Identification of Authenticity Requirements in Systems of Systems by Functional Security Analysis

Andreas Fuchs and Roland Rieke
Fraunhofer Institute for Secure Information Technology (SIT)
Rheinstrasse 75, 64295 Darmstadt, Germany
Email: {andreas.fuchs,roland.rieke}@sit.fraunhofer.de

Abstract

Cooperating systems typically base decisions on information from their own components as well as on input from other systems. Safety critical decisions based on cooperative reasoning, such as automatic emergency braking of a vehicle, raise severe concerns to security issues. In this paper we address the security engineering process for such systems of systems. The presented authenticity requirements elicitation method is based on functional dependency analysis. It comprises the tracing down of functional dependencies over system boundaries right onto the origin of information. A dependency graph with a safety critical function as root and the origins of decision relevant information as leaves is used to deduce a set of authenticity requirements. This set is comprehensive and defines the maximal set of authenticity requirements from the given functional dependencies. Furthermore, the proposed method avoids premature assumptions on the architectural structure and mechanisms to implement security measures.

1. Introduction

Architecting novel mobile systems of systems poses new challenges to getting the dependability and specifically the security requirements right as early as possible in the system design process. Security engineering is one important aspect of dependability [1]. The security engineering process addresses issues such as how to identify and mitigate risks resulting from connectivity and how to integrate security into a target architecture [2]. Security requirements need to be explicit, precise, adequate, non-conflicting with other requirements and complete [7].

A typical application area for mobile systems of systems are vehicular communication systems in which vehicles and roadside units communicate in ad hoc manner to exchange information such as safety warnings and traffic information. As a cooperative approach, vehicular communication systems can be more effective in avoiding accidents and traffic congestion than current technologies where each vehicle tries to solve these problems individually. However, introducing dependence of possibly safety critical decisions in a vehicle on information from other systems, such as other vehicles or roadside units, raises severe concerns to security issues. Security is an enabling technology in this emerging field because without security some applications within those systems of systems would not be possible at all. In some cases security is the main concern of the architecture [3].

The first step in the design of an architecture for a novel system of systems is the requirements engineering process. With respect to security requirements this process typically covers at least the following activities [4], [5], [6]

- the identification of the target of evaluation and the principal security goals and the elicitation of artifacts (e.g. use case and threat scenarios) as well as risk assessment
- the actual security requirements elicitation process
- a requirements categorisation and prioritisation, followed by requirements inspection

In this paper we address the security requirements elicitation step in this process. We present a model-based approach to systematically identify security requirements for system architectures to be designed for cooperative applications in a systems of systems context. Our contribution comprises the following distinctive features.

Identification of a consistent and complete set of authenticity requirements. We base our method on the assumption that the overall security goal with respect to authenticity requirements is: *For every safety critical action in a system of systems all information that is used in the reasonig process that leads to this action has to be authentic*.

To achieve this, we first derive a functional model by identification of atomic actions and functional dependencies in a use case description. From this model we generate a dependency graph with a safety critical function as root and the origins of decision relevant information as leaves. Based on this graph, we deduce a set of authenticity requirements that is comprehensive and defines the maximal set of authenticity requirements from the given functional dependencies.

Security mechanism independence. The most common problem with security requirements is that they tend to be replaced with security-specific architectural constraints that may unnecessarily constrain the choice of the most appropriate security mechanisms [8].

In our approach we avoid to break down the overall security requirements to requirements for specific components or communication channels prematurely. So the requirements identified by this approach are independent of decisions not only on concrete security enforcement mechanisms to use, but also on the structure, such as hop-by-hop versus end-to-end security measures.

Throughout this paper we use the following terminology taken from [1]: A *system* is an entity that interacts with other entities, i.e., other systems. These other systems are the *environment* of the given system. A *system boundary* is the common frontier between the system and its environment. Such a system itself is composed of *components*, where each component is yet another system. Furthermore, in [1] the *dependence* of system A on system B represents the extent to which system A's dependability is affected by that of system B. Our work though focuses on purely functional aspects of dependence and omits quantitative reasoning.

For the approach proposed, we describe the *function* of such a system by a *functional model* and treat the components as atomic and thus we do not make preliminary assumptions regarding their inner structure. Rather, the adaption to a concrete architecture is considered to be a task within a follow-up refinement and engineering process.

