Politecnico di Milano TR-2012-09

Secure Integration of Mobile Devices for Automotive Services

Roman Kochanek[1,2], Andrea Dardanelli[2], Federico Maggi[2]
Stefano Zanero[2], Mara Tanelli[2], Sergio Savaresi[2], and Thorsten Holz[1]

[1] Horst Görtz Institute for IT-Security, Ruhr-University Bochum
[2] Dipartimento di Elettronica e Informazione (DEI), Politecnico di Milano

Abstract. Modern vehicles, and in particular electric vehicles, are increasingly being equipped with interconnected computer systems, which collect information through vehicular sources and remote, Internet-connected services. Unfortunately, this creates a non-negligible attack surface, which extends even more when vehicles are integrated with smartphones to offer advanced services. In fact, embedded systems on vehicles have been developed to address safety, not security requirements. Furthermore, vehicles have real-time constraints, and the typical embedded architectures used on board significantly complicate security designs.
In this paper, we introduce a communication framework that addresses these challenges and we demonstrate how a smartphone can interact with a vehicle in a secure and safe manner. To this end, we design a security session layer that ensures end-to-end security transparently. We conduct an experimental evaluation on a real implementation of our security layer, which shows that our solution is practical and easy to use, satisfies performance constraints, and meets real-time requirements by taking into account the limited capabilities of our target architecture. More precisely, we implement our approach for an electrically-powered two-wheeler manufactured by Piaggio, and show how a smartphone can interact via a wireless link with the battery-life controller in a secure manner. Interestingly, our approach is not limited to vehicles, but can be used in other application domains where a smartphone needs to securely interact with an embedded device.

1 Introduction

We are moving to the often cited *Internet of Things* where (embedded) devices and sensors regularly exchange data locally with each other and remotely via the Internet, to assist and support humans. An important aspect is the ongoing transition to *smarter* devices that are more energy efficient. Two notorious examples are the concept of "smart grid" and of electric mobility (typically abbreviated as *e-mobility*). E-mobility, in particular, has gained a lot of traction recently and there is a clear trend to produce electrically-powered vehicles. Although this is mainly prompted by the limited availability of fuel, it also enables to develop smarter vehicles that provide additional functionality for a user. Such vehicles are becoming an integral part of the interconnected world, which is for example illustrated by the integration of social networks within the infotainment system of modern cars [1, 2] and the research efforts in the area of so called car-to-X systems [3–5], which describe the communication between a vehicle and its environment (e.g., car-to-car, car-to-infrastructure, car-to-grid, or car-to-mobile systems).

Modern vehicles collect information (e.g., current traffic volume, tire pressure, or power consumption) through different sources, in order to improve the usage of the vehicle. On the downside, such devices (e.g., modern infotainment systems and GSM connections directly embedded in a vehicle) and sensors also lead to an increased attack surface that may enable an adversary to gain remote control of vehicles. In several recent case studies, different research groups highlighted this aspect and successfully demonstrated attacks against different cars [6–10]. In each case study, it was possible to control certain parts of a car and interfere with safety critical or otherwise sensitive components. These vulnerabilities hamper novel use cases such as the usage of smartphones to unlock the vehicle's door or to start the engine, mainly due to the fear of successful attacks against such systems. Adding security mechanisms to vehicles is a challenging task, because vehicles are commonly designed with safety requirements as opposed to security requirements. Recently, however, security requirements are gaining more and more traction. Further challenges arise as vehicles have typically real-time constraints, use broadcast networks often based on controller area network (CAN), embedded devices, and have some other characteristics that complicate security as discussed in §2.

In this paper, we focus on a security solution to protect against potential attacks and introduce a communication framework that addresses the challenges raised above. We demonstrate how a smartphone can interact with a vehicle in a secure and safe manner. More specifically, we discuss how a smartphone can pair with a vehicle over a Bluetooth connection and then establish a *security session layer* that provides additional security guarantees—regardless of the security mechanisms already implemented in the physical layer (if any). We implement an asymmetric key-establishment scheme according to the Elliptic Curve Diffie-Hellman (ECDH) protocol (NIST 800-56A [11]) and a standardized ECC-192 curve (NIST P-192 [12]), For the data encryption, we use a symmetric encryption scheme (AES-128 [13]) based on a long-term shared secret. As a result of our approach, the entire application layer is transparently secured by our security extension. We have designed our solution with performance constraints and real-time requirements in mind. Furthermore, we also took the capabilities of our target architecture into account (e.g., no input capabilities on the vehicle side, limited output capabilities, and lack of trusted execution environment on the mobile-device side). The design of our solution ensures that all these challenges are overcome.

We have implemented our approach for an electric powered two-wheeler (PTW) manufactured by Piaggio and show how the mobile device (an iPhone 4, in our proof-of-concept implementation) can interact securely with the in-vehicle battery-life controller. We performed a brief user study, which indicates that the solution is practical and easy to use. More importantly, our experimental measurements show that the overhead introduced by our security layer is small and reasonable. Interestingly, our approach is not limited to vehicles, but can be used in many other application domains where a smartphone needs to securely interact with an embedded device (e.g., keyless door opening or mobile payment).

In summary, we make the following three key contributions in this paper:

- We introduce a security framework for communication between a mobile device and a vehicle that is both theoretically sound and practical: we have implemented both a software simulator and a real implementation for an electric powered two-wheeler.

– The proposed framework is easy to use and simple, the user is not required to make complex interactions to pair a device with the vehicle in a secure manner. In particular, we conducted hands-on tests with actual users (i.e., non-security people) and collected feedback from a large Italian motorbike manufacturer. Both user studies found that the solution is easy to use and the manufacturer was convinced by the simplicity of our design.
– Performance tests suggest that our implementation has a small (almost absent) computational overhead because it leverages a digital signal processor (DSP) to maximize the cryptographic computations.

2 Security Needs in Modern Automotive Services

Today's automotive market offers a broad range of services [1, 2, 14] that aim to improve drivers' and passengers' safety, enhance their comfort (e.g., infotainment), or provide useful remote services. Typically, such services comprise at least three sets of entities. First, there are several different *electronic control units* (ECUs), sensors and actuators on board a vehicle, usually interconnected through an in-vehicle network. Second, there is at least one *radio interface*, which opens the in-vehicle network towards the outside world, and thus turns the vehicle(s) into full-fledged, Internet-connected devices that implement a truly distributed system. Third, there is at least one external entity (called "service user" from hereinafter), which works closely with the vehicle's ECUs via the radio interface (e.g., a web service used by the vehicle's local services, or a mobile device used for some sort of computation).

