Car-to-Smartphone Interactions: Experimental Setup, Risk Analysis and Security Technologies

Bogdan Groza, Horatiu Gurban, Lucian Popa, Adriana Berdich, Pal-Stefan Murvay
Faculty of Automatics and Computers, Politehnica University of Timisoara, Romania
Email: {*bogdan.groza, eugen.gurban, lucian.popa, adriana.berdich, pal-stefan.murvay* }*@aut.upt.ro*

***Abstract*—Vehicle access control and in particular access to in-vehicle functionalities from smart mobile devices, e.g., phones or watches, has become an increasingly relevant topic. Security plays a critical part, due to both a long history of car keys that succumbed to attacks and recently reported intrusions that use various vehicle communication interfaces to further gain access to in-vehicle safety-critical components. In this work we discuss existing technologies and functionalities that should be embedded in an experimental setup that addresses such a scenario. We make emphasis on existing cryptographic technologies, from symmetric to asymmetric primitives, identity-based cryptography and group signatures. We also discuss risks associated with in-vehicle functionalities and mitigation, e.g., intrusion detection systems.**

I. Introduction and motivation

In-vehicle components and networks are somewhat unprepared for malicious adversaries as recent investigations have proved, e.g., [5]. Fortunately, the industry moves quickly to adopt security technologies in recent standards, e.g., AUTOSAR, and the academic community has also reacted with numerous research proposals for in-vehicle security. Car access, traditionally mediated by mechanical keys, is a particular area of focus due to the numerous reported attacks, e.g., [10]. The pervasiveness of smart mobile devices, e.g., smartphones, smartwatches, makes them suitable candidates for replacing traditional keys. Various attempts already exist, e.g., [4]. The adoption of smartphones is also encouraged by the adoption of Android inside car units which may further ease the deployment of such access control application.

In this work we discuss about the development of an experimental setup for car-to-smartphone interactions. Figure 1 presents the experimental setup from our past project CSEAMAN [6] on top of which we plan to deploy a car access system based on smartphones. Our previous model was mostly focused on in-vehicle subsystems, e.g., body control module (BCM), instrument cluster, and vehicle networks, e.g., CAN, FlexRay. The schematic of the new setup is depicted in Figure 2. In the current setup we introduce smartphones that connect to in-vehicle units. More discussion on components follow in a later section. There are several points that need to be covered by a car-to-smartphone interaction setup. We enumerate only a few that we find to be more relevant:

- analyzing existing functionalities in terms of both *benefits and risks*, since security is a trade-off, a clear understanding of the associated risks is necessary (remote control for functionalities with high degree of risk may be unwise),
- a good *selection of components* that are representative for real-world in-vehicle electronic control units (ECU) and communication interfaces is relevant since vehicles are highly heterogeneous having almost anything from low-end 8-bit ECUs to high-end 32-bit ECUs with multiple cores and low speed networks such as LIN to high speed CAN-FD, FlexRay or even BroadRReach (an Ethernet based bus),
- *security enforcing technologies* such as NFC cards and TPM (Trusted Platform Module) are needed since relying on software security alone may open easier roads for attack (hardware support has its own shortcomings as flaws/backdoors may exist and are harder to detect/patch but it adds an additional layer of security),
- correct use of *security services*, from regular symmetric and asymmetric cryptographic primitives, to more advanced functionalities, e.g., identity-based cryptography, as well as the correct tool-set for protocol verification, e.g., model-checkers and/or penetration testing tools, intrusion detection mechanisms, etc.