2. Related work

The development of new security relevant systems that interact to build new systems of systems requires the integration of a security engineering process in the earliest stages of the development life-cycle. This is specifically important in the development of systems where security is the enabling technology that makes new applications possible.

A comprehensive concept for an overall security requirements engineering process is described in detail in [5]. The authors propose a 9 step approach called SQUARE (Security Quality Engineering Methodology). The elicitation of the security requirements is one important step in the SQUARE process. In [6] several concrete methods to carry out this step are compared. These methods are based on misuse cases (MC), soft systems methodology (SSM), quality function deployment (QFD), controlled requirements expression (CORE), issue-based information systems (IBIS), joint application development (JAD), feature-oriented domain analysis (FODA), critical discourse analysis (CDA) as well as accelerated requirements method (ARM). A comparative rating based on 9 different criteria is also given but none of these criteria measures the completeness of the security requirements elicited by the different methods.

A similar approach based on the integration of Common Criteria (ISO/IEC 15408) called SREP (Security Requirements Engineering Process) is described in [4]. However the concrete techniques that carry out the security requirements elicitation process are given only very broadly. A threat driven method is proposed but is not described in detail.

In [7] anti-goals derived from negated security goals are used to systematically construct threat trees by refinement of these anti-goals. Security requirements are then obtained as countermeasures. This method aims to produce more complete requirements than other methods based on misuse cases. The refinement steps in this method can be performed informally or formally.

In [8] different kinds of security requirements are identified and informal guidelines are listed that have proven useful when eliciting concrete security requirements. The author emphasises that there has to be a clear distinction between security requirements and security mechanisms.

In [9] it is proposed to use Jackson‘s problem diagrams to determine security requirements which are given as constraints on functional requirements. Though this approach presents a methodology to derive security requirements from security goals, it does not explain the actual refinements process, which leaves open, the degree of coverage of requirements, depending only on expert knowledge.

In [10] actor dependency analysis is used to identify attackers and potential threats in order to identify security requirements. The so called i^* approach facilitates the analysis of security requirements within the social context of relevant actors. In [11] a formal framework is presented for modelling and analysis of security and trust requirements at an organisational level. Both of these approaches target organisational relations among agents rather than functional dependence. Those approaches might be utilised complementary to the presented. Also the output of organisational relations analysis may be an input to our functional security analysis.

3. Motivation

The derivation of security requirements in general, especially the derivation of authenticity requirements represents an essential building block for system design. With an increase in the severity of safety-relevant systems' failures the demand increases for a systematic approach of requirements derivation with a maximum coverage. Also during the derivation of security requirements, no pre-assumptions should be made about possible implementations.

We will further motivate this with respect to the requirements derivation process with an example from the field of vehicle-to-vehicle communications and demonstrate the common mistakes.

3.1. Example use case

For a better illustration of the described problems we will refer to an example, illustrating use case descriptions for a possible vehicle-to-vehicle scenario.

Use case 1. A vehicle's Electronic Stability Protection (ESP) sensor recognises that the ground is very slippery when accelerating in combination with a low temperature. In order to warn successive vehicles about a possibly icy road, the vehicle uses its communication unit (CU) to send out information about this danger including the Global Positioning System (GPS) position data, where the danger was detected.

Use case 2. A vehicle receives a warning about an icy road at a certain position. It compares the information to its own position and heading and signals the driver a warning, if the dangerous area lies up front. Additionally the vehicle will retransmit the warning, given that the position of this occurrence is not too far away.

3.2. Common approaches

There are several possible approaches, that may be taken, depending on the system architect's background.

An architect with a background in Mobile Adhoc Networks (MANETs) would first define the data origin authentication [12] of the transmitted message. In a next step he may reason about the trustworthiness of the transmitting system.

An architect with a background in Trusted Computing [13] would first require for the transmitting vehicle to attest for its behaviour [14]. Advanced experts may use the Trusted Platform Module (TPM) techniques of sealing, binding, key restrictions and TPM-CertifyKey to validate the trustworthiness and bind the transmitted data to this key [15].

A distributed software architect may first start to define the trust zones. This would imply that some computational means of composing slippery wheels with temperature and position happen in an untrusted domain. Results may be the timestamped signing of the sensor data and a composition of these data at the receiving vehicle.

3.3. Problem evaluation

The presented methods shall only illustrate a few different approaches that might be taken in a security engineering process for new systems of systems. Very different types of security requirements are the outcome. Some of these leave attack vectors open, such as the manipulation of the sending or receiving vehicle's internal communication and computation.

