A Security Layer for Smartphone-to-Vehicle Communication Over Bluetooth

A. Dardanelli, F. Maggi, M. Tanelli, S. Zanero, S. M. Savaresi, R. Kochanek, and T. Holz

***Abstract*—Modern vehicles are increasingly being interconnected with computer systems, which collect information both from vehicular sources and Internet services. Unfortunately, this creates a nonnegligible attack surface, which extends when vehicles are partly operated *via* smartphones. In this letter, a hierarchically distributed control system architecture which integrates a smartphone with classical embedded systems is presented, and an *ad-hoc*, end-to-end security layer is designed to demonstrate how a smartphone can interact securely with a modern vehicle without requiring modifications to the existing in-vehicle network. Experimental results demonstrate the effectiveness of the approach.**

***Index Terms*—Automotive systems, embedded architecture, security, smartphone, two-wheeled vehicles.**

I. Introduction

THE current trend in automotive products and services is to improve the accessibility of the vehicles through novel services, which require a connection to some Internet-based source. This is used both to collect information on the external environment (e.g., traffic conditions, weather forecasts, vehicle position and orientation, often integrated within the on-board vehicle control systems), and to offer "infotainment" services. In doing so, the new devices that interact with the vehicle (e.g., modern infotainment systems, GSM, and Bluetooth connections) lead to an increased attack surface, which may enable an adversary to break into the vehicle itself, causing severe safety hazards. Recently, several researchers highlighted this aspect and successfully demonstrated attacks against different vehicles [1], [2]. Each of these works showed that it was possible to take control of certain functionalities of the vehicle, and interfere with safety-critical or sensitive components. These vulnerabilities hamper novel solutions (e.g., smartphones to unlock the vehicle door or to start the engine), because of the risk of successful attacks. Adding security mechanisms to vehicles is a challenging task, as the related embedded architectures are commonly designed with safety requirements rather than security ones in mind. In addition, the available computing resources are in general tailored to fit tightly to the control systems needs, in order to limit the costs. This obviously restricts the options available for adding a security layer in subsequent redesign phases. Recently, however, security requirements are gaining more and more priority, especially in the communication between embedded computers with ad-hoc wireless networks (see, e.g., [3]).

In this letter, we consider a smartphone-in-the-loop vehicle architecture in the use case of a two-wheeled vehicle (see [4]). We propose a security solution that protects against attacks by addressing the challenges raised above, meeting both performance and real-time constraints. Furthermore, we explicitly take the capabilities of the target architecture into account (i.e., no input capabilities on the vehicle side, limited output capabilities, and lack of a trusted execution environment on the mobile device). Our proposed solution allows a smartphone to establish a *secure session layer* over an insecure radio connection, which provides additional security guarantees regardless of the security mechanisms already implemented in the physical layer (if any). As a result, the entire application layer is transparently secured. The applicability and the effectiveness of the proposed solution is corroborated by experimental results.

The structure of this letter is as follows. Section II presents the vehicle embedded architecture, while Section III details the related security issues. Further, Section IV describes the designed security solution. Section V, the experimental results are presented and discussed.

II. System Architecture

The system architecture under consideration directly relates to the vehicle control logic. As detailed in [4], the control logic can be split into two main stages: the "high-level" stage take cares of the vehicle motion and energy control, while the "low-level" stage takes care of data acquisition and actuation. This layout is quite common in complex automotive control systems, as they are often characterized by cascade structures that exploit the frequency-separation paradigm in order to decouple nested control loops. Accordingly, we translate the logical division between the two control loops into a technological separation: The high-level and the low-level control routines run on different devices, thus leading to a hierarchically distributed control system. The border between these two levels of abstraction may be traced according to different policies such as computational considerations, safety issues, technological constraints. Clearly, the two subsystems must share information. As a consequence, a communication channel that guarantees the robust and persistent interconnection between the two subsystems is needed. Thus, as depicted in Fig. 1, the system architecture comprises two main elements. The first element is the Gateway *electronic control unit* (ECU), which is physically mounted on the vehicle, runs the low-level control logic, and

Manuscript received January 23, 2013; accepted April 18, 2013. Date of publication June 21, 2013; date of current version August 26, 2013. This work was supported by the EU FP7 Project "SysSec" (257007). Opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the EU Commission. This manuscript was recommended for publication by A. Ghosal.