In this way, as discussed throughout this paper, the vehicle exposes an interface to the outside that may affect the processing that happens inside it. In particular, the reliability and availability of the vehicular systems may depend on the integrity and safety of the externally supplied data. Therefore, security mechanisms are of paramount importance.

2.1 Attacker Objectives and Model

In the aforementioned communication and execution environment, we assume that an adversary is able to transmit and receive arbitrary data packets on the radio interface, armed with the sole knowledge of the radio protocol in use [6]. Under this assumption, which is perfectly reasonable and realistic, we develop the attacker model summarized in Table 1. The attacker's objectives may be very different, and they may shift, because they are driven by the underlying economic motivations. For instance, due to the increasing capabilities of ECUs (e.g., recent infotainment system designs are based on mobile hardware architectures [15]) and their enhanced connectivity with the next generation of cellular networks, an attacker objective moves from stealing a vehicle towards using the capacity for his or her criminal ecosystem like already seen on mobile platforms [16–18].

Our attacker model encompasses these characteristics and outlines the security threats for a generic automotive-based service. This provides the background for a security assessment of our proposed security framework in §3. As an example, an attacker motivated to steal a vehicle, would need to accomplish **Obj. 1–5**. On the other hand, an attacker who wishes to transform the ECU—and hence the vehicle—into a member of a so-called botnet would need to execute persistently malicious code on an ECU—**Obj. 1–3**.

Table 1. The attacker's objectives in an automotive system architecture.

NAME	DESCRIPTION	IMPACT
Obj. 1	Disclosure or interruption of the security mechanisms integrated in the vehicle's radio protocol	Unauthorized communication path towards an ECU
Obj. 2	Compromising the software implementation of the vehicle's radio interface or protocol	Unauthorized communication path towards an ECU
Obj. 3	Manipulate the execution flow of an ECU	Execution of arbitrary code
Obj. 4	Transmission of specific network packets towards the in-vehicle network	Manipulation of vehicle settings
Obj. 5	Recovery of any security information from a service user	Impersonation of an authorized entity

2.2 In-vehicle Network and CAN Bus

In this work we consider vehicles equipped with *Controller Area Network* (CAN) buses according to ISO11898 [19]. These CANs are highly resilient to external disturbances, and thus are suitable for high-speed distributed applications, which can exchange data (up to 1 Mbit/s) between ECUs positioned in different locations on the vehicle.

Inside the vehicle's network, each ECU connected to the CAN bus is identified with 11 (standard) or 29 (extended) bits. Each ECU can listen, transmit, and receive messages—called *data frames*—on the bus. A data frame is characterized by 8 bytes for the data plus the identifier and several bits for error detection and fault confinement. Typically, ECUs are organized in sub-networks depending on their functionalities and needed speed (e.g., braking system, engine system, or infotainment). The CAN can be viewed as a three-layers protocol: the *object layer* and *transfer layer* are responsible for the post-processing of the message (e.g., synchronization, arbitration, error detection and message filtering) and the *physical layer* handles the electrical issues on the bus.

Clearly, CANs' security requirements differs from the security requirements of the highly-interconnected scenarios typical of modern in-vehicle services. Unsurprisingly, CANs lack confidentiality, integrity, availability, authenticity, and non-repudiation mechanisms, as discussed in [20] and other works such as, for instance, [6–10]. As a matter of fact, CANs are a closed and proprietary system that cannot be modified to secure automotive services. Therefore, in this work, we concentrate on the security problems that arise when the CAN is connected to external devices, which are more significant—especially in today's automotive services—than the security problems that exist within the CAN itself (e.g., ECUs that send unauthenticated data).

2.3 Wireless Connectivity via Bluetooth

Typically, connectivity to the outside world is implemented with the help of a radio interface module connected to the in-vehicle network; more precisely, a special ECU that we call in the following "Gateway ECU'. This ECU acts as a gateway between the internal network and the external world. In our work, we consider the Bluetooth standard as the wireless communication protocol, but the presented concept can be applied to other communication protocols as well.

The Bluetooth protocol has a two-phase session setup: after the so-called *pairing process*, which allows the peers get to know each other and set up the network properties,

the actual *communication* between the peers is enabled. During the pairing process, different security features can be applied for a secure network session depending on the Bluetooth version supported by the peers. For instance, the owner of each device must check that the information displayed on each peer (e.g., a random number) is consistent, or has to choose a (static) personal identification number (PIN), usually propagated out of band. Most of the current Bluetooth authentication schemes are driven by a human-based processing. The first Bluetooth standard also includes the possibility to agree on using no security features before starting a communication session—not a recommended setting as it opens a broad range of potential attacks. The early Bluetooth standard suffered from further security threats due to weak cryptographic primitives, as discussed in [21, 22]. Fortunately, Bluetooth v2.1 enforces the *secure simple pairing* (SSP) protocol [23], which mitigates these security threats and takes into account the constrained resources as well as I/O capabilities of Bluetooth devices. The SSP provides confidentiality and authenticity—unidirectional or mutual—for all peers in a wireless personal-area network. Nevertheless, the SSP protocol still suffers from similar security threats such as the previous Bluetooth security mechanisms (see, e.g., [24, 25]). Unfortunately, most Bluetooth applications' security (especially in embedded scenarios) rely solely on static PINs with no way to change it.

The low security offered by mainstream Bluetooth deployments, the open accessibility of the radio interface, and the closed-world assumption of in-vehicle CANs raise new and important security concerns. Other researches have recently demonstrated the feasibility of successfully compromising automotive services through the radio interface [6–10]. Most of these attacks exploit software implementation flaws on the ECUs or just the disclosure of the communication protocols of the ECUs to take advantage of implementation flaws. However, no mitigation and defense approaches against these threats have been proposed so far.

3 A Security Layer for Automotive Services

Given the attacker model and the application scenario that we described above, it is necessary to devise an *application-level security mechanism* that is independent from the underlying wireless layer, allows secure communication between portable devices and vehicles, and mitigates the security drawbacks detailed in §2.

Our approach secures the communication with respect to the (generic) attacker model described in §2.1. In addition, our approach brings the benefit of a relaxed dependency on proprietary (untrusted) parts, because it provides a "unified" layer on top of which car-to-X applications can be developed. Our experimental evaluation described in §4 shows that our solution has minimal deployment impact; more importantly, it complies with the real-time requirements and constrained resources of the Gateway ECU, the implementation and distribution of the application's counterpart on mobile devices, and the I/O capabilities of the deployed vehicle.