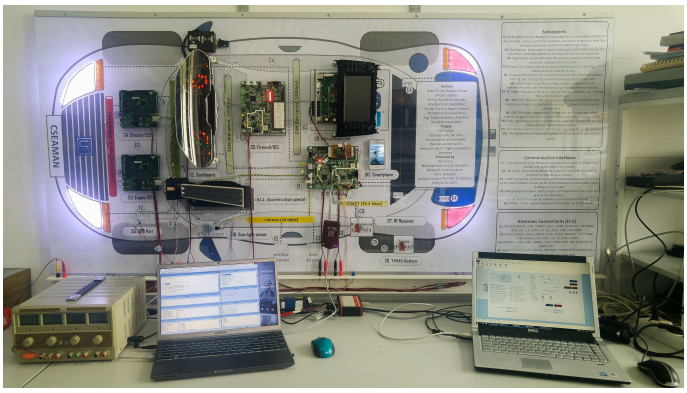

Figure 1. Experimental setup of our previous project CSEAMAN

In the following sections, our discussion tries to focus around these lines. First we give a brief overview on

existing applications and risk analysis then we discuss on components, security countermeasures and technologies.

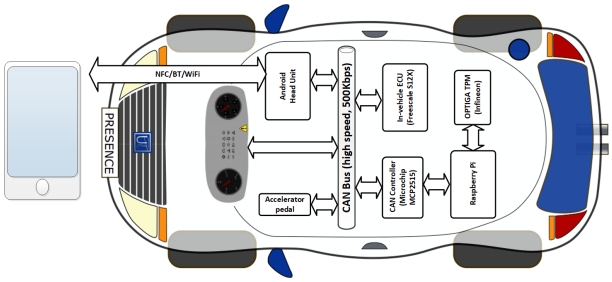

Figure 2. Schematic of the proposed setup for PRESENCE

II. Smartphone apps and risk analysis

Beside improved safety and performance, newer vehicles provide better comfort functions for the driver and passengers and an improved user interaction. Configuring all the new features, obtaining visual feedback from the vehicle subsystems can no longer be sustained by classic instrument clusters with analog styled gauges and dashboard warning lights or center consoles with traditional buttons. For this reason, many producers already offer interfaces for accessing cars via mobile devices. Table I summarizes some of the remote commands provided by the car manufacturer in the official smartphone app (this is just a short summary based on information that we could retrieve from the Internet since we cannot have direct access to all these cars). In this table we considered the following applications: MyOpel (1), MyChevrolet (2), Tesla (3), BMW Connected (4), Audi MMI Connect (5), Mercedes me (6), Volkswagen Car-Net Security Service (7), Volvo On Call (8) and Toyota Remote Connect (9). The functionalities provided are very diverse, from media control features to keyless start of the engine. Smartphone apps are also good candidates for remote enabling self-driving functionalities. One example is Tesla's enhanced summon feature where the Tesla app is used to send a remote request to the car to navigate autonomously from the parking place to the driver's location.

But all these functionalities may carry safety and security risks. Consequently, a risk assessment is necessary for identifying the most critical ones. Risk assessments methodologies are well known and have been already applied to automotive scenarios, e.g., [2]. Such methodologies can be used to rate and rank the risk associated with these functionalities. The risk has two main factors, the impact and the difficulty of carrying out the attacks. The impact can be further refined along three terms that take into account safety I_{Sf}, financial cost I_{Fin} and operational aspects I_{Op}. For each term five classes can be defined and rated 0 to 4. Safety can be ranked as: 0-none, 1-light, 2-severe, 3-life threatening, 4-fatal, all these accounting for the impact on the safety of the driver, other car occupants and traffic participants. The financial term is ranked according to the associated financial lost: 0-none, 1-10$, 2-100$, 3-1000$, 4-10000$. Operational impact ranks across the alteration of the default behavior: 0-no impact, 1-indiscernible operational impact, 2-discernible operational impact with insignificant performance degradation, 3-noticeable impact, 4- significant impact for the driver and other traffic participants. Because we consider the safety aspects being the most important followed by the financial and operational ones, for each term weight factors are employed as in [7]: $\alpha_{Sf} = 8$, $\alpha_{Fin} = 4$ and $\alpha_{Op} = 2$. The impact becomes a weighted sum of the previously described three terms: $\mathbb{I} = \alpha_{Sf} I_{Sf} + \alpha_{Fin} I_{Fin} + \alpha_{Op} I_{Op}$. On the other hand the difficulty of the attack quantifies the time, expertise, inside knowledge, window of opportunity and equipment necessary for carrying out the attack. The risk is computed as the product between the impact and difficulty inverse: $\mathbb{R} = \mathbb{I} \times \mathbb{D}^{-1}$.

