A Formal Methodology Applied to Secure Over-the-Air Automotive Applications

Gabriel Pedroza
Telecom ParisTech, LTCI CNRS
Sophia Antipolis, France
pedroza@telecom-paristech.fr

Muhammad Sabir Idrees
EURECOM
Sophia Antipolis, France
idrees@eurecom.fr

Ludovic Apvrille
Telecom ParisTech, LTCI CNRS
Sophia Antipolis, France
ludovic.apvrille@telecom-paristech.fr

Yves Roudier
EURECOM
Sophia Antipolis, France
yves.roudier@eurecom.fr

***Abstract*—The expected high complexity in future automotive applications will require to frequently update electronic devices supporting those applications. Even if in-car devices are trusted, potential attacks on over the air exchanges impose stringent requirements on both safety and security. To address the formal verification of safety properties, we have previously introduced the AVATAR UML profile whose methodology covers requirement, analysis, design, and formal verification stages [1]. We now propose to extend AVATAR to support both safety and security during all methodological stages, and in the same models. The paper applies the extended AVATAR to an over-the-air protocol for trusted firmware updates of in-car control units, with a special focus on design and formal verification stages.**

I. Motivation and Outline

One promising avenue to decrease the number of fatal traffic accidents is to rely on V2X[1] communications [2]. However, adding new in-vehicle services obviously facilitates novel applications, but also imposes stringent requirements on security and safety. Indeed, as explained in [3], attacks on in-vehicle networks may have serious consequences. More precisely, if an attacker could install malicious firmware into a vehicle, then he might virtually control and perform arbitrary actions on the on-board system [4]. A way for an attacker to install such a firmware is to attack the Over-the-Air (OTA) diagnosis and firmware update procedure. Finally, V2X applications - including remote flashing procedures - must be developed with a high safety and security assurance level.

Several methodologies have already been introduced to ensure a given level of security in embedded systems. For example, Common Criteria include a formal certification methodology that is unfortunately costly, paperwork oriented, and that neither addresses safety issues nor attack protection. Many other methodologies cover the development of critical embedded systems, with a formal verification stage unfortunately only focused on safety properties, e.g., [5]. Formal methodologies could directly be used, but many of them mainly target verification of a subpart of the system - e.g., they target only the communication parts, i.e. communication protocols [6] - while letting the rest almost uncovered. Also, verification methodologies targeting the entire system are often quite complex and demand specialized knowledge [7]. Furthermore, approaches with partial or no automation support may not be adequate for non-experimented users [8]. Our approach extends the safety-oriented environment AVATAR with security constructs and verification techniques. AVATAR is a Unified Modeling Language (UML) profile targeting real-time embedded systems. AVATAR covers all usual methodological stages (*requirement capture, system analysis, system design, property modeling,* and *formal verification*) [1]. Further, AVATAR is fully supported by TTool, an open-source toolkit offering a friendly front-end for modeling and automated property verification [9]. In the scope of the EVITA project [2], we apply this novel approach to formally secure an automotive safety critical application. We target the verification of the Firmware Updates (FU) protocol described in [10] which was recently proposed as a V2X solution for remote and wired firmware updates. Several approaches have already been used to specifically verify firmware updates specifications [11], [12], [13]. They nevertheless take questionable assumptions in protocol verification such as weak access control and one-way authentication (no vehicle authentication). Our approach partially tackles these identified issues.

This paper is structured as follows. First, we present the underlying in-car security architecture in Section II. Then, Section III depicts how we extend AVATAR with security in mind. The formal verification process is illustrated with the FU protocol in Section IV. Verification results are then presented in Section V. Section VI finally concludes the paper.