Before explaining the security approach and the details of our implementation, we provide a brief introduction of the background on our security approach and its analysis.

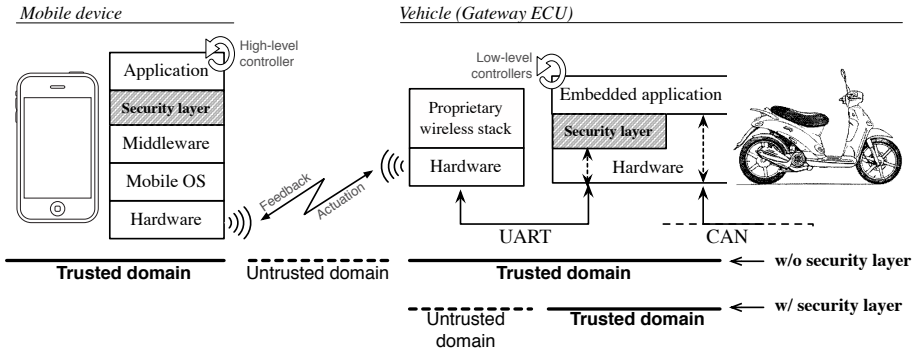


Fig. 1. Overview of the architecture proposed in §4 based on our approach of *trusted domains*.

3.1 Security Analysis

We derive the requirements of our security layer through the evaluation of the application scenario by means of trust domains and relationships between communicating parties (or entities). A party is a *trusted domain* if we trust in the correct processing and execution of the software implementation and thus in the integrity of the entity. Otherwise we consider the party as an *untrusted domain*. Depending on the characteristics and the security properties of the communication between entities, we can define *trusted relationships* (or *accepted dependence*) between entities.

For the sake of backward compatibility, we assume that the security mechanisms available are those provided by the Bluetooth standard—with the known security threats that we discussed in §2.3. The mobile device and the Gateway ECU are each defined as trusted domains. To mitigate the security threats by means of a potential adversary, the focus of our proposed security approach is on the vehicle side. On the mobile device side, an application serves trusted relationships to lower software layers with respect to the security mechanisms offered by the mobile operating system. We assume that the integrity of a mobile application relies on the appropriate security mechanisms (e.g., sandboxing, code signing, rights management, or secure storage) of current mobile devices and operating systems, respectively. Therefore, we focus on the integration of a given mobile system architecture and its security mechanisms without interfering the safety of actual automotive architectures. The trusted domains and relationships are summarized in Fig. 1, which also depicts our application scenario.

3.2 Security Requirements

The results of our analysis are the following security requirements, which describe the background of our security framework:

Req. 1: The execution of any data is based on its context (e.g., the Bluetooth module is only dedicated to transmitting and receiving data without interrupting the execution of the ECU's application layer).

Req. 2: No dependencies on proprietary subparts of an ECU and its interfaces towards other entities.

Req. 3: Cryptographic mechanism must be under the developer's authority.

Req. 4: End-to-end confidentiality and authenticity between the application layer of a service user and an ECU.

For applying our security approach on a real application, we must take into account any security flaws of the radio interface and remove any dependency of the application layer towards proprietary parts on an ECU. In contrast to the security requirements, the current architecture introduces a dependence on the provided information flow by the Bluetooth module for the microcontroller of the ECU. Besides creating a dependence on the security mechanisms of the Bluetooth standard and software implementation of the Bluetooth stack, the processing and execution of the embedded application is explicitly influenced by this data source. According to our security analysis, both entities share a trusted relationship between each other and execute the data in a bidirectional way without any security properties.

3.3 Implementation Approach

We follow a two-stage approach and rely on standardized and state-of-the-art cryptography. The first stage sets up an end-to-end trusted relationship between both application layers (i.e., on the mobile device or service user, and on the ECU). Due to the constraints of the scenario (e.g., distribution of the mobile application through app stores, connectivity capabilities of the ECU), we cannot pre-compute and store any static credentials or cryptographic keys on the mobile device, nor use a public key infrastructure on the ECU. Therefore, we assume that only the vehicle's owner is able to initiate the first stage by enabling the authorization procedure on the vehicle's side, only allowing the authentication of a mobile device user for a distinct time. Within this time span, the ECU accepts the delivery of a service user's identity and the user receives the identity information of the ECU, respectively. In our implementation, the identity information includes the public keys of an asymmetric cryptographic scheme. The second stage ensures that the real-time communication requirements are met. To this end, it implements a symmetric cryptographic scheme that establish a secure communication session, where the session key is derived from the long-term shared secret of the first stage.

For integrating our two-stage approach on the Gateway ECU's microcontroller, we implemented an ECDH key-establishment scheme [11], for the authorization of a mobile device, on a standardized curve (NIST P-192) [12]. For each authentication process, the mobile device computes a new random key set and transmit the corresponding public key to the ECU. In contrast to the key set of the mobile device, the ECU possesses a static long-term key set for the key establishment scheme (see *C(1,1)* in [11]). For the session encryption, we implemented the Advanced Encryption Standard (AES) in a chaining block cipher (CBC) mode [13, 26] with a 128-bits key. The key-derivation function is implemented according to the standard and provides a fresh 128-bits symmetric key for each communication session. We rely on the DSP-capability of the underlying hardware layer [27, 28] to compute the long-term secret and the operations on the finite field of an elliptic curve. This allows us to enhance the speed of multi-precision arithmetic operations. To further optimize the implementation, we developed most of the cryptographic primitives in assembly code. Besides these two cryptographic schemes, we implemented the SHA-1 hash function and defined a protocol structure for the integration in a communication protocol stack.

Instead of implementing cryptographic primitives on a mobile device, we opted to use a standardized cryptographic library to guarantee the proper execution and runtime behavior of the cryptographic primitives. For the integration of our security layer on a mobile device, we choose the *OpenSSL* library. Due to the constrained resources of the Gateway ECU, we enable the mobile device's random number capabilities as a source for any random number needed in the cryptographic protocols.

3.4 Security Analysis of the Framework

We hereby evaluate our approach with respect to the attacker model discussed in §2.1 and show that we can mitigate most of the identified security threats.