A. Dardanelli, F. Maggi, M. Tanelli, S. Zanero, and S. M. Savaresi are with the Dipartimento di Elettronica e Informazione, Politecnico di Milano, Milano 20133, Italy (e-mail: dardanellielet.polimi.it; maggielet.polimi.it; tanelli, zaneroelet.polimi.it; savaresi@elet.polimi.it).

R. Kochanek and T. Holz are with the Department of Electrical Engineering and Information Technology, Ruhr-University Bochum, Bochum 44801, Germany (e-mail: roman.kochanek@rub.de; thorsten.holz@rub.de).

Digital Object Identifier 10.1109/LES.2013.2264594

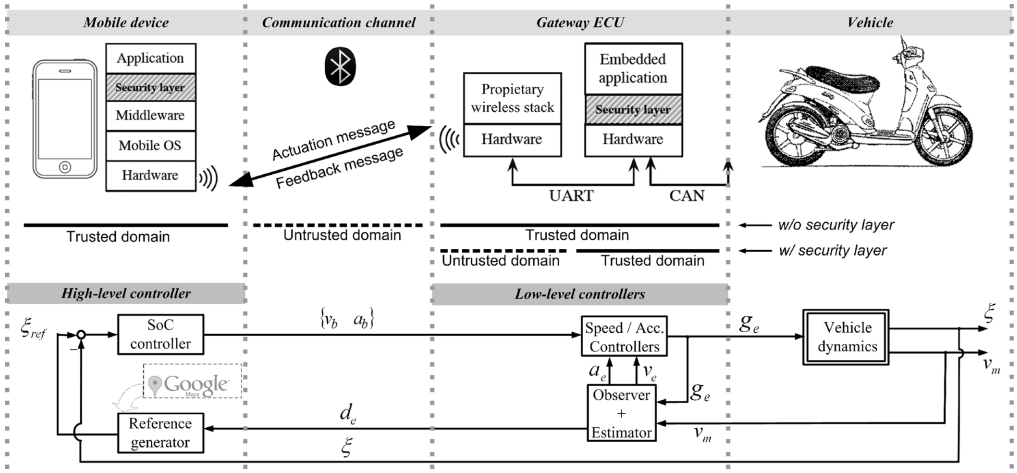

Fig. 1. System overview. The hierarchically distributed architecture and the security layer extensions are based on our approach of *trusted domains*.

communicates with sensors and actuators *via* an in-vehicle network (e.g., the CAN bus). The Gateway ECU is equipped with a radio interface that allows wireless communication between the in-vehicle network and external devices. The second element is an *external device* that works closely with the vehicle ECUs via the radio interface. In our scenario, the external device is a mobile device that runs the high-level control routines and acts as a driver-to-vehicle interface. This paradigm is very appealing and is gaining increasing interest among vehicle manufacturers, as drivers are likely to be already familiar with mobile apps, and because this deployment method facilitates both software updates and the integration with other web-based services [5].

We successfully implemented the aforementioned system architecture. Specifically, we implemented an intelligent range extender for lightweight electric vehicles, with the goal of optimizing the energy consumption by actively modifying the vehicle dynamic behavior, as detailed in [4]. This task is accomplished with a two-layer structure. A high-level controller keeps track of a reference profile, ξ_{ref}, for the battery *state of charge* (SoC), ξ. The profile is generated by taking into account the route length and its elevation profile, as detailed in [6]. The mobile device implements the SoC controller within an *ad-hoc* app that we developed, which also includes navigation features that leverage on Internet-based services (e.g., Google Maps API). Furthermore, the low-level control loops enforce speed and acceleration constraints (v_b and a_b in, which allow to meet the desired energy consumption profile. The low-level controllers act on the gas handle opening g_e to guarantee that the dynamical behavior of the vehicle (i.e., speed v_e and acceleration a_e) is kept within the prescribed limits. The Gateway ECU implements and executes the low-level control loops on a 16-bits dsPIC microcontroller with a CPU speed of 20 Mips [7], and communicates with sensors and actuators *via* CAN bus. The Gateway ECU and the mobile device communicate *via* a Bluetooth layer. They exchange both initialization and real-time control data. Initialization data is packed into a 48 bytes frame and the communication is unidirectional—from the mobile device to the gateway ECU. On the contrary, the real-time communication is bidirectional: The Gateway ECU sends a 64-bytes payload every 0.2 s (5 Hz), whereas the mobile device communicates a 6-bytes control-data packet every time the vehicle travels 50 m. Simulation results and experimental data collected on a prototype light 2-wheeled electric vehicle prove the effectiveness and the robustness of the proposed approach. The vehicle equipped with the SoC controller saves approximately 20% of the energy supplied by the battery, with respect to a nominal driving behavior.