II. Security Architecture and Vulnerabilities

Recent on-board Intelligent Transport (IT) architectures comprise a very heterogeneous landscape of communication network technologies (e.g., LIN, CAN, MOST, and FlexRay) that interconnect in-car Electronic Control Units (ECUs) [14]. The increasing number of such equipments triggers the development of novel applications, and reciprocally but it also introduces more complex requirements on safety and security. Based on security requirements identified in [3], we introduced a security architecture [15] that covers cross-layer security in ECUs, targeting platform integrity, com-

[1]V2X stands for any external vehicular communications such as Vehicle-to-Vehicle (V2V) or Vehicle-to-Infrastructure (V2I) communications.

978-1-4244-8325-9/11/$26.00 ©2011 IEEE

munication channels, access control and intrusion detection, by specifying software and hardware security mechanisms. This security architecture is further enhanced with a set of cryptographic protocols [16]. The combination of the architecture and the protocols shall enforce the security requirements identified for the respective use cases [14]. However, as stated in [17], the deployment of security protocols may introduce vulnerabilities such as *design vulnerabilities* (e.g., security flaws in protocol design), *implementation vulnerabilities* (protocol implementation vulnerable to attacks), *configuration vulnerabilities* (improper configuration of components), and *system vulnerabilities* (host platform vulnerable to attacks). Thus, the EVITA in-depth formal analysis takes into account both security-related elements of the architectures and cryptographic protocols. However, this paper focuses on the verification of the cryptographic protocols against configuration and system vulnerabilities (see Section III-B).

III. Formal Verification Methodology

A. The AVATAR Profile

1) AVATAR: AVATAR is a SysML environment [1] with a 5-stage methodology: *requirement capture, system analysis, system design, property modeling* and *formal verification*. All those stages are supported with SysML diagrams. An AVATAR design is made upon a SysML Block representation. A SysML Block is defined by a list of *attributes*, a list of *methods*, a list of *synchronous and asynchronous signals* and a SysML *State Machine Diagram* (SMD). A SMD includes a set of states linked with transitions: the latter are labeled with guards, actions on variables and method calls. SMDs (see figure 3) also support signal sending and receiving operators - respectively denoted by *chanOut()*, *chanIn()* - and deterministic and non deterministic temporal operators (e.g., timers).
TTool is an open-source toolkit [9] that supports several UML profiles, e.g., TURTLE [18], DIPLODOCUS [19], and AVATAR. In TTool, an AVATAR design can be checked against safety properties at the push of a button: the underlying formal language and toolkit (UPPAAL [20]) are indeed totally hidden to the user.

2) Extending AVATAR for security purposes: All methodological stages of AVATAR have been extended as follows: **safety and security issues shall be modeled all together on the same model**, and **safety and security properties shall be provable from the same model**. For example, the requirement stage now supports security-oriented requirements and attack trees [21]. More precisely, the design and verification stages are extended as follows:

a. ***Design:*** The design supports cryptographic-oriented data types (e.g., keys) and SysML crypto Blocks defining cryptographic-oriented methods (e.g., $encrypt()$, $mac()$). Communication channels between Blocks can now be declared as public or private. Only public ones can be listened to by an attacker. Security oriented modeling pragmas can now be defined in SysML text notes. The pragma **#InitialCommonKnowledge** $\mathbf{B1.x\ B2.y}$ represents the fact that the attributes x and y of Blocks $B1$ and $B2$ respectively have the same initial value. This pragma is useful for modeling preshared keys, for example.
b. ***Security Properties:*** Security properties to be verified are also defined as pragmas. The **#Confidentiality** $\mathbf{B.dat}$ pragma shall verify whether the attribute dat of Block B remains confidential. The pragma **#Authenticity** $\mathbf{B1.st1.msg1\ B2.st2.msg2}$ states that all messages $msg2$ received by $B2$ in state $st2$ must be authentic with respect to $B1$, i.e., they must have been sent in $msg1$ by $B1$ after state $st1$.
c. ***Formal Verification:*** Extended AVATAR diagrams can be formally verified against security properties, while being still amenable to safety proof. For security proofs, we have defined an *AVATAR-to-ProVerif()* translation process and have implemented that process in TTool. ProVerif is a toolkit for automatically proving confidentiality and authenticity properties [22].