This methodology can be applied to any of the previously mentioned functionalities. For example, the app feature designed to remotely start and stop the engine can have serious consequences if an adversary can trigger it while the vehicle is at speed in an overtaking maneuver. From the safety perspective it could lead to fatal injuries, from the financial one to more than 10.000$ loses, and finally from the operational perspective it has a significant impact for the driver and also for other traffic participants because the power steering and power brakes will be disabled and in some cases even the steering wheel can be locked. In this case $\mathbb{I} = 8 \times 4 + 4 \times 4 + 2 \times 4 = 56$. If the same functionality cannot be triggered while the vehicle is running, then it is likely that it may only become annoying for the driver (assuming it is triggered by an adversary). There should be no significant injuries, little financial costs and a clear alteration of behaviour. This leads to $\mathbb{I} = 8 \times 1 + 4 \times 1 + 2 \times 3 = 18$ which is a reduced impact.

Table I
Functionalities in some smartphone car apps

Functionality	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
lock/unlock car	✓	✓	✓	✓	✓	✓	✓	✓	✓
keyless start	–	–	✓	–	–	–	–	–	–
horn	✓	✓	–	✓	–	–	✓	✓	–
flash the lights	✓	✓	✓	✓	–	–	✓	✓	–
start/stop engine	–	✓	✓	–	–	–	–	✓	✓
seat heaters	–	–	✓	–	–	–	–	✓	–
HVAC on/off	–	–	✓	✓	–	✓	✓	✓	–
HVAC temp.	–	–	✓	–	–	–	✓	–	–
windshield defroster	–	–	–	–	–	–	✓	–	–
vent/close sunroof	–	–	✓	–	–	–	–	–	–
enable/disable valet mode	–	–	✓	–	–	–	–	–	–
media control	–	–	✓	–	–	–	–	–	–

III. Interfaces, components and libraries

Car components. Our experimental setup uses a traditional-style instrument cluster that can be seen in Figure

1. The instrument cluster is interconnected with the other car ECUs by using a low speed CAN interface. All the interfaces elements on the instrument cluster, dashboard lights and gauges, are controlled by their corresponding CAN signals. In addition to this, our setup also has an accelerator pedal that communicates over CAN. We discuss later why these components are relevant in the interaction with smartphones. As in-vehicle ECU we use NXP's S12X platform. This is built around a main 16-bit core with an additional XGATE CPU dedicated to reducing the load of the main CPU by serving interrupts. The S12X family covers a wide range of configurations enabling various communication options such as SPI, I2C, CAN, LIN or FlexRay. Its simple programming model, varied memory options and communication abilities make it suitable for various automotive applications for the body, chassis or power-train functional domains.

