-Time Research Centre (MRTC)
Mälardalen University, Västerås, Sweden
{mehrdad.saadatmand, antonio.cicchetti, mikael.sjodin}@mdh.se

***Abstract*—Designing secure embedded systems is a challenging task. Many of the challenges unique to embedded systems in this regard are due to the constraints that these systems have and thus impacts that security features will have on other properties of the system. Therefore, security decisions should be considered from early phases of development and together with other requirements. In model-driven methods, this means including security features in the design models. On the other hand, code generation from models is one of the promises of model-driven approaches. In this paper, by discussing the impacts of security design decisions on timing properties, we present the idea of automatic security code generation. We identify what issues a model for an embedded system should be able to answer and cover so that the security implementations that are later generated from it, will be consistent with the timing constraints and specifications of the system.**

***Index Terms*—Embedded security; MDA; Code generation; UML modeling**

I. Introduction

In the design of systems, security should be considered from early phases of development and along with other aspects of the system. While this approach to security is important for consistent and efficient security decisions, it becomes critical in case of embedded systems. Due to resource constraints in embedded systems, it is important to perform careful balance among different properties to satisfy all the requirements. Therefore, security should not be considered just as an addition of features but as a new dimension and metric [1].

Regarding design complexity of embedded systems, model-driven methods are a promising approach in raising abstraction levels and coping with the complexity of embedded systems. However, due to the characteristics of embedded systems, security requirements cannot be considered in separation from other requirements, and the modeling solutions that are used should be able to model security aspects along with other requirements such as timing, performance, and power consumption. This is especially important not to just document security requirements in the model, but also to enable analysis of them and their impacts on other requirements of the system, and generation of code that includes security features and implementations. The possibility to perform such analyses is the key to ensure correct design of an embedded system and that the code to be generated will be consistent with the specification. However, it should also be noted that the actual behavior of the generated code at runtime may deviate from what is specified in the model and expected. One reason is that some information may only be available at runtime. These deviations can be detected and controlled by using runtime verification and monitoring methods [2].

In this paper, considering challenges of designing secure embedded systems, we discuss what is needed at model level to enable proper security code generation. We do this by identifying necessary analyses that are required to realize implications of security design decisions and therefore predict the side effects of generated security implementations on other aspects, particularly timing properties.

The remainder of the paper is structured as follows. Section II discusses security challenges in embedded systems and implications of security design decisions in general. In Section III, we describe automatic payment system for toll roads, and explain the relation between timing and security requirements in this system. In Section IV, we will have a look at several UML profiles for modeling security and discuss their suitability for generation of security implementations. We propose a solution for modeling security by extending MARTE [3], and present our partial work on that (in the scope of this work, we focus only on UML profiles and not other ways of defining domain specific languages). Finally, Section V summarizes the paper and states how we continue with the work and possible future directions.

II. Security in Embedded Systems

Security is an aspect that is often neglected in the design of embedded systems. However, the use of embedded systems for critical applications such as controlling power plants, vehicular systems control, and medical devices makes security considerations even more important. This is due to the fact that there is now a tighter relationship between safety and security in these systems.

Also because of the operational environment of embedded systems, they are prone to specific types of security attacks that might be less relevant for other systems such as a database inside a bank. Physical and side channel attacks [1] are examples of these types of security issues in embedded systems that bring along with themselves requirements on hardware design and for making systems tamper-resistant. Examples of side channels attack could be the use of time and power measurements and analysis to determine security keys and types of used security algorithms.

Increase in use and development of networked and connected embedded devices also opens them up to new types of security issues. Features and devices in a car that communicate with other cars (e.g., the car in front) or traffic data centers to gather traffic information of roads and streets, use of mobile phones beyond just making phone calls and for purposes such as buying credits, paying bills,and transferring files (e.g. pictures, music,etc.) are tangible examples of such usages in a networked environment.

Besides physical and side channel attacks, often mobility and ease of access of these devices also incur additional security issues. For example, sensitive information other than user data, such as proprietary algorithms of companies, operating systems and firmwares, are also carried around with these devices and need protection.

A. *Implications of Introducing Security*

Because of the constraints and resource limitations in embedded systems, satisfying a non-functional requirement such as security requires careful balance and trade-off with other properties and requirements of the systems such as performance and memory usage. Therefore, introducing security brings along its own impacts on other aspects of the systems. This further emphasizes the fact that security cannot be considered as a feature that is added later to the design of a system and needs to be considered from early stages of development and along with other requirements.

From this perspective, there are many studies that discuss implications of security features in embedded systems such as [1]. Considering the characteristics of embedded systems, major impacts of security features are on the following aspects: Timing and Performance, Power Consumption, Flexibility and Maintainability, and Cost.