Finally, from an AVATAR design, confidentiality and authenticity properties can now be modeled and proved at the push of a button.

B. Target of Verification

Our AVATAR-based verification methodology relies upon a Target of Verification: $ToV := (S, SR, M, P, H, At, VT)$ with:

- $\mathbf{S}$ is a set of representative - but non exhaustive - behaviors of the system assets. S is represented with UML Use Case, Interaction Overview, and Sequence Diagrams.
- $\mathbf{SR}$ is a set of Security or safety Requirements represented by mean of SysML Security Requirements Diagrams [3] which provide a hierarchical structure for refinement and association to assets.
- $\mathbf{M}$ is an AVATAR Model abstracting the system specified in S.
- $\mathbf{P}$ represents properties derived from SR. P is represented in AVATAR using confidentiality and authenticity pragmas (see Subsection III-A).
- $\mathbf{H}$ is a set of hypotheses and abstractions taken into account during the modeling steps $S \rightarrow M$ and $SR \rightarrow P$. H also includes model constraints required for verification, e.g., to state that a given key can not be guessed.
- $\mathbf{At}$ is a model of attackers. Security properties in P are verified with respect to an attacker At who challenges M. AVATAR relies upon the Dolev-Yao attacker model of ProVerif [22]: the attacker can listen, intercept, alter, and inject messages over public channels.

- **VT** is a Verification Technique to prove whether properties of P are satisfied by M. For safety properties, VT depends on the approach described in [1]: AVATAR handles the proof of liveness, reachability, and deadlock-freedom properties. For the verification of security properties, VT is the resolution algorithm implemented in ProVerif [22].

At first, *requirement* and *analysis* stages produce SR and S, respectively. Then, M is built at *system design* stage. *Property modeling* stage defines P. The hypotheses H are directly or implicitly taken as input in several stages. The last two ToV components - At & VT -, are implicitly handled by the *TTool-to-Proverif()* automatic translation function, and by ProVerif itself, during the formal verification stage.

IV. Verification of the Firmware Updates Protocol

The formal verification methodology described in previous Section is now applied to the FU protocol [10].

A. Protocol Analysis (S)

To simplify the modeling and description of S, the FU Protocol is split into two sequential phases named *Diagnosis* and *Download*, respectively. Even if the paper describes only the *Diagnosis* phase, both phases were modeled and verified. However, more details on the *Download* phase and FU Protocol can be found in [10].
The *Diagnosis* phase consists in the initial exchanges between an external Diagnosis Tool (*DT*), the car Communications Control Unit (*CCU*) and the target in-car *ECU* that requires the firmware update. Initially, *DT* establishes a secure channel to efficiently communicate with the *ECU*. A symmetric key Mk is thus generated by *DT* and transferred to *ECU* (see figure 1, Messages 1 to 4). Once Mk is distributed, subsequent exchanges between *DT* and the target *ECU* are MAC protected with Mk. To be ready for flashing, *ECU* should be turned to a programming mode (see figure 1, Messages 5 to 8). To be unlocked, the target *ECU* challenges *DT* to compute a code Smk relying on a seed Na - provided by *ECU* - and a factory preshared symmetric key ssk ($Smk := ComputeKey(Na, ssk)$). The *ECU* unlocks for programming only if both Smk codes match. Then, the firmware is securely transferred from the Original Equipment Manufacturer (*OEM*) server to the target *ECU* through *DT* (*Download* phase).

B. Security Requirements (SR)

As a result of risk and threat analyses [3], a set of stringent security requirements must be satisfied by the FU protocol. Here are a subset of the most critical ones taken from SR. More detailed security requirements can be found in [21].