Another conclusion that can be derived from these examples is related to premature assumptions about the implementation. Whilst in one case the vehicle is seen as a single computational unit that can be trusted, in another case it has to attest for its behaviour when sending out warnings. The trust zone based analysis of the same use cases however requires for a direct communication link and cryptography between the sensors and the receiving vehicle and the composition of data is moved to the receiver side. A direct result of falsely defined system boundaries typically are security requirements that are formulated against internal subsystems rather than the system at stake itself,

Though all of the approaches may lead to a sufficient level of security for the designed architecture, there is no obvious means by which they can be compared regarding the security requirements that they fulfil. The choice of the appropriate abstraction level and system boundaries constitutes a rather big challenge to systems of systems architecture design, especially with respect to systems of systems applications like the one presented here.

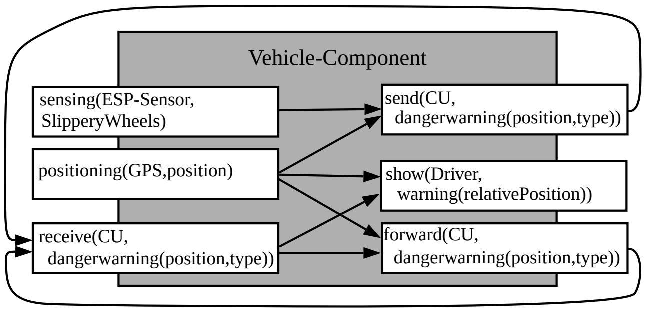
Vehicle-Component
sensing(ESP-Sensor, SlipperyWheels)
positioning(GPS,position)
receive(CU, dangerwarning(position,type))
send(CU, dangerwarning(position,type))
show(Driver, warning(relativePosition))
forward(CU, dangerwarning(position,type))

Fig. 1. Example - functional component model

4. Approach

The approach described in the following can be decomposed into three basic steps. The first one is the derivation of the functional model from the use case descriptions in terms of an action oriented system. In a second step the system at stake is defined and possible instantiations of the first functional model are elaborated. In a third and final step, the actual requirements are derived in a systematic way, resulting in a consistent and complete set of security requirements.

4.1. Functional model

For the description of the functional model from the use cases an action-oriented approach is chosen. The approach is based on the work from [16]. For reasons of simplicity and readability the formal description of the model is omitted here and a graphical representation is used to illustrate the behaviour of the evaluation target.

A functional model can be derived from a use case description by identifying the atomic actions in the use case description. These actions are set into relation by defining the functional flow among them. This action oriented approach considers possible sequences of actions (control flow) and information flow (input/output) between interdependent actions.

In the case of highly distributed systems and especially a distributed system of distributed systems, it is very common that use cases do not cover a complete functional cycle throughout the whole system under investigation. Rather only certain components of the system are described regarding their behaviour. This must be kept in mind when deriving the functional model. In order to clarify this distinction, functional models that describe only parts of the overall system behaviour will be called *functional component model*.

Figure 1 shows a functional component model for a vehicle derived from the example use cases given in section 3.1. The functional flow arrows outside of the vehicle's boundaries refer to functional flows between different instances of the component, whilst internal flow arrows refer to flows within the same instance of the component. For the given example, the external flows represent data transmission of one vehicle to another, whilst the internal flows represent communication within a single vehicle.

4.2. System of systems instances

Based on the functional component model, one may now start to reason about the overall system of systems which consists of a number of instances of the functional components. The synthesis of the internal flow between the actions within the component instances and the external flow between systems (in this case vehicles) builds the global system of systems behaviour. In order to model instances of the global system of systems, all structurally different combinations of component instances shall be considered. Isomorphic combinations can be neglected. Finally, all possible instances may be regrouped and the system's boundary actions (denoting the actions that are triggered by or influence the system environment) have to be identified. These will be the basis for the security requirements definition in the next step.

In Fig. 2 an example for possible instances of the vehicle description in a distributed vehicle-to-vehicle scenario is presented. In this example the forwarding of a message is restricted by a *position based forwarding policy* with respect to the distance from the danger that is being warned about and the time of issue of the danger sensing. We could therefore assume a maximal number of system instances involved general enough to cover all these cases, e.g. by utilising a description in a parameterised way.