Unauthorized Communication: As depicted in Fig. 1, our security framework implements requirements **Req. 1** and **Req. 2** with the help of cryptography and takes into account a compromised communication interface. This prevents the attacker from fulfilling the objectives **Obj. 1** and **Obj. 2** (i.e., compromising the execution flow of the Bluetooth module). The focus of the attacker is to use the radio interface as an intermediate entity to transmit data to an ECU. However, even if an attacker obtains access to the ECU via the radio interface, it is not possible to transmit any commands towards the ECU without the knowledge of a session key or the long-term secret.

Implementation Flaws or Malicious Code Injection: The objectives **Obj. 3** and **Obj. 4** represent the deployment of malicious code on the ECU—with the final goal of issuing specific commands to the in-vehicle network. Typically, the attacker accomplishes the deployment by first interrupting the execution flow of an ECU, by exploiting for instance implementation flaws in the executed software code running in the ECU. **Req. 2** removes the dependency from proprietary implementations, and thus reduces dramatically the risk of exploitation. However, in the unlikely event that the attacker compromises the security mechanisms of the communication interface, she will also need to compromise the encryption provided by our security layer, which we can reasonably assume to be impossible. In addition, **Req. 3** ensures that any implementation flaw or security vulnerability in the deployed cryptographic primitives could also be fixed conveniently via a software update: Our security layer do not rely on any hardware device, instead the security is under the developer's authority.

Disclosure of Cryptographic Primitives: The disclosure of cryptographic primitives is one of the most crucial attacks against the security framework. There is always the possibility of dedicated, physical attacks (e.g., side-channel attacks against the ECU's cryptographic implementations) without using tamper-proof devices. Clearly, the attacker would need to obtain physical access to the ECU. In addition, our security framework makes it non trivial for the attacker to obtain the cryptographic, long-term secret. In particular, a man-in-the-middle (MITM) attack is more difficult to conduct than in regular Bluetooth pairing: The attacker would need to be in the range of communication (i) during the legacy Bluetooth pairing process and (ii) during the first stage of our security framework (i.e., the exchange of the public keys). However, it is reasonable to assume that only the owner of the vehicle is able to enable the authorization process for a mobile device (e.g., in his own garage) and, more importantly, within a predefined and very brief time span.

Instead of compromising the ECU's security layer, an attacker may achieve **Obj. 5** with the help of a dedicated attack against the service user (e.g., mobile malware). First, our security framework addresses this type of security threat by fulfilling **Req. 3** and is flexible with respect to future updates to the mobile device or operating system. In fact, we are able to change any cryptographic primitive or protocol in order to protect from actual or future vulnerabilities and thus fulfill **Req. 4**. Second, we assume that the security of the mobile application—and thus of our security layer–is based on the integrity of the operating system and its services. Note that this assumption applies on any non-trusted computing platform, and thus it is perfectly reasonable.

4 Experimental Evaluation

We hereby describe the experimental evaluation that we conducted using our implementation of the security layer detailed in §3. The goal of our evaluation is to ensure that our security protocol can be used in practice. To this end, we show that the security protocol does not impose any significant communication overhead. Furthermore, we demonstrate that it can be applied and deployed in real-world applications with minimal development efforts and with almost no impact on the existing hardware architecture that needs to be secured.

4.1 Case Study Overview: Electric Powered Two-Wheeler

We deployed and extensively tested our security protocol on an existing prototypical energy-management system for light electric vehicles. This system works as an intelligent range extender. More specifically, the goal of this system is to control and optimize the energy consumed by the vehicle by actively modifying the dynamical behavior of the vehicle in real-time [29, 30]. This task is accomplished with the following cascade structure (refer again to Figure 1):

High-level controller This is the state-of-charge (SoC) controller, designed so that the battery SoC tracks a reference profile. The desired discharge policy is generated according to the a priori knowledge of the track, thus taking into account the total track distance and its elevation profile [31]. As a further degree of freedom for the user, the reference discharge profile depends on the driver's demand for energy saving. A mobile device implements the SoC controller logic by means of a mobile app, which also includes navigation features that leverage the Internet capabilities of the device.

Low-level control loops These loops prevent speed and acceleration to exceed certain limits [32]. The high-level controller updates these bounds according to control algorithms based on optimization procedures. The Gateway ECU implements and runs the low-level control loops on a 16-bits dsPIC microcontroller with a CPU speed of 20 Mips [27]. The PTW exchanges data with this device via CAN messages.

The Gateway ECU and the mobile device communicate via the Bluetooth standard and exchange the data as summarized in Table 2. The mobile device acts as a driver-to-vehicle interface for a service (i.e., energy management) exposed by the vehicle. This paradigm is very appealing and is gaining increasing interest among vehicle manufacturers. First because modern drivers are likely to be already familiar with mobile apps, and secondly

because this deployment method facilitates the spread of software updates and the integration with other web-based services (see for instance [33]).

However, as discussed in §2, the use of Bluetooth for the communication of real-time data requires safety-critical issues to be addressed. In this kind of applications, sensitive data need to be exchanged between the mobile device and the Gateway ECU on the vehicle (see Table 2 for a summary). More precisely, this sensitive data include inputs and outputs used to actively control the vehicle through the energy-management system, as opposed to mere logging or display functionality (e.g., a virtual dashboard embedded in the smartphone application). Therefore, if the data is compromised, then the functionality of the control system may be severely altered. As a consequence, the vehicle "driveability" may decrease and, depending on the attacker's skills, the driver could loose control of the vehicle.

Based on the case study, we successfully secured the existing architecture using our security framework (as described in §3) and conducted the experiments described in the remainder of this section with an actual PTW developed by Piaggio, a very large Italian motorbike manufacturing company, and currently in production. This PTW is used within the Green Move[3] research project, a two-year project funded by the Lombardy Region, involving eight research centers at Politecnico di Milano (Italy). This case study allows us to assess the feasibility of the security approach, characterize its computational performance, and highlight the impact on the overall control strategy.

In addition to these measurements, we also collected very positive feedback from Piaggio and from a selected pool of end users. Clearly, this is by no means intended to be a thorough usability study (which is part of our planned future work), but it provides a qualitative idea of the perceived ease of use and simplicity of our approach. We argue that this is an important aspect to ensure a good level of acceptance of automotive security solutions.

4.2 Working and Measurements Conditions

The overall system has the two fundamental working modes *pairing* and *payload exchange*, during which different types of data are exchanged. These working modes are the most critical ones in terms of computational burden and provide measurable feedback on how the security layer impacts the dynamic behavior of the control system. Thus, we concentrate our performance measurements on these two modes as we discuss in the following:

Pairing (one shot) This mode is active when the mobile device is paired with the vehicle, *after* the typical Bluetooth pairing mechanism has taken place. For this

[3] http://www.greenmove.polimi.it

Table 2. Information exchanged between the Gateway ECU and the mobile device.