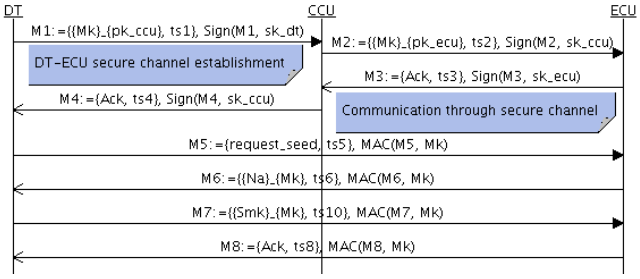

Figure 1. Sequence diagram of the *Diagnosis* Phase

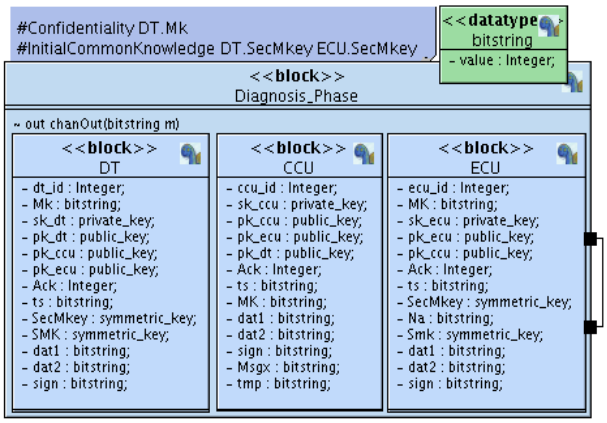

Figure 2. AVATAR Block diagram for *Diagnosis* phase in FU Protocol

- *Firmware confidentiality (CSR.1):* Firmware must be kept confidential when transferred from OEM to DT and from DT to the target ECU.
- *Key confidentiality (CSR.2):* Exchanges transferring or using non-public keys must preserve key secrecy along the whole flashing process.
- *Internal Authenticity (ASR.1):* Whenever an exchange between CCU and ECU happens, the correspondence between claimed and real authors must be ensured.
- *External Authenticity (ASR.2):* Whenever data is exchanged between DT utility and OEM server or between DT and in-car components, the correspondence between claimed and real authors must be ensured.

C. Target of Verification Model (M)

The protocol model M is built taking S as reference. An AVATAR Block is used for each Communicating Entity (CE) in S - *DT*, *CCU*, *ECU*, and *OEM* (see figure 2). For this case study, we only describe the modeling of the first exchange in the FU Protocol. Other exchanges are captured in a similar way. As shown in figure 1, that first exchange comprises an encrypted key ($\{Mk\}\_\{pk\_ccu\}$), a time stamp ($ts1$) and $M1$'s signature performed with *DT*'s secret key ($Sign(M1, sk\_dt)$). The exchange must be represented in the State Machine Diagrams (SMD) of both sending and receiving Blocks. First, an initial state is defined in the