4.3. Functional security requirement identification

The set of possible instantiations of the functional component model is used in a next step to derive security requirements. First, the boundary actions of the system model instances are determined. Let the term *boundary action* refer to the actions that form the interaction of the internals of the system with the outside world. These are actions that are either triggered by occurrences outside of the system or actions that involve changes to the outside of the system.

With the boundary actions being identified, one may now follow the functional graph backwards. Beginning with the boundary actions by which the system takes influence on the outside, we may propagate backwards along the functional flow. These backwards references basically describe the functional dependencies of actions among each other. From the functional dependency graph we may now identify the end points - the boundary actions that trigger the system behaviour that depends on them. Between these and the corresponding starting points, the requirement exists that without such an action happening as input to the system, the corresponding output action must not happen as well. From this we formulate the security goal of the system at stake:

Whenever a certain output action happens, the input action that presumably led to it must actually have happened.

This requirement shall now be enriched by additional parameters. In particular, it shall be identified which is the entity that must be assured of the aforementioned requirement. With these additional parameters set, we may utilise the definition of authenticity from the formal framework of Fraunhofer SIT [17] to specify the identified requirements. The syntax to describe these requirements in parameterised form is defined as:

Definition 1: $Authentic(A, B, P)$: Whenever an action B happens, it must be authentic for an Agent P that in any course of events that seem possible to him, a certain action A has happened (for a formal definition we refer the reader to [17]).

It shall be noted that the requirements elicitation process in this case utilises positive formulations of how the system should behave, rather than preventing a certain malicious behaviour. Also it has to be stressed that this approach guarantees for the system / component architect to be free regarding the choice of concepts during the security engineering process.

Once an exhaustive list of security requirements is identified, a requirements categorisation and prioritisation process can evaluate them according to a maximum acceptable risk strategy. This manual analysis may reveal that certain functional dependencies are presented only for performance reasons. This can be valuable input for the architects as well, and sometimes reveals premature decisions about mechanisms that were already done during the use case definition phase.

This approach cannot prevent the specification of circular dependencies among systems' actions but usually this is avoided for well-defined use cases. This actually originates from the fact that every action represents a progress in time. Accordingly an infinite loop among actions in the system would indicate that the system described will not terminate.

The requirements derivation process will however highlight every functional dependency that is described within the use cases. Accordingly, when the use case description incorporates more than the sheer safety related functional description, additional requirements may arise. Therefore, the requirements have to be evaluated towards their meaning for the system's safety. Whilst one can be assured not to have missed any safety relevant requirement, this is a critical task because misjudging a requirement's relevance would induce security holes.

4.4. Formalisation

Formally, the functional flow among actions can be interpreted as an ordering relation ζ_i on the set of actions Σ_i in a certain system instance i. To derive the requirements the reflexive transitive closure ζ_i^* is constructed. In the following we assume that the functional flow graph is sequential and free of loops, as every action can only depend on past actions. Accordingly, the relation is anti-symmetric. ζ_i^* is a partial order on Σ_i, with the maximal elements max_i corresponding to the outgoing boundary actions and the minimal elements min_i corresponding to the incoming boundary actions. After restricting ζ_i^* to these elements
$\chi_i = \{(x, y) \in \Sigma_i\Sigma_i \mid (x, y) \in \zeta_i^* \wedge x \in min_i \wedge y \in max_i\}$
this new relation represents the authenticity requirements for the corresponding system instance: *For all* $x, y \in \Sigma_i$ *with* $(x, y) \in \chi_i$: $auth(x, y, stakeholder(y))$ is a requirement. Accordingly the union of all these requirements for the different instances poses the set of requirements for the whole system. This set can be reduced by eliminating duplicate