III. Security Issues

In the aforementioned scenario, the mobile device and the Gateway ECU exchange sensitive data. If this data is compromised by an adversary, then the functionality of the control system, and thus the vehicle "driveability" may be severely affected: Depending on the attacker's skills, the driver even could loose the control of the vehicle. Our focus is on the Bluetooth layer. The protocol has a two-phase session setup: after the *pairing process*, which allows the peers to get to know each other and set up the network properties, the actual *communication* is enabled. Depending on the protocol version, different security features are available. However, the early Bluetooth standard and its successors, with the introduction of the *secure simple pairing* (SSP) protocol [8], suffer from various security vulnerabilities due to weak cryptographic primitives, as discussed in [9], [10]. The security of most Bluetooth applications (e.g., in embedded scenarios) relies on a static PIN only, with no way to change it.

IV. A Security Layer for Automotive Services

Given the application scenario and the aforementioned security issues, it is necessary to devise an *application-level security mechanism* that mitigates the vulnerabilities that lie in the wireless link. Such security layer must be independent from the underlying wireless layer and must allow secure communication between the mobile device and the vehicle. In our attack model the adversary knows the radio protocol in use, and is able to transmit and receive arbitrary data packets on the radio interface. The objective of the attacker is to obtain access to the information exchanged between the vehicle and the mobile device, and ultimately manipulate the ECU execution flow. We concentrate on the application layer. Therefore, attacks against the physical layer (e.g., jamming) or attacks that require physical, even temporary, access to the vehicle (e.g., forceful shutdown) fall outside the scope of our security layer.

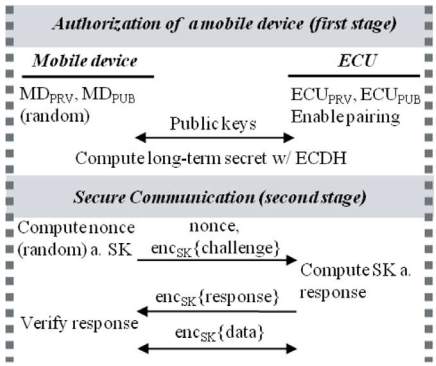

Fig. 2. Two stage security protocol: the session key (SK) is computed with a nonce, and the long-term secret is computed via a key-derivation function.

A. Security Analysis

We derive the requirements of our security layer through the evaluation of the application scenario by means of trust domains and trust relationships between communicating parties (or entities). A party is considered a *trusted domain* if we trust its correct processing and execution of the software implementation, and thus its integrity. 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. The trusted domains and relationships are summarized in Fig. 1. Our solution is designed to account for security flaws (e.g., unknown data leaks) or dependencies of the application layer on proprietary parts of an ECU.

B. Secure Session Layer

Our security layer follows the two-stage protocol depicted in Fig. 2. The *first stage* sets up an end-to-end trusted relationship between both application layers (i.e., on the mobile device 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 do not assume any precomputed, static credentials or cryptographic keys on the mobile device, nor use a public-key infrastructure on the ECU: Only the vehicle's owner is able to initiate the first stage by enabling the one-off authorization procedure on the vehicle's side. For instance, this procedure can be enabled by pushing a button—only reachable using the vehicle key. A classic PIN-based procedure is not always feasible, due to the limited input capabilities on the ECU side (e.g., absence of keypads). Within a short time span the ECU accepts a mobile device's identity and the user receives the identity information of the ECU, respectively. The *second stage* ensures that the real-time communication requirements are met. To this end, it implements a symmetric cryptographic scheme that establishes a secure communication session. The symmetric session key is derived from the long-term secret exchanged during the first stage, plus some random data generated on the mobile device.

