Automotive Proxy-Based Security Architecture for CE Device Integration

Alexandre Bouard[1], Johannes Schanda[2], Daniel Herrscher[1], and Claudia Eckert[3]

[1] BMW Forschung und Technik GmbH, D-80788 Munich, Germany
alexandre.bouard,daniel.herrscher@bmw.de
[2] itestra GmbH, D-80796 Munich, Germany
schanda@itestra.de
[3] Technische Universität München, D-85748 Garching, Germany
claudia.eckert@in.tum.de

Abstract. Increasing adoption of Consumer Electronic (CE) devices in the automotive world encourages car makers to propose new CE-related features each year. However, car complexity and security concerns slow down this process. The ubiquitous and personal nature of such devices represents a real threat for car IT systems. We believe that the arrival of IP standards in car should solve most of these issues. In this paper, we describe a proxy-based security architecture for an on-board IP-based network allowing deep and total integration of external mobile wireless services. The proposed architecture has been integrated in an automotive IP-based communication middleware and supports security mechanisms complying