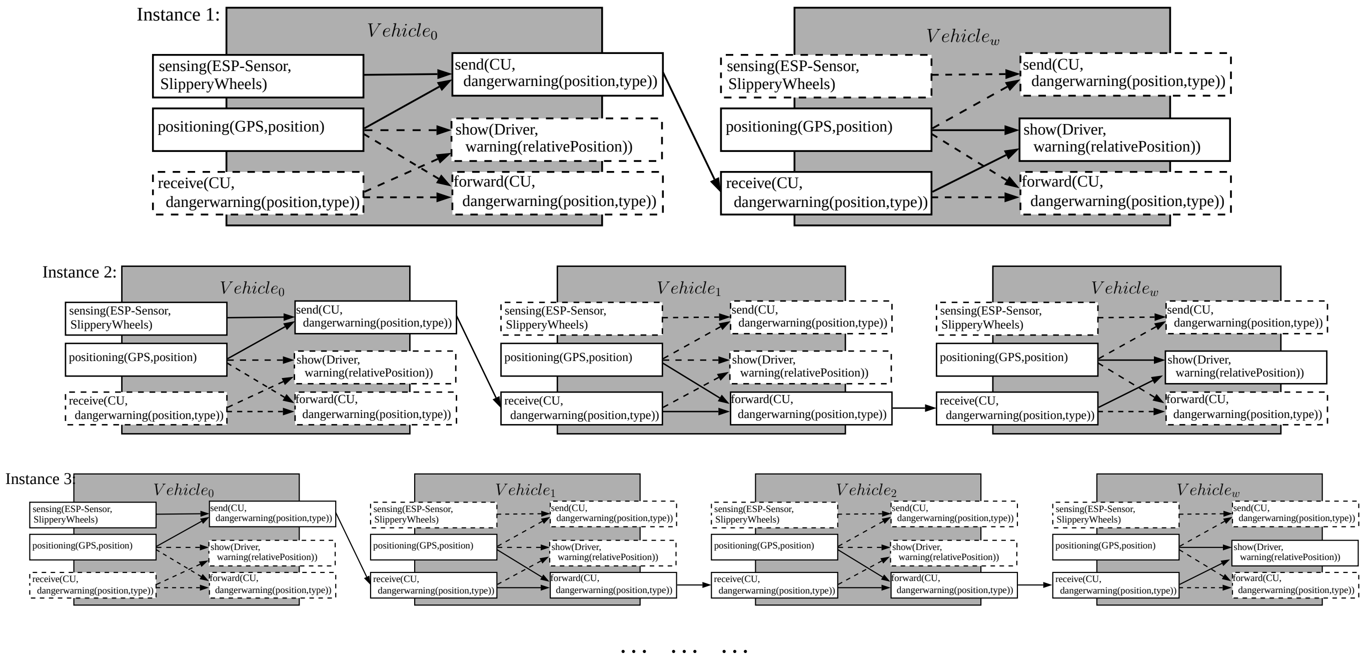
Instance 1:
$Vehicle_0$
sensing(ESP-Sensor, SlipperyWheels)
positioning(GPS,position)
receive(CU, dangerwarning(position,type))
send(CU, dangerwarning(position,type))
show(Driver, warning(relativePosition))
forward(CU, dangerwarning(position,type))
$Vehicle_w$
Instance 2:
$Vehicle_1$
Instance 3:
$Vehicle_2$
… … …

Fig. 2. Example - functional model instances

requirements or by use of first-order predicates for a parameterised notation of similar requirements.

Example derivation of authenticity requirements. For the given system model instances, we may now identify the authenticity requirements for the action $show(V_w, D_w, warn(rP))$ (with V = Vehicle, D = Driver, ESP = ESP-sensor, $warn$ = warning, pos = positioning, $sens$ = sensing, rP = relativePosition, pD = positionData, sW = slipperyWheels, dw = dangerwarning). Graphically, this could be done by reversing the arrows and removing the dotted arrows and boxes. Formally, for the system of systems instance 1 from Fig. 2, we can analyse:

$$\begin{aligned}\zeta_1 = \{&(sens(ESP(V_0), sW), send(CU(V_0), dw)),\\ &(pos(GPS(V_0), pD), send(CU(V_0), dw)),\\ &(send(CU(V_0), dw), rec(CU(V_w), dw)),\\ &(pos(GPS(V_w), pd), show(V_w, D_w, warn(rP))),\\ &(rec(CU(V_w), dw), show(V_w, D_w, warn(rP)))\}\end{aligned}$$

$$\begin{aligned}\zeta_1^* = \zeta_1 \cup \{(x, x) \mid x \in \Sigma\} \cup \{&\\ &(sens(ESP(V_0), sW), rec(CU(V_w), dw)),\\ &(sens(ESP(V_0), sW), show(V_w, D_w, warn(rP))),\\ &(pos(GPS(V_0), pD), rec(CU(V_w), dw)),\\ &(pos(GPS(V_0), pD), show(V_w, D_w, warn(rP)))\}\end{aligned}$$

$$\begin{aligned}\chi_1 = \{&(sens(ESP(V_0), sW), show(V_w, D_w, warn(rP))),\\ &(pos(GPS(V_0), pD), show(V_w, D_w, warn(rP))),\\ &(pos(GPS(V_w), pD), show(V_w, D_w, warn(rP)))\}\end{aligned}$$