We implemented our two-stage approach on the Gateway ECU's microcontroller. We used a ECDH key-establishment scheme (asymmetric cryptography) [11] on a standardized curve (NIST P-192) [12]. For each authentication process, the mobile device computes a new random key set and transmits 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 AES in a chaining block cipher (CBC) mode [13], [14] with a 128-bit key. The key-derivation function is implemented according to the standard and provides a fresh 128-bit symmetric key for each session. For the implementation of our security layer on the mobile device, we choose the *OpenSSL* library. 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.

C. Security Evaluation

Our solution can mitigate the security threats described in Section III under the adversary model described in Section IV. The goal of our security layer is to prevent attacks through the radio interface. As depicted in Fig. 1, our solution implements cryptographic session layer, which removes the dependency from proprietary implementations, dramatically reducing the risk of exploitation. Even if the attacker obtains access to the ECU via the radio interface, the application data is encrypted with the session key. The attacker has less chances of obtaining the cryptographic, long-term secret, than in a regular Bluetooth pairing. In particular, a man-in-the-middle attack is difficult to conduct: the attacker would need to be within the communication range: 1) during the legacy Bluetooth pairing process; and 2) during the first stage of our security protocol (i.e., the exchange of the public keys). Only the vehicle owner can enable the authorization process for a mobile device (e.g., in his own garage) and, more importantly, within a predefined and short time span. Instead of compromising the ECU's security layer, an attacker may perform a dedicated attack against the mobile device (e.g., mobile malware). Our security framework addresses this type of security threat by providing the cryptographic mechanisms under the developer's authority 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. 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.

V. Experimental Results

In this section, we present the experimental results and explain the feasibility of our proposed solution. In our experimental setting, the Gateway ECU is installed on a lightweight, electric two-wheeled vehicle currently in production and manufactured by Piaggio. As explained in Section II, the Gateway ECU implements intelligent, range-extending algorithms. Our security protocol has the two main working modes: *pairing* and *payload exchange*. Pairing is active when the mobile device is paired with the vehicle, *after* the typical Bluetooth pairing mechanism has taken place. In pairing mode, we measured the performance of the asymmetric cryptography both on the mobile device and on the Gateway ECU, and the performance of the key-generation routine (on the mobile device). The payload exchange mode activates when the AES key are actually exchanged, and encrypted or decryption takes place. In this mode, we analyzed the performance of the decryption (on the mobile device) and the performance of the encryption (on the Gateway

TABLE I
AVERAGE EXECUTION TIMES (5000 TESTS). MD: MOBILE DEVICE; (G) GATEWAY ECU; (S): SIMULATION AND (D) DRIVING MODE

MODE	PHASE	DEVICE	AVERAGE VALUE		TEST
Runtime	Data encryption (64-bytes payload)	G	2.52	ms	S
			2.57	ms	D
		MD	33.98	μs	S
			34.5	μs	D
Pairing	Key establishment protocol	G	131.31	ms	S
			131.43	ms	D
		MD	7161.2	μs	D
	EC key generation	MD	6939.8	μs	D

ECU). The payload consists of 64 bytes of data, which includes a padding scheme for supporting arbitrary payload size.

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, because the AES encryption-decryption of the 64-bytes payload is executed every time one of the peers transmits or receives a message via Bluetooth (i.e., every 200 ms). To test its performance, we collected runtime data both on a simulator and on a real implementation on the test vehicle, to ensures an accurate characterization. Both at runtime and pairing time, the execution time is a significant performance indicator. Table I summarizes the results that we obtained (averaged on all experiments). The results support the feasibility of the proposed approach in practical applications. As expected, the bottleneck of the key exchange is the Gateway ECU due to its lower CPU speed: The execution time is approximately 130 ms—20 time bigger than the average time recorded on the mobile device.

Last, we analyzed the instantaneous Bluetooth sending frequency, f_b, which provides a concise view of the impact of the security layer on the real-time exchange of data. To this end, we measured the time interval ΔT between two received frames on the smartphone, and computed the frequency as $f_b = 1/\Delta T = 1/(\Delta T_d + \Delta T_r + \Delta T_e + \Delta T_b)$, where ΔT_d and ΔT_e are the time of the decryption and of the encryption, respectively; ΔT_r is a random time interval between two sent messages, whereas 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(see Table I). On the other hand, the size of the message sent *via* Bluetooth is 40% larger if the security layer is enabled, and different payload sizes induce 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.