Android smartphones. The Android OS is now used in car head-units and smartphones that are frequently used to communicate with the car to access different functionalities from a car. Smartphones usually offer three types of communication that can be easily used to interact with the car: Bluetooth, NFC, and WiFi. Besides classical car key functionalities, smartphones can have even more appealing features. For example, our experimental setup has an accelerator pedal which is periodically checked by in-car diagnosis and monitoring systems to signal potential malfunctions. Indeed the accelerator pedal is a safety-critical component and its failure may lead to catastrophic accidents. Checking if the pedal is working can be done by using a MATLAB model of the accelerator pedal which verifies the range (whether measured sensor voltage is too high or too low) and the plausibility of sensor values (sensors are out of range). Such diagnosis is commonly done by manufacturers in environments such as MATLAB. Thanks to MATLAB's availability on smartphones, we can at least imagine that this can be ported to user's phone or imagine any other MATLAB testing functionality that is used by manufacturers. The MATLAB mobile apps usually work by connecting to the MathWorks Cloud, store the files on the MATLAB Drive and edit/run the scripts from the MATLAB Mobile command line on the MathWorks Cloud. Another way is to connect the MATLAB Mobile application to a remote computer with MATLAB installed. If the computer and the smartphone are on the same network we can access the files on the computer from MATLAB Mobile and we can edit/run the scripts from the MATLAB Mobile command line on the computer. The main limitation of the MATLAB Mobile application for our work is that it does not offer support to open or to create a graphical user interface similar to traditional smartphone apps. But this can be solved since we can use the MATLAB Compiler SDK to generate a Java Package (.jar file), e.g., based on the MATLAB files with the accelerator pedal sensor diagnosis model. The generated .jar file can be easy imported in Android Studio where we can draw a graphical user interface.

Other components. We found that other components, which are not necessary automotive-grade, such as the Raspberry Pi are very handy to use in such a setup. The Raspberry Pi 3 is a computer board capable of running various distributions of Linux, e.g., Fedora, Arch and also the producer's official supported Raspbian operating system (based on Debian) which gives it the benefits of having easy-access to all the Linux tools and high capacity of adaptation to numerous open-source projects. The Raspberry Pi 3 microprocessor does not have an embedded CAN controller. For this reason we chose to use an additional SPI connected board that encapsulates a CAN controller (Microchip MCP2515) and a CAN transceiver (NXP TJA1050). For enabling CAN communication on Linux the SocketCAN API provides the necessary functions. Raspberry Pi can communicate on Bluetooth or Wi-Fi using its on-board capabilities for wireless networks. In order to secure the software operations done by the Raspberry Pi using a hardware component, it can be integrated with a TPM board in the same application environment with the prerequisite of rebuilding and using a modified kernel of Raspbian with support for TPM 2.0 and configuration for the desired trusted platform module. For this purpose we used the OPTIGA SLB 9670 TPM which is a board with SPI interface produced by Infineon capable of performing all TPM 2.0 operations which are specified by the Trusted Computing Group. It can be directly attached on the header pins of the Raspberry Pi 3 being supplied with 3.3V and being able to receive or transmit data using SPI. So it is easy to use in an application with the Raspberry Pi if its OS has support for the board and for TPM 2.0 operations. As a security background for the Infineon Optiga TPM, it is validated according to FIPS and is used by wolfSSL Inc. to test the wolfTPM software module.

IV. Security technologies

Traditional car keys were built around symmetric cryptography. This requires a shared secret key and building advanced functionalities such as rights delegation may be less secure if the same secret key is shared between multiple devices. Also, symmetric primitives cannot assure non-repudiation which allows a malicious device to delegate its rights to other devices and this will be harder to trace. Even in the simpler symmetric key setup with basic functionalities, previous research has shown lots of flaws, e.g., the use of poor randomness and of the insecure stream cipher HITAG [10].

In contrast, public key primitives can be used to delegate rights and assure non-repudiation. Still, this requires the existence of public-key certificates and of a hierarchy of trust. Identity-based cryptographic primitives may help in this respect as they can be used without certificates since the name of a principal is sufficient to derive its public-key. Going even further, *group signatures* can assure anonymity

since the signature cannot be traced to a particular signer in the group. A trusted third party can still trace the original signer in case that a dispute arises. Symmetric functions are fast and have modest memory requirements. In contrast, public-key primitives require more computations and memory. Identity-based cryptosystems come close to regular public-key primitives and group signatures have even higher requirements. Some research proposals have used even more advanced cryptographic functionalities such as secure-multiparty computation, e.g., [9]. Secure multi-party computation can be further used to assure privacy by processing over encrypted data whenever access is mediated via an untrusted remote server.