An analysis for the second instance will result in:

$$\begin{aligned}\chi_2 = \{&(sens(ESP(V_0), sW), show(V_w, D_w, warn(rP))),\\ &(pos(GPS(V_0), pD), show(V_w, D_w, warn(rP))),\\ &(pos(GPS(V_w), pD), show(V_w, D_w, warn(rP))),\\ &(pos(GPS(V_1), pD), show(V_w, D_w, warn(rP)))\}\end{aligned}$$

And the third system of systems instance will result in:

$$\begin{aligned}\chi_3 = \{&(sens(ESP(V_0), sW), show(V_w, D_w, warn(rP))),\\ &(pos(GPS(V_0), pD), show(V_w, D_w, warn(rP))),\\ &(pos(GPS(V_w), pD), show(V_w, D_w, warn(rP))),\\ &(pos(GPS(V_1), pD), show(V_w, D_w, warn(rP))),\\ &(pos(GPS(V_2), pD), show(V_w, D_w, warn(rP)))\}\end{aligned}$$

The first three elements in each χ_i will obviously always be the same in all instances of the example. The rest of the elements can be expressed in terms of first-order predicates. This leads to the following authenticity requirements for all possible system instances for the action $show(V_w, D_w, warn(rP))$:

1) $auth(\ pos(GPS(V_w), pD),$
$\quad show(V_w, D_w, warn(rP)), D_w\ )$
2) $auth(\ pos(GPS(V_0), pD),$
$\quad show(V_w, D_w, warn(rP)), D_w\ )$
3) $auth(\ sens(ESP(V_0), sW),$
$\quad show(V_w, D_w, warn(rP)), D_w\ )$
4) $\forall\ V_x \in V_{forward} : auth(\ pos(GPS(V_x), pD),$
$\quad show(V_w, D_w, warn(rP)), D_w\ )$

$V_{forward}$ denotes the set of vehicles per system instance, that forward the warning message.

As mentioned above, the resulting requirements have to be evaluated regarding their meaning for the functional safety of the system. For the first three requirements the argumentation is very straight forward regarding why they have to be fulfilled:

1) It must be authentic for the driver that the relative position of the danger he/she is warned about is based on correct position information of his/her vehicle.
2) It must be authentic for the driver that the position of the danger he/she is warned about is based on correct position information of the vehicle issuing the warning.
3) It must be authentic for the driver that the danger he/she is warned about is based on correct sensor data.

The last requirement 4) however must be further evaluated. Studying the use case, we see that this functional dependency originates from the position based forwarding policy. This policy is introduced for performance reasons, such that bandwidth is saved by not flooding the whole network. Braking this requirement would therefore result either in a smaller or in a larger broadcasting area. As bad as those cases may be, they cannot cause the warning of a driver that should not be warned. Therefore we do not consider requirement 4 to be a safety related authenticity requirement. It can be considered a requirement regarding availability by preventing the denial of a service or unintended consumption of bandwidth.

Further steps. Starting from this set of very high-level requirements, the security engineering process may start. This will include decisions regarding the mechanisms to be included. Accordingly the requirements may be refined to more concrete requirements in this process. The design and refinement process may reveal further requirements regarding the internals of the system that have to be addressed as well.

5. Conclusion

The presented approach for deriving safety critical authenticity requirements in systems of systems solves several issues compared to existing approaches. It incorporates a clear scheme that will ensure a consistent and complete set of security requirements. Also it is based directly on the functional analysis, ensuring the safety of the system at stake. The systematic approach that incorporates formal semantics leads directly to the formal validation of security, as it is required by certain evaluation assurance levels of Common Criteria (ISO/IEC 15408). Furthermore the difficulties of designing systems of systems are specifically targeted.

In practice, the method described here has been applied in the project EVITA (E-Safety Vehicle Intrusion Protected Applications) to derive authenticity requirements for the development of a new automotive on-board architecture utilising vehicle-to-vehicle and vehicle-to-infrastructure communication. A total of 29 authenticity requirements have been elicited by means of a system model comprising 38 component boundary actions with 16 system boundary actions comprising 9 maximal elements and 7 minimal elements.