VI. FUTURE WORK

First, depending on the specific needs of the application domain or case study (e.g., ECU or mobile device upgrades), other encryption algorithms may be implemented and tested. For example, if a security session layer must be established with less-powerful ECUs, lightweight cryptographic algorithms such as PRESENT [15] might be more suitable. Interestingly, 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. Secondly, the impact of the security layer on battery life should be measured, although we expect no remarkable results. Indeed, as our security layer barely affects the execution time, the computational resources of the mobile device will be little affected as well. Finally, future work includes the requirements for security engineering on embedded automotive devices considering software attack surfaces (*e.g., buffer overflows*). The objective is to develop appropriate countermeasures on the application layer for a holistic security framework.

VII. CONCLUDING REMARKS

We analyzed and discussed the security issues related to modern, smartphone-based automotive embedded architectures. To cope with such issues, we designed, implemented and evaluated a security layer that protects from an attacker that has full control over the Bluetooth wireless link between the mobile device and the vehicle. Thanks to our evaluation, we showed the real-world applicability of our approach.

REFERENCES

[1] I. Rouf, R. Miller, H. Mustafa, T. Taylor, S. Oh, W. Xu, M. Gruteser, W. Trappe, and I. Seskar, "Security and privacy vulnerabilities of incar wireless networks: A tire pressure monitoring system case study," in *Proc. 19th USENIX Conf. Security*, Berkeley, CA, USA, 2010, pp. 21–21.

[2] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, and T. Kohno, "Comprehensive experimental analyses of automotive attack surfaces," in *Proc. 20th USENIX Conf. Security*, Berkeley, CA, USA, 2011, pp. 6–6.

[3] F. Stajano, *Security for Ubiquitous Computing*. Hoboken, NJ, USA: Wile, 2002.

[4] A. Dardanelli, M. Tanelli, B. Picasso, S. Savaresi, O. di Tanna, and M. Santucci, "A smartphone-in-the-loop active state-of-charge manager for electric vehicles," *IEEE ASME Trans. Mechatron.*, vol. 17, no. 3, pp. 454–463, 2012.

[5] C. Spelta, V. Manzoni, A. Corti, A. Goggi, and S. M. Savaresi, "Smartphone-based vehicle-to-driver/environment interaction system for motorcycles," *IEEE Embed. Systems Lett.*, vol. 2, no. 2, pp. 39–42, Jun. 2010.

[6] A. Dardanelli, M. Tanelli, and S. M. Savaresi, "Active energy management of electric vehicles with cartographic data," presented at the 2012 IEEE Int. Electr. Veh. Conf., 2012.

[7] Microchip Technology Inc., 16-bit dsPIC® Digital Signal Controllers.

[8] NIST Special Publication 800-121 Revision 1, Guide to Bluetooth Security: Recommendations of the National Institue of Standards and Technology 2012.

[9] C. Hager and S. Midkiff, "Demonstrating vulnerabilities in bluetooth security," in *Proc. IEEE Global Telecommun. Conf. (GLOBECOM'03)*, 2003, vol. 3, pp. 1420–1424.

[10] K. Haataja and P. Toivanen, "Two practical man-in-the-middle attacks on bluetooth secure simple pairing and countermeasures," *IEEE Trans. Wireless Commun.* vol. 9, no. 1, pp. 384–392, Jan. 2010 [Online]. Available: http://dx.doi.org/10.1109/TWC.2010.01.090935

[11] NIST Special Publication 800-56A, Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography.

[12] FIPS-186-3, Digital Signature Standard (DSS). NIST 2009.

[13] FIPS-197, Advanced Encryption Standard (AES). NIST 2001.

[14] NIST Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation - Methods and Techniques 2001.

[15] A. Bogdanov *et al.*, "PRESENT-An ultra-lightweight block cipher," in *Proc. Int. Workshop Cryptographic Hardware Embed. Syst. (CHES)*, 2007, pp. 450–466, ser. LNCS, no. 4727 Springer.