FROM	TO	KIND	SIZE [bytes]	FREQUENCY
Mobile device	Gateway ECU	Initialization	48	One-shot
Gateway ECU	Mobile device	Real-time (control data)	60	5 [Hz]
Mobile device	Gateway ECU	Real-time (control data)	6	every $\overline{s}$ [m]

mode, we measure the performance of the asymmetric cryptography both on the mobile device and on the Gateway ECU. Moreover, as the key generation routine runs on the mobile device, we also quantify its performance.

Payload Exchange (runtime) This is when the AES key exchange, encryption and decryption take place. For this mode, we analyze the decryption on the mobile device and the encryption on the Gateway ECU. The payload consists of 64 bytes of data, which includes a padding scheme for supporting arbitrary payload size.

Note that the pairing is a one-shot task, whereas the system normally works in payload-exchange mode. At runtime, the payload exchange must satisfy real-time constraints. Here, the bottleneck lies in the Bluetooth stack as the AES encryption-decryption of the 64-bytes payload is executed each time one of the peers transmits or receives a message via Bluetooth (i.e., every 200 ms). Thus, given its importance, we collect runtime data both with a simulator and on a real implementation deployed on the PTW. This ensures an accurate performance characterization. More precisely, during the simulation, the gateway is not connected to the vehicle, but the Bluetooth connection is still active.

Impact of Interrupts: An important aspect to consider is the non-deterministic behavior of the vehicle that may affect the performance of the system on the Gateway ECU: the interrupts on the microcontroller may interfere with the execution of the security code in a noticeable way and have a significant impact. This issue is manifested at runtime, when interrupts from the CAN bus and the UART decrease the normal sequential behavior of the executed code. Note that we can disable the interrupts on the Gateway ECU at pairing time since we need no measurements from the vehicle— the reason is that the application layer does not execute any safety crucial data during the devices' pairing.

4.3 Measured Performance Indicators

Table 3 summarizes how, when and where we measured the execution time in our experiments. Both at runtime and pairing time, the execution time is a significant performance indicator. To measure the execution time on the Gateway ECU, we acquire the number of instructions N executed when the code runs and divide it by CPU speed c, thus obtaining the time elapsed in seconds, or in number of clocks (short for "clock cycles")—on the Gateway ECU. In the remainder of the paper, we explicitly plot the number of clocks N needed to execute the code so as to make the analysis independent from the actual CPU speed of the Gateway ECU.

We implemented data-logging routines on the mobile device that receive from the Gateway ECU samples of the number of clocks N. These samples are submitted within the exchanged Bluetooth messages. Although this strategy has the minor side effect of an increased payload size, it makes data collection easier compared to, for instance, collecting data directly on the Gateway ECU. In addition, our results show that this has no significant impact on the results.

4.4 Performance Measurements

Table 3 summarizes the average results that we obtained from running our experiments. In the following, we explain these results in more detail and provide more analysis results regarding the performance measurements.

Pairing: The pairing phase is characterized by the computational time needed to generate the key set on the mobile device and the execution of the ECDH protocol needed to perform the authentication scheme between the two devices.

Fig. 2 and 3 show the execution time measured on the Gateway ECU and on the mobile device, respectively. The small average values of the measurements both on the ECU and on the smartphone proves the feasibility of our proposed implementation in a real application. Obviously, the bottleneck of the key exchange is the Gateway ECU due to its lower CPU speed: the execution time on the microcontroller is approximately 130 ms, that is 20 time bigger than the average computational time recorded on the mobile device. In addition, notice that the results achieved on the Gateway ECU are very similar both in simulation and while driving with the vehicle; this proves that disabling the interrupts—as explained in §4.2—is beneficial for the execution time.

Payload exchange: The communication overhead at runtime affects the day-to-day use of the mechanism. Hence, a significant overhead would lead to functionality issues on the control system. Fig. 4 shows the measurements from the Gateway ECU. As expected, the simulated results differs from the on-vehicle tests: the quasi-periodical pattern shown by the real-time data while driving the electric PTW is mainly due to the periodical interrupts of the UART and the CAN bus on the microcontroller, as discussed earlier in this section. This is also clearly depicted in the statistical domain, as summarized in the boxplot shown in Fig. 5, which clearly shows that the average values obtained during four different tests (5000 samples for each test) are remarkably constant. The maximum and minimum values are mainly due to the oscillations induced by the interrupts.

Again, the time required by the smartphone for executing the security layer code can be neglected while working on the synthesis of the control loop. The measurement results for the AES decryption of 64 bytes payloads on the mobile device are shown in Fig. 6.

Table 3. Summary of the average values that we obtained over 5000 samples collected by running each routine on the mobile device and on the Gateway ECU in both simulation mode (S) and while driving (D).

MODE	PHASE	DEVICE	AVERAGE VALUE	TEST
Runtime	Data encryption (64-bytes payload)	Gateway ECU	50.52 kClocks	S
			51.41 kClocks	D
		Mobile device	33.98 μs	S
			34.5 μs	D
Pairing	Key establishment protocol	Gateway ECU	2626.21 kClocks	S
			2628.67 kClocks	D
		Mobile device	7161.2 μs	D
	EC key generation	Mobile device	6939.8 μs	D

Impact of our Security Protocol on Execution Time: Fig. 7 shows the instantaneous Bluetooth sending frequency, which provides a concise view of the impact of the security layer on the real-time exchange of data. We derived this frequency values by first measuring the time interval ΔT between two received data frames on the smartphone. Therefore, the instantaneous frequency f_b is equal to:

$$f_b = \frac{1}{\Delta T} = \frac{1}{\Delta T_d + \Delta T_r + \Delta T_e + \Delta T_b}$$

ΔT_d and ΔT_e are the computational time of the decryption and of the encryption, respectively. ΔT_r is a random time interval between two sent messages, and ΔT_b is the time needed by the Bluetooth stack to send and receive data.

The average values of the Bluetooth frequency with and without the security layer are 4.83 Hz and 5.01 Hz, respectively. The cause of this slight discrepancy is twofold. On the one hand, the security layer introduces a delay because the terms ΔT_d and ΔT_e are significant, as shown in Table 3. On the other hand, the size of the message sent via Bluetooth is 40% larger compared to the case where the security layer is disabled. Therefore, different payload sizes lead to different behaviors. In general, the increased size of the message decreases the Bluetooth frequency due to the low-level mechanisms implemented in the Bluetooth stack. Despite this slight decrease of sending frequency, the performance of the closed-loop system is not affected by the security routines when the high-level control strategies equipped with this additional layer are tested on the electric PTW.