Considering these points, the security code that is generated for an embedded system should be from a model that satisfies the aforementioned criteria. This means that the model should contain enough information to enable impact analysis of security features (such as timing and performance), and ensure that they are in line with the system specification before generating code from the model. In the scope of this work, we focus on the timing costs of security mechanisms that are important for schedulability analysis and performance of a system, particularly in real-time embedded systems.

III. Automatic Payment System Example

Figure 1 shows internal interactions of a real-time embedded device in vehicles for automatic payment system in toll roads. The main goal in the design of this system is to allow a smoother traffic flow and reduce waiting times at tolling stations. This is an example of systems in which the impact of security features on timing properties are important and critical. The sequence diagram shows that when a payment station, through its camera, detects that a vehicle is approaching, it starts communicating with the vehicle and sends information such as the amount to pay to the vehicle. The vehicle, then shows this information to the driver through its User Interface

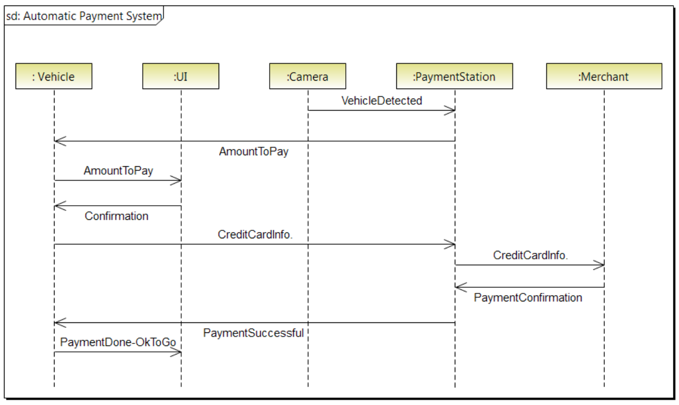

Fig. 1. Automatic Payment Systems for toll roads.

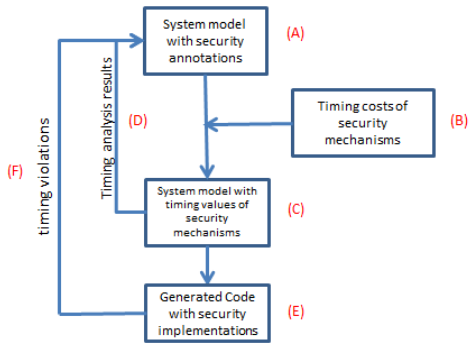
System model with security annotations
(A)
Timing costs of security mechanisms
(B)
System model with timing values of security mechanisms
(C)
Timing analysis results
(D)
Generated Code with security implementations
(E)
timing violations
(F)

Fig. 2. Suggested Approach

(UI). Upon confirmation of this payment by the driver, the vehicle sends credit card information to the station through a secure wireless connection. However, in this system, not only there are several security requirement, but also we have timing requirements as well. For instance, there is a critical time window from the moment that the camera detects a vehicle until the time it reaches the tolling station. It is within this time window that a successful payment transaction should be done; otherwise, the vehicle has to stop.

To implement such a system while ensuring the satisfaction of timing requirements, it is necessary to take into account the timing costs of security mechanisms that are used to implement security requirements of the system. For example, there are operations such as the transfer of CreditCardInfo that not only require encryption to protect sensitive data, but also have constraints on their execution times and cannot just take any arbitrary amount of time to execute.

To achieve this, the approach depicted in Figure 2 is suggested.

To enable the generation of appropriate security implementations, with respect to the timing constraints of the system, the following challenges are identified:

1) Modeling security mechanisms with enough detail to enable both timing analysis on the model and generation of the code implementing them,
2) Obtaining timing costs of security mechanisms,

3) Generating code for security mechanisms and detecting possible timing violations of the generated code at run-time.

The first challenge is discussed in the following section. To get the timing costs of security mechanisms, we rely on studies such as [4] that have done such measurements. To solve the third challenge, some hints are provided in the last section, but we leave its thorough discussion and implementation as a future work.

IV. Modeling Security Mechanisms

In this section, we discuss how to model security mechanisms, namely confidentiality, for our example system.

A. *Current Solutions for Modeling Security*

There are several efforts on defining UML profiles for security. For example, SecureUML [5] focuses on modeling role-based access control. AuthUML [6] provides a framework for analysis of access control requirements. [7] introduces a set of stereotypes for specification of vulnerabilities that serve as guidelines for developers to avoid them during implementation. UMLsec [8] offers a broader range of security concepts and comes with an analysis tool. Article [9] tries to offer a solution for modeling security along with timing characteristics of the system using UMLsec and MARTE.