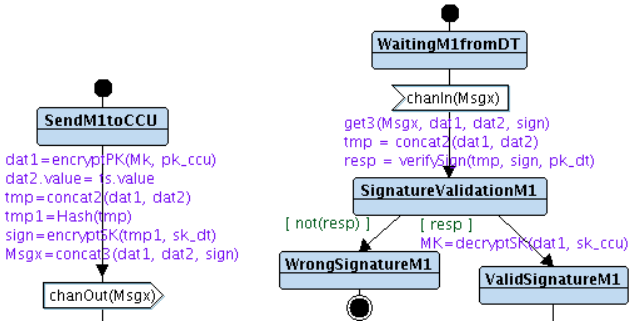

Figure 3. First message exchange between *DT* and *CCU* in the FU Protocol

SMD of *DT* (see figure 3). Second, several predefined crypto methods are called to build $M1$ as an outgoing transition of the initial state. Indeed, $dat1$, $dat2$, and $sign$ are assigned to attribute $Msgx$, which captures $\{Mk\}\_\{pk\_ccu\}$, $ts1$, and $Sign(M1, sk\_dt)$, respectively. Once composed, $Msgx$ truly corresponds to $M1$ and is broadcasted via the public output signal operator $chanOut(Msgx)$. Therefore, the SMD of *CCU* is accordingly designed to receive $M1$. Thus, an outgoing transition from the initial state in the SMD of *CCU* includes an input signal operator $chanIn(Msgx)$ that waits for $M1$ (see figure 3). After receiving $M1$, the predefined method $verifySign()$ is called to verify the signature of $M1$. The boolean attribute $resp$ retrieves the answer: a wrong signature ($not(resp) == true$) leads to a blocking state. Conversely, an authentic one ($resp == true$) shall lead to valid states in the protocol. This modeling pattern for message exchange is repeated for all other Protocol Data Units.

D. Properties (P)

Each security requirement in SR is translated to one or more properties in P. Properties are formalized in the AVATAR model M by relying upon *property pragmas* as defined in Section III-A. For instance, the Security Requirement $CSR.1$ (see Subsection IV-B) is translated to the pragma described on the first row of Table I. This pragma models the confidentiality of the attribute $firmware$. $CSR.2$ is translated as well into several analogous pragmas thus modeling the confidentiality of the following secret Block attributes: *sk_dt, sk_ccu, sk_ecu, sk_oem, Mk, Na, Smk* and *ssk*. Along with that, $ASR.1$ expresses a mutual authenticity whenever an exchange happens between the in-car components *CCU* and *ECU* (*CCU*↔*ECU*). In addition, and in compliance with $ASR.2$, mutual authentication is also required with external car Communicating Entities (CE) and for every exchange (*DT*↔*CCU*, *DT*↔*ECU* and *OEM*↔*DT*). The second and third rows of Table I contain pragmas depicting how to achieve the mutual authentication of *DT* and *CCU* for message $M1$ and $M4$, respectively (see figure 1, $M1$ and $M4$).

Table I
CONFIDENTIALITY AND AUTHENTICITY PRAGMAS

Pragma	Semantics
#Confidentiality OEM.firmware	Confidentiality of $firmware$ defined in OEM Block
#Authenticity DT.Send.M1 CCU.ValidSign.M1	Authenticity of $M1$
#Authenticity CCU.Send.M4 DT.ValidSign.M4	Authenticity of $M4$

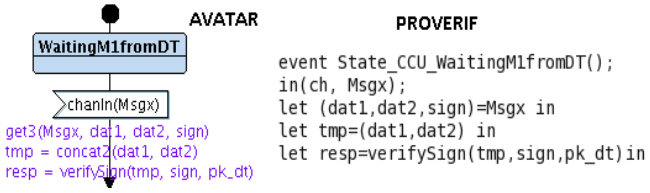

Figure 4. Five elements in an AVATAR-ProVerif translation

E. Hypotheses (H)

Assumptions on values preshared by Blocks prior to message exchanges - e.g., a key or seed - are explicitly considered within **#InitialCommonKnowledge** pragmas (see Subsection III-A). Assumptions on the fact that values are secret when the system starts are implicitly declared within the **#Confidentiality** pragmas. According to Section II, we assume that each CE Block is a trusted domain inside of which operations are carried out securely. In contrast, communication channels between CE Blocks are assumed public, and thus possibly attacked. Even if cryptographic methods declared in CE Blocks can be known by the attacker At, non invertible ones - like Hash, MAC, or Signature - prevent At for disclosing data that are taken as input by the crypto method. Indeed, secret material is never revealed by At unless the system model M has exploitable weaknesses. Thus, At can not guess secret key values. Note that the ToV approach does not consider computational attacks.