Future work may include the derivation of confidentiality requirements in a similar way as was presented here. Though this will require for different security goals, as confidentiality is not related to safety in a similar way, but rather to privacy. Non-Repudiation may also be a target that should be approached in cooperation with lawyers in order to find the relevant security goals. Furthermore, the refinement throughout the design process should be evaluated regarding possibility of formalising it in schemes with respect to the security requirements refinement process.

Acknowledgement

Part of the work presented in this paper was developed within the project EVITA (http://www.evita-project.org) being co-funded by the European Commission within the Seventh Framework Programme.

References

[1] A. Avizienis, J.-C. Laprie, B. Randell, and C. E. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Trans. Dependable Sec. Comput.*, vol. 1, no. 1, pp. 11–33, 2004.

[2] D. J. Bodeau, "System-of-Systems Security Engineering," in *In Proc. of the 10th Annual Computer Security Applications Conference, Orlando, Florida*. IEEE Computer Society, 1994, pp. 228–235.

[3] P. Papadimitratos, L. Buttyan, J.-P. Hubaux, F. Kargl, A. Kung, and M. Raya, "Architecture for Secure and Private Vehicular Communications," in *IEEE International Conference on ITS Telecommunications (ITST)*, Sophia Antipolis, France, June 2007, pp. 1–6.

[4] D. Mellado, E. Fernández-Medina, and M. Piattini, "A common criteria based security requirements engineering process for the development of secure information systems," *Comput. Stand. Interfaces*, vol. 29, no. 2, pp. 244–253, 2007.

[5] N. R. Mead and E. D. Hough, "Security requirements engineering for software systems: Case studies in support of software engineering education," in *CSEET '06: Proceedings of the 19th Conference on Software Engineering Education & Training*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 149–158.

[6] N. R. Mead, "How To Compare the Security Quality Requirements Engineering (SQUARE) Method with Other Methods ," Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU/SEI-2007-TN-021, 2007.

[7] A. van Lamsweerde, "Elaborating security requirements by construction of intentional anti-models," in *ICSE '04: Proceedings of the 26th International Conference on Software Engineering*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 148–157.

[8] D. Firesmith, "Engineering security requirements," *Journal of Object Technology*, vol. 2, no. 1, pp. 53–68, 2003.

[9] C. B. Haley, R. C. Laney, J. D. Moffett, and B. Nuseibeh, "Security requirements engineering: A framework for representation and analysis," *IEEE Trans. Software Eng.*, vol. 34, no. 1, pp. 133–153, 2008.

[10] L. Liu, E. Yu, and J. Mylopoulos, "Analyzing security requirements as relationships among strategic actors," in *2nd Symposium on Requirements Engineering for Information Security (SREIS'02)*, 2002.

[11] P. Giorgini, F. Massacci, J. Mylopoulos, and N. Zannone, "Requirements engineering meets trust management: Model, methodology, and reasoning," in *In Proc. of iTrust 04, LNCS 2995*. Springer-Verlag, 2004, pp. 176–190.

[12] R. Shirey, "Internet Security Glossary, Version 2," RFC 4949 (Informational), Aug. 2007. [Online]. Available: http://www.ietf.org/rfc/rfc4949.txt

[13] T. C. Group, "TCG TPM Specification 1.2 revision 103," www.trustedcomputing.org, 2006.

[14] R. Sailer, X. Zhang, T. Jaeger, and L. van Doorn, "Design and implementation of a TCG-based integrity measurement architecture," in *Proceedings of the 13th USENIX Security Symposium*. USENIX Association, 2004.

[15] A.-R. Sadeghi and C. Stüble, "Property-based attestation for computing platforms: caring about properties, not mechanisms," in *NSPW '04: Proceedings of the 2004 workshop on New security paradigms*. New York, NY, USA: ACM, 2004, pp. 67–77.

[16] P. Ochsenschläger, J. Repp, and R. Rieke, "Abstraction and composition – a verification method for co-operating systems," *Journal of Experimental and Theoretical Artificial Intelligence*, vol. 12, pp. 447–459, June 2000, copyright: ©2000, American Association for Artificial Intelligence (www.aaai.org). All rights reserved. [Online]. Available: http://sit.sit.fraunhofer.de/smv/publications/download/flairs-2000c.pdf

[17] S. Gürgens, P. Ochsenschläger, and C. Rudolph, "Authenticity and provability - a formal framework," in *Infrastructure Security Conference InfraSec 2002*, ser. Lecture Notes in Computer Science, vol. 2437. Springer Verlag, 2002, pp. 227–245.