One main issue with these modeling profiles is that modeling of security requirements is often considered in separation from other requirements such as timing [9], [10]. One solution could be to combine security profiles with other profiles that enable modeling requirements of embedded real-time systems and their analysis such as MARTE. However, it should be noted that combining different UML profiles can be tricky as these profiles can have overlapping and conflicting semantics and notations. This issue can be even trickier remembering that most of available security profiles are limited in the sense that they usually focus on a certain aspect of security and several of them may need to be combined as well [10]. Supporting hardware modeling and hardware devices with built-in security mechanisms is also another issue that is important for evaluating different deployment scenarios and is often not covered in security profiles. We have discussed this issue with more details in [11].

Finally, to generate code that includes implementations of security mechanisms from a model of an embedded system, the modeling concepts for security should provide the necessary information to derive code. This level of information is equally important to enable certain types of analyses at model level such as performing schedulability analysis by taking into account execution times of security features (e.g. encryption/decryption) or energy consumption analysis. For example, execution time and energy consumption of a block cipher algorithm can vary depending on the used algorithm, number of rounds, key size and so on. Therefore, these influencing parameters are required to be annotated at the model to enrich and make analysis more accurate. However, many of the currently available security profiles do not provide sufficient semantics to model and include the details necessary to perform these types of analyses and generate code.

B. *Modeling Security Using MARTE*

In this section, we discuss how MARTE modeling language can help to include timing costs of security mechanisms and include them in timing analysis of the system.

In order to alleviate the mentioned issues regarding security modeling in embedded systems, we propose extending MARTE with security concepts and building modeling semantics for security upon it. MARTE offers rich semantics for modeling non-functional requirements in real-time embedded systems and provides dedicated packages for schedulability and performance analysis. It also includes concepts for modeling deployment, hardware and annotating models with energy usage values. By extending MARTE with security concepts, it becomes possible to include impacts of security design decisions in the model for timing analysis, and evaluate their side effects before starting the implementation phase. Therefore, the code that will be generated from these models will better satisfy the requirements and constrains of an embedded system with less unknown and unmanaged side effects on other properties of the system such as timing, energy consumption, and memory usage.

Figure 3 shows part of our suggested MARTE extension for modeling block ciphers.

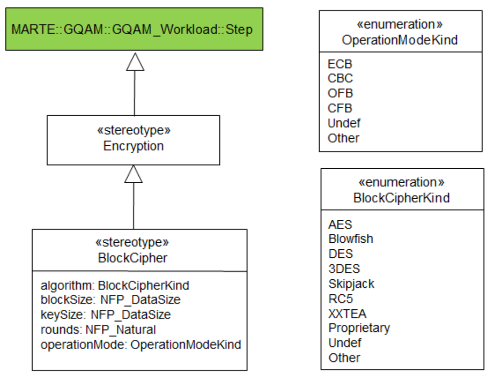

Fig. 3. Definition of BlockCipher stereotype

Using such concepts we can now annotate the operation of sending the CreditCardInfo, in the Payment System Example mentioned before, with the information that follows:

≪BlockCipher≫ CreditCardInfo() {algorithm=AES , blockSize=(128,bit), keySize=(128,bit), rounds=12, operationMode=ECB}

C. *Introducing Timing Costs of Security Mechanisms*

So far, we have managed to annotate sensitive operations in the model, such as CreditCardInfo, with parameters (type of algorithm, blocksize, keysize,etc.) of the encryption algorithms that are selected to protect them. This information is not only required to generate code that implements each selected encryption algorithm, but also enables us now to evaluate their timing costs at model level by using the result of studies such as [4] and [12] that have performed measurements of timing costs of encryption algorithms. We assume the existence of such measurements for the platform used in the automatic payment system example, in the form of Table I.

Algorithm	Key Size	BlockSize	Rounds	OperationMode	Execution Time (Bytes/Sec)
AES	128	128	10	ECB	490
AES	192	128	12	ECB	560
AES	256	128	14	ECB	710
...					

TABLE I
EXECUTION TIMES OF ENCRYPTION ALGORITHMS

Using this information, execution times of modeled encryption algorithms can be determined. In [11], we have discussed in more detail how the model can also be analyzed for energy costs of security mechanisms using a similar approach. The result would be similar to what follows:

≪GaCommStep≫ ≪BlockCipher≫ CreditCardInfo() {algorithm=AES , blockSize=(128,bit), keySize=(128,bit), rounds=10, operationMode=ECB, msgSize=(150,B), execTime=(306,ms,min,calc) }

GaCommStep is a MARTE concept which is a specialization of MARTE Step to describe communication workloads and is used in generic quantitative analysis contexts. Specification of execution time values are done here based on MARTE NFP concepts.