V. VERIFICATION AND RESULTS

The FU Protocol ToV can be verified at the push of a button and thus no specialized knowledge is required. Here, we briefly describe this automated verification process.
In a first step, the model M, the properties P and hypotheses H - taken into account within M - are automatically translated to a ProVerif specification, i.e. into a set of Horn clauses. Figure 4 shows the translation to ProVerif of a basic SMD transition. Furthermore, ProVerif includes an attacker At described with a set of Horn clauses $\{h \Rightarrow c\}$ ($h$ypotheses imply $c$onclusions) thus reflecting the knowledge the attacker may gain. For each clause $c_p \Rightarrow h_p$ associated to a pragma, the algorithm formally proves that whenever the facts c_p are reached, the facts in h_p are necessarily implied:

- ***Proof of confidentiality:*** For a pragma **#Confidentiality** $\mathbf{B.dat}$, it is proved whether At can derive a chained sequence of Horn clauses $h_1 \Rightarrow c_1, \ldots, h_n \Rightarrow c_n$, $c_i \subset h_{i+1}$, in which h_1 is a subset of its gained knowledge and dat is in c_n, what discloses the secret dat.
- ***Proof of authenticity:*** For each pragma **#Authenticity** $\mathbf{B_1.State_1.dat_1}$ $\mathbf{B_2.State_2.dat_2}$, it is proved whether the attacker At can partially or completely impersonate B_1. In such a case, B_2 accepts at least one message made by At thus breaking the correspondence (*injective agreement*, [22]) between message origin (B_1) and destination (B_2). Therefore, authenticity is proved only if sender-receiver correspondence is preserved. Since B_1 and B_2 may interchange their roles (B_2 origin, B_1 destination), an analogous pragma can be accordingly written to achieve mutual verification of authenticity.

Finally, At's behavior is obviously defined according to allowed operations - e.g., listen/inject only for public channels - as well as by restrictions imposed by statements in H - e.g., a given key is not guessable. Verification results $R(ToV)$ for the Firmware Updates Protocol are shown in Table II as a traceability matrix, with establishes a correspondence between Security Requirements *SR*, Properties *P*, and the verification result. Note that '*X.o*' and '*Y.d*' are used to respectively shorthen origin and destination statements in authenticity pragmas.

Table II
RESULTS FOR VERIFICATION OF THE FU PROTOCOL

Requirement *SR*	Property *P*	Result
CSR.1 Firmware confidentiality	**#Confidentiality** $firmware$	Satisfied
CSR.2 Key confidentiality	**#Confidentiality** *sk_dt, sk_ccu, sk_ecu, sk_oem, Mk, Na, Smk, ssk*	Satisfied
ASR.1 Internal Authenticity	**#Authenticity** *CCU.o ECU.d*	Satisfied
	#Authenticity *ECU.o CCU.d*	Satisfied
ASR.2 External Authenticity	**#Authenticity** *DT.o CCU.d*	Satisfied
	#Authenticity *CCU.o DT.d*	Satisfied
	#Authenticity *DT.o ECU.d*	Satisfied
	#Authenticity *ECU.o DT.d*	Satisfied
	#Authenticity *DT.o OEM.d*	Satisfied
	#Authenticity *OEM.o DT.d*	Satisfied

VI. CONCLUSIONS

The gain in complexity of in-vehicle SW/HW architectures pushes current methodologies for developing such systems to their limit. Even if safety properties have been explicitly considered for a long time, methodologies for handling both safety and security properties at the same time are still to be defined. Relying on a well-known modeling language (SysML), AVATAR is an integrated framework suitable for designers. The automated formal verification reduces the need for skills in formal techniques.

The new AVATAR security capabilities have been successfully applied to the modeling and formal verification of a Firmware Updates Protocol. Indeed, strong mutual authentication as well as confidentiality were verified. Other protocols and in-vehicle system components were also verified in the scope of the EVITA project, thus demonstrating the applicability of AVATAR.
AVATAR shall now be extended with new security capabilities, in particular integrity and freshness. AVATAR has also been selected for modeling and proving both safety and security properties in next generation aeronautics platforms, and is thus expected to further evolve in the next years.