We also note that not all of the proposals that we found (for gaining car access via smartphones) included formal arguments on security. One easy way to obtain a proof of security would be to use a model checker. We find that ProVerif [3] and AVIPSA [1] are the most convenient to use. One particular problem is that there may be no support for modelling non-standard primitives such as identity-based schemes or group signatures. Such model-checkers may not be able to model mathematical properties that are essential to the security of some of the proposed protocols.

Intrusion detection systems are one particular technology that so far seems not to have been ported on car-to-phone communication, and in particular car access control. Given the wide availability of wireless hacking tools, e.g., HackRF [1], it is likely that attacks will become even more common. Various attack strategies using HackRF are discussed in [8]. One particular research direction may be detecting intrusions in case of smartphone-to-car interactions and in particular in car access via smartphones.

V. Conclusion

In this work we tried to give a brief overview of technologies, tools and risks associated to vehicle-to-smartphone interactions. Without a doubt, cryptography offers a rich toolset for securing access to future cars and smart mobile devices are a good vector for carrying such technologies. These devices open the road for many new applications and functionalities starting from car access up to real-time car diagnosis, but it may clearly open the road to new attacks. Further research and cooperation between industry and academics will clearly contribute to make cars safer and mitigate future attacks. An experimental setup is a playground from which lessons could be learned, it is useful both as a research and educational tool.

Acknowledgement. This work was supported by a grant of the Romanian National Authority for Scientific Research and Innovation, CNCS-UEFISCDI, project number PN-III-P1-1.1.-TE-2016-1317 (2018-2020) http://www.aut.upt.ro/~bgroza/projects/presence/index.html.

[1] https://greatscottgadgets.com/hackrf/

References

[1] A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, Cuéllar, et al. The AVISPA tool for the automated validation of internet security protocols and applications. In *International conference on computer aided verification*, pages 281–285. Springer, 2005.

[2] L. ben Othmane, R. Ranchal, R. Fernando, B. Bhargava, and E. Bodden. Incorporating attacker capabilities in risk estimation and mitigation. *Computers & Security*, 2015.

[3] B. Blanchet. An efficient cryptographic protocol verifier based on prolog rules. In *Proceedings. 14th IEEE Computer Security Foundations Workshop, 2001.*, pages 82–96. IEEE, 2001.

[4] C. Busold, A. Taha, C. Wachsmann, A. Dmitrienko, H. Seudié, M. Sobhani, and A.-R. Sadeghi. Smart keys for cyber-cars: secure smartphone-based nfc-enabled car immobilizer. In *Proceedings of the third ACM conference on Data and application security and privacy*, pages 233–242. ACM, 2013.

[5] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, T. Kohno, et al. Comprehensive experimental analyses of automotive attack surfaces. In *USENIX Security Symposium*. San Francisco, 2011.

[6] B. Groza, H. Gurban, and P.-S. Murvay. An experimental model for in-vehicle networks and subsystems. In *VEHITS*, pages 326–331, 2017.

[7] E. H. Gurban, B. Groza, and P.-S. Murvay. Risk assessment and security countermeasures for vehicular instrument clusters. In *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, pages 223–230. IEEE, 2018.

[8] O. A. Ibrahim, A. M. Hussain, G. Oligeri, and R. Di Pietro. Key is in the air: Hacking remote keyless entry systems. In *Security and Safety Interplay of Intelligent Software Systems*, pages 125–132. Springer, 2018.

[9] I. Symeonidis, A. Aly, M. A. Mustafa, B. Mennink, S. Dhooghe, and B. Preneel. Sepcar: A secure and privacy-enhancing protocol for car access provision. In *European Symposium on Research in Computer Security*, pages 475–493. Springer, 2017.

[10] R. Verdult, F. D. Garcia, and J. Balasch. Gone in 360 seconds: Hijacking with hitag2. In *Proceedings of the 21st USENIX conference on Security symposium*, pages 37–37. USENIX Association, 2012.