5 Discussion and Future Work

While our reference implementation is capable of providing a security session layer that ensures end-to-end security transparently, there are three aspects that need fur-

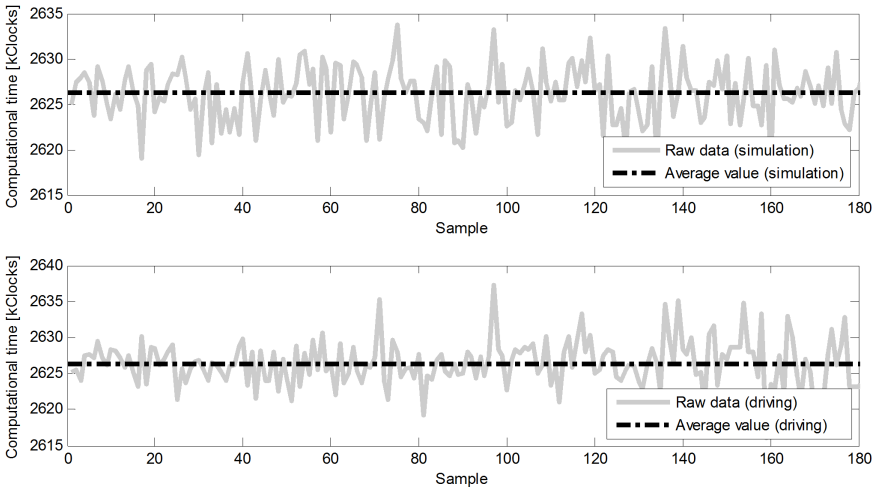
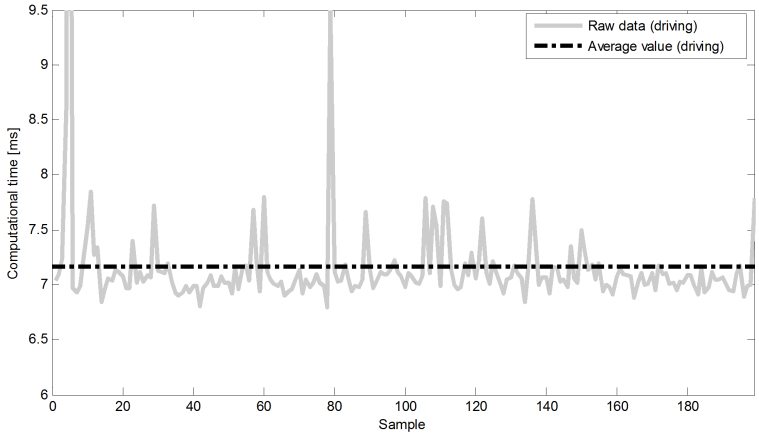
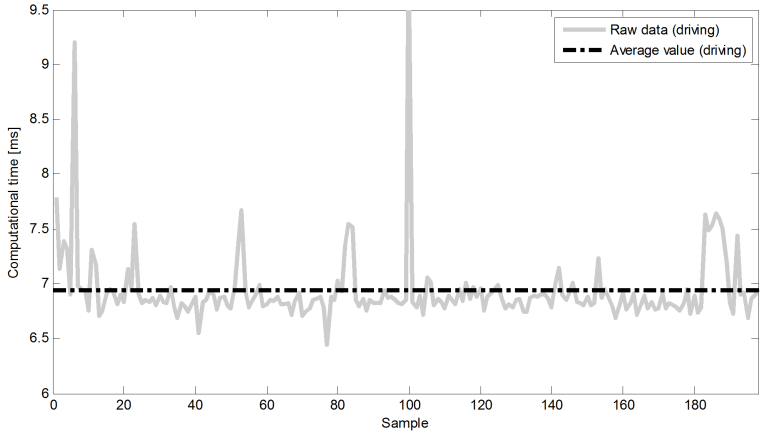


Fig. 2. Execution time for the computation of the ECDH protocol measured on the Gateway ECU. Top plot: simulation. Bottom plot: on-vehicle tests.



(a) Key establishment protocol.



(b) EC key generation.

Fig. 3. Execution time for the pairing phase measured on the mobile device. The measurements have been taken independently from each other.

ther investigation in the future. First, in this work we concentrated on one symmetric encryption algorithm (i.e., AES/FIPS 197) and an elliptic curve key establishment protocol. Depending on the specific needs of the application domain or case study, other algorithms may be implemented and tested. For example, if a security session layer should be established for an ECU with even less computational power than the Gateway ECU, lightweight cryptographic algorithms like PRESENT [34] might be more suitable. However, embedding other algorithms in our system only requires implementation—in assembly, as we did for AES/FIPS 197 and ECDH/NIST P-192—and integration.

The second and third aspect that could be investigated further both regard the evaluation of our approach. As discussed in §4.1, we already collected some initial feedback

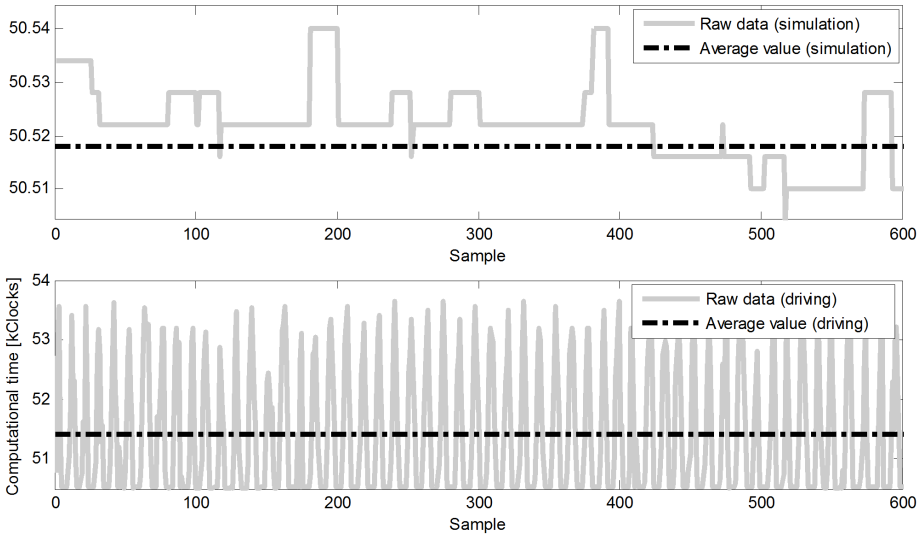


Fig. 4. Measurements acquired for the encryption of 64 bytes payloads on the Gateway ECU. Top plot: Simulation. Bottom plot: On vehicle tests.