ACKNOWLEDGEMENT

This work has been carried out in the EVITA (E-safety Vehicle Intrusion proTected Applications) project, funded by the European Commission.

REFERENCES

[1] D. Knorreck *et al.*, "TEPE: A SysML language for timed-constrained property modeling an formal verification," in *Proceedings of the UML&Formal Methods Workshop (UML&FM)*, November 2010.
[2] "The EVITA european project," http://www.evita-project.org/.
[3] A. Ruddle *et al.*, "Security Requirements for Automotive On-Board Networks based on Dark-side Scenarios," EVITA Project, Tech. Rep. Deliverable D2.3.
[4] D. K. Nilsson *et al.*, "Key management and secure software updates in wireless process control environments," *WiSec 08*, 2008.
[5] I. Ober *et al.*, "Unambiguous UML composite structures: The OMEGA2 experience," in *SOFSEM 2011: Theory and Practice of Computer Science*, ser. LNCS, 2011.
[6] Y. Chevalier *et al.*, "A high level protocol specification language for industrial security-sensitive protocols," in *Workshop on Specification and Automated Processing of Security Requirements (SAPS 2004)*, 2004.
[7] P. Ochsenschläger *et al.*, "The SH-Verification Tool- A tutorial," SIT- Institute for Secure Telecooperation, Tech. Rep., 1999-2000.
[8] Y. Ali *et al.*, "A rigorous methodology for security architecture modeling and verification," in *Proceedings of the 42nd Hawaii International Conference on System Sciences*. IEEE, 2009.
[9] "TTool in," http://ttool.telecom-paristech.fr.
[10] M. Idrees *et al.*, "Secure automotive on-board protocols: A case of over-the-air firmware updates," ser. 3rd International Workshop on Communication Technologies for Vehicles, LNCS, Ed., March 2011.
[11] T. Miehling *et al.*, "HIS flashloader specification version 1.1," HIS Consortium, Tech. Rep., 2006.
[12] D. Nilsson *et al.*, "Secure Firmware Updates Over the Air in Intelligent Vehicles," in *Proc. ICC Workshops*, 2008.
[13] ——, "A Framework for Self-Verification of Firmware Updates Over the Air in Vehicle ECUs," in *GLOBECOM Workshops 08*.
[14] E. Kelling *et al.*, "Specification and Evaluation of E-security Relevant Use Cases," EVITA Project, Tech. Rep. Deliverable D2.1, 2009.
[15] B. Weyl *et al.*, "Secure On-board Architecture Specification," EVITA Project, Tech. Rep. Deliverable D3.2, 2010.
[16] H. Schweppe *et al.*, "Secure On-Board Protocols Specification," EVITA Project, Tech. Rep. Deliverable D3.3, 2010.
[17] S. N. Foley *et al.*, "Security protocol deployment risk," IWSP 08, 16th International Workshop on Security Protocols, April 2008.
[18] L. Apvrille *et al.*, "TURTLE: A real-time UML profile supported by a formal validation toolkit," in *IEEE Transactions on Software Engineering*, vol. 30. IEEE Computer Society, 2004, pp. 473–487.
[19] ——, "A UML-based environment for system design space exploration," in *13th IEEE International Conference on Electronics, Circuits and Systems (ICECS'2006)*, Nice, France, Dec 2006.
[20] J. Bengtsson *et al.*, "Timed automata: Semantics, algorithms and tools," in *Lecture Notes on Concurrency and Petri Nets*. W. Reisig and G. Rozenberg (eds.), LNCS 3098, Springer-Verlag, 2004.
[21] M. Idrees *et al.*, "A Framework Towards the Efficient Identification and Modeling of Security Requirements," *5th Conf. on Network Architectures and Information Systems Security*, 2010.
[22] B. Blanchet, "Automatic verification of correspondences for security protocols," *Journal of Computer Security*, vol. 17, no. 4, pp. 363–434, JUL 2009.