This way, the impact of security requirements on timing requirements in the system are identified. At this point, it is now possible to determine whether the chosen security mechanism is feasible considering specification and constraints on the allowed execution times. If not, blocksize, keysize, number of rounds, operationmode or even the size of the input message can be tweaked to balance security level with timing properties. This is done by iterating over steps A, B, C, and D of Figure 2. That is, timing costs for security mechanisms in the original model (A) are calculated resulting in a model with timing values for its security mechanisms using the MARTE concepts introduced above. These values are then checked against the timing specifications of the system. If violations are detected, the user modifies security mechanisms in the original model and goes through steps B, C and D again. After this phase, it is feasible to generate implementation of the defined security features for CreditCardInfo.

While, this approach seems to also enable energy consumption analysis on the model, this topic deserves a separate study; especially that detecting energy consumption violations later at runtime is a much bigger challenge than the detection of timing violations.

V. NEXT STEPS AND FUTURE WORK

In this paper, we presented the idea of generating security implementations from models of embedded systems. The challenges of designing secure embedded systems were identified. We discussed impacts of security on other requirements on the system, namely timing requirements, and the importance of trade-off analysis among requirements to predict the side effects of the generated code. Therefore, to generate security implementations, it was realized that the main challenge is at the model level so that the generated code respects the constraints of embedded systems. We proposed using MARTE as the basis for modeling embedded systems to enable necessary analyses on security decisions before generating code for them. However, as pointed out, timing violations can still happen at runtime. Therefore, it is needed to relate requirements in the model to their corresponding implementations in the generated code, and report any timing violations back to the user at the model level. As a solution to develop this feature, we are investigating suitability of Java Modeling Language (JML) to annotate the code and define pre/post-conditions for generated methods as suggested in [13].

REFERENCES

[1] P. Kocher, R. Lee, G. McGraw, and A. Raghunathan, "Security as a new dimension in embedded system design," in *Proceedings of the 41st annual Design Automation Conference*, ser. DAC '04, 2004, pp. 753–760, moderator-Ravi, Srivaths.

[2] S. Colin and L. Mariani, "Run-time verification," in *Model-Based Testing of Reactive Systems*, ser. Lecture Notes in Computer Science, M. Broy, B. Jonsson, J.-P. Katoen, M. Leucker, and A. Pretschner, Eds., vol. 3472. Springer Berlin / Heidelberg, 2005, pp. 525–555.

[3] MARTE specification version 1.0 (formal/2009-11-02), http://www.omgmarte.org, Last Accessed: June 2011.

[4] A. Nadeem and M. Javed, "A performance comparison of data encryption algorithms," in *First International Conference on Information and Communication Technologies, ICICT 2005.*, 2005, pp. 84 – 89.

[5] T. Lodderstedt, D. A. Basin, and J. Doser, "Secureuml: A uml-based modeling language for model-driven security," in *Proceedings of the 5th International Conference on The Unified Modeling Language*, ser. UML '02. London, UK: Springer-Verlag, 2002, pp. 426–441.

[6] K. Alghathbar and D. Wijesekera, "authuml: a three-phased framework to analyze access control specifications in use cases," in *FMSE '03: Proceedings of the 2003 ACM workshop on Formal methods in security engineering*. New York, NY, USA: ACM, 2003, pp. 77–86.

[7] K. P. Peralta, A. M. Orozco, A. F. Zorzo, and F. M. Oliveira, "Specifying security aspects in uml models," in *First International Modeling Security Workshop*, ser. MODSEC08, Toulouse, France, September 2008.

[8] J. Jürjens, "Umlsec: Extending uml for secure systems development," in *UML '02: Proceedings of the 5th International Conference on The Unified Modeling Language*. London, UK: Springer-Verlag, 2002, pp. 412–425.

[9] V. Thapa, E. Song, and H. Kim, "An approach to verifying security and timing properties in uml models," in *Engineering of Complex Computer Systems (ICECCS), 2010 15th IEEE International Conference on*, 2010, pp. 193 –202.

[10] R. J. Rodríguez, J. Merseguer, and S. Bernardi, "Modelling and Analysing Resilience as a Security Issue within UML," in *SERENE'10: Proceedings. of the 2nd International Workshop on Software Engineering for Resilient Systems*. ACM, 2010, accepted for publication.

[11] M. Saadatmand, A. Cicchetti, and M. Sjödin, "On the need for extending marte with security concepts," in *International Workshop on Model Based Engineering for Embedded Systems Design (M-BED 2011)*, March 2011.

[12] J. Lee, K. Kapitanova, and S. H. Son, "The price of security in wireless sensor networks," *Computer Networks*, vol. 54, pp. 2967–2978, December 2010.

[13] J. Lloyd and J. Jürjens, "Security analysis of a biometric authentication system using umlsec and jml," in *Proceedings of the 12th International Conference on Model Driven Engineering Languages and Systems*, ser. MODELS '09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 77–91.