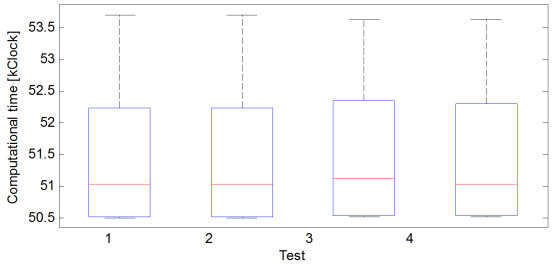


Fig. 5. Performance of the encryption of 64 bytes payloads. Boxplot of four different acquisitions on the Gateway ECU while driving the electric PTW.

from real-world users, which helped us in the design of the user interactions. This initial feedback met the qualitative evaluation needs of this paper, whereas an extensive usability study could help improving the user-interaction aspects of our approach—although these are slightly out of scope for this paper.

Furthermore, the impact of our security layer on battery life should be measured, although we expect no remarkable results. More precisely, as we discussed in §4.4, our security layer barely affects the execution time; consequently, the computational resources of the mobile device are little affected as well. Therefore, we expect that the battery life is also not affected significantly. These conclusions are also substantiated by a series of short (e.g., 10 to 20 minutes) test drives that we performed while we monitored the battery discharge: We noticed no discrepancy when driving with and without the security layer enabled.

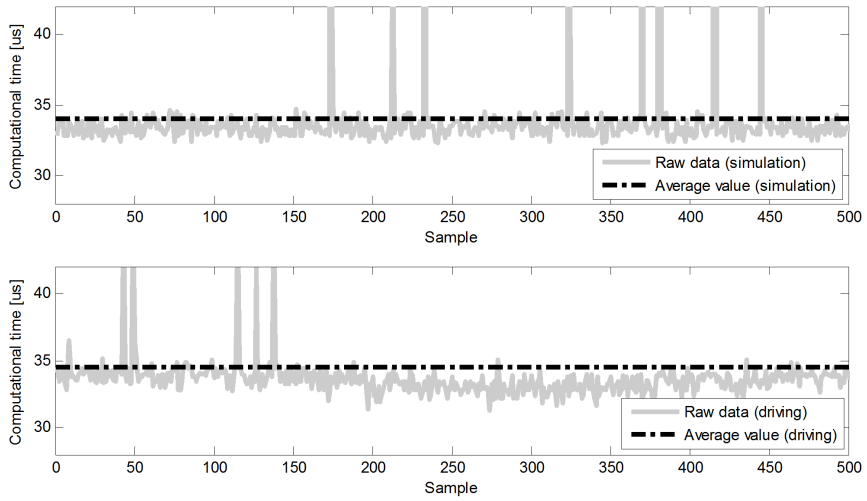


Fig. 6. Measured execution time acquired for the decryption of 64 bytes payloads on the mobile device. Top plot: Simulation. Bottom plot: On vehicle tests.

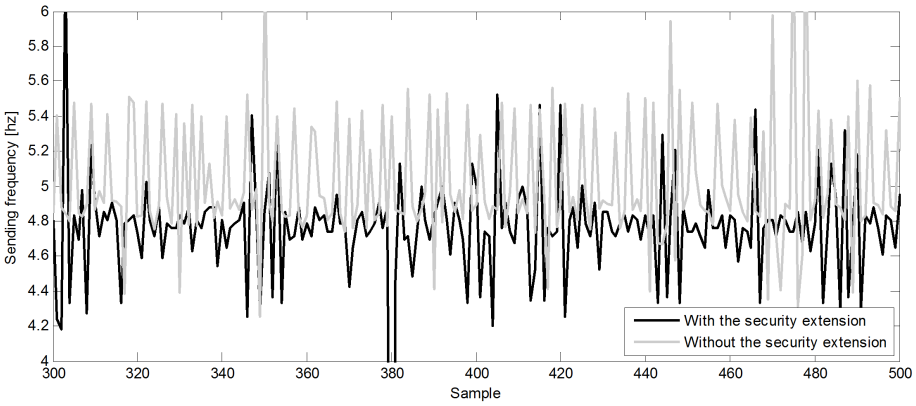


Fig. 7. Instantaneous Bluetooth sending frequency estimated with and without the security layer.

6 Conclusions

We proposed a security layer that sits on top of the Bluetooth standard (or actually, any other communication layer), ensuring a secure communication between smartphones and in-vehicle networks. This enables modern automotive services to interact with vehicles in a secure manner. Our proposed approach can be applied to real-world cases, as shown by our practical evaluation, because (1) it has very *low impact* on the (often small) computational resources available on the vehicle and the smartphone, (2) it requires *no hardware modifications* (i.e., it is agnostic with respect to the adopted wireless communication standard), and (3) it requires no complex user interactions.

We implemented our proposed system on an electric vehicle and an iPhone application that actively monitors the vehicle's battery and controls the driving speed, so that the battery lasts longer. This case study is suitable for our proposed system, because the mobile device and the vehicles exchange sensitive control data, which may affect the vehicle driveability. Our tests on this case study confirm that our system meets both the security and the real-time performance requirements.

We conclude that our approach effectively mitigates the security threats that commonly affect car-to-X applications. Furthermore, the recent attacks against cars [6–10] would be significantly harder if a security session layer would be used in vehicles since simple sniffing of protocol messages is not feasible anymore given our approach.

7 Acknowledgments

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement nr. 257007 (SysSec) and from the German Federal Ministry of Economisc and Technology under grant agreement nr. 01ME12025 (SecMobil). The opinions expressed in this paper are those of the authors and do not necessarily reflect the views of the European Commission.

References

1. Mercedes-Benz mbrace2™: Connected in-vehicle services. `http://www.mbusa.com/mercedes/mbrace2` (2012)
2. BMW: ConnectedDrive. `http://www.bmw.com/com/en/insights/technology/connecteddrive/2010/index.html` (2012)
3. sim^{TD}: Safe and Intelligent Mobility Test Field Germany. `http://www.simtd.de` (2008)
4. EVITA: E-safety vehicle intrusion protected applications. `http://evita-project.org` (2008)
5. PRESERVE: preparing secure v2x communication systems. `http://www.preserve-project.eu` (2011)
6. Rouf, I., Miller, R., Mustafa, H., Taylor, T., Oh, S., Xu, W., Gruteser, M., Trappe, W., Seskar, I.: Security and privacy vulnerabilities of in-car wireless networks: a tire pressure monitoring system case study. In: Proceedings of the 19th USENIX conference on Security. USENIX Security'10, Berkeley, CA, USA, USENIX Association (2010) 21–21
7. Koscher, K., Czeskis, A., Roesner, F., Patel, S., Kohno, T., Checkoway, S., McCoy, D., Kantor, B., Anderson, D., Shacham, H., Savage, S.: Experimental security analysis of a modern automobile. In: Proceedings of the 2010 IEEE Symposium on Security and Privacy. SP '10, Washington, DC, USA, IEEE Computer Society (2010) 447–462
8. Checkoway, S., McCoy, D., Kantor, B., Anderson, D., Shacham, H., Savage, S., Koscher, K., Czeskis, A., Roesner, F., Kohno, T.: Comprehensive experimental analyses of automotive attack surfaces. In: Proceedings of the 20th USENIX conference on Security. SEC'11, Berkeley, CA, USA, USENIX Association (2011) 6–6
9. Hoppe, T., Kiltz, S., Dittmann, J.: Security threats to automotive CAN networks — practical examples and selected short-term countermeasures. In: Proceedings of the 27th international conference on Computer Safety, Reliability, and Security. SAFECOMP '08, Berlin, Heidelberg, Springer-Verlag (2008) 235–248
10. Wright, A.: Hacking cars. Commun. ACM **54**(11) (November 2011) 18–19
11. NIST Special Publication 800-56A: Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography (2007)

12. FIPS-186-3: Digital Signature Standard (DSS). NIST (2009)
13. FIPS-197: Advanced Encryption Standard (AES). NIST (2001)
14. General Motors: OnStar®. http://www.onstar.com/web/portal/home (2012)
15. Audi at the CES 2012: Intelligent networking with Audi connect™. http://www.audi-mediaservices.com (2012)
16. Xiang, C., Binxing, F., Lihua, Y., Xiaoyi, L., Tianning, Z.: Andbot: towards advanced mobile botnets. In: Proceedings of the 4th USENIX conference on Large-scale exploits and emergent threats. LEET'11, Berkeley, CA, USA, USENIX Association (2011) 11–11
17. Damopoulos, D., Kambourakis, G., Gritzalis, S.: iSAM: An iPhone Stealth Airborne Malware. In Camenisch, J., Fischer-Hübner, S., Murayama, Y., Portmann, A., Rieder, C., eds.: Future Challenges in Security and Privacy for Academia and Industry. Volume 354 of IFIP Advances in Information and Communication Technology. Springer Boston, Berlin, Heidelberg (2011) 17–28
18. Enck, W., Octeau, D., McDaniel, P., Chaudhuri, S.: A Study of Android Application Security. In: USENIX Security Symposium. (2011)
19. International Standard Organization: Road vehicles - Controller area network (CAN). ISO 11898:2003 (2003)
20. Wolf, M., Weimerskirch, A., Paar, C.: Security in Automotive Bus Systems. In: Embedded Security in Cars Workshop. (2004)
21. Hager, C., MidKiff, S.: An analysis of bluetooth security vulnerabilities. In: Wireless Communications and Networking, 2003. WCNC 2003. 2003 IEEE. Volume 3., IEEE (2003) 1825–1831
22. Hager, C., Midkiff, S.: Demonstrating vulnerabilities in bluetooth security. In: Global Telecommunications Conference, 2003. GLOBECOM'03. IEEE. Volume 3., IEEE (2003) 1420–1424
23. NIST Special Publication 800-121: Guide to Bluetooth Security: Recommendations of the National Institue of Standards and Technology (2008)
24. Haataja, K., Hypponen, K.: Man-in-the-Middle Attacks on Bluetooth: A Comparative Analysis, a Novel Attack, and Countermeasures. In: Communications, Control and Signal Processing, 2008. ISCCSP 2008. 3rd International Symposium on, IEEE (2008) 1096–1102
25. Haataja, K., Toivanen, P.: Two Practical Man-in-the-Middle Attacks on Bluetooth Secure Simple Pairing and Countermeasures. Trans. Wireless. Comm. **9**(1) (January 2010) 384–392
26. NIST Special Publication 800-38A: Recommendation for Block Cipher Modes of Operation - Methods and Techniques (2001)
27. Microchip Technology Inc.: 16-bit dsPIC® Digital Signal Controllers
28. Wenger, E., Werner, M.: Evaluating 16-bit processors for elliptic curve cryptography. Smart Card Research and Advanced Applications (2011) 166–181
29. Savaresi, S.M., Dardanelli, A., Tanelli, M., Picasso, B., Di Tanna, O., Santucci, M.: System and method for the active management of the driving range of a vehicle, with particular reference to electric vehicles. Italian Patent n. MI2011A000393, filed on 11/03/2011 Applicant: Piaggio & C. S.p.A. and Politecnico di Milano.
30. Dardanelli, A., Tanelli, M., Picasso, B., Savaresi, S.M., Di Tanna, O., Santucci, M.: A smartphone-in-the-loop active state-of-charge manager for electric vehicles . IEEE Transactions on Mechatronics (2012) To appear.
31. Dardanelli, A., Tanelli, M., Savaresi, S.M.: Active energy management of electric vehicles with cartographic data. In: Electric Vehicle Conference, 2012 IEEE International. (2012) To appear.
32. Dardanelli, A., Tanelli, M., Picasso, B., Savaresi, S.M., di Tanna, O., Santucci, M.: Speed and acceleration controllers for a light electric two-wheeled vehicle. In: Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on. (dec. 2011) 2523 –2528

33. Spelta, C., Manzoni, V., Corti, A., Goggi, A., Savaresi, S.M.: Smartphone-based vehicle-to-driver/environment interaction system for motorcycles. IEEE Embedded Systems Letters **2**(2) (2010) 39–42
34. Bogdanov, A., Leander, G., Knudsen, L.R., Paar, C., Poschmann, A., Robshaw, M.J., Seurin, Y., Vikkelsoe, C.: PRESENT—An Ultra-Lightweight Block Cipher. In: International Workshop on Cryptographic Hardware and Embedded Systems (CHES). Number 4727 in LNCS, Springer (2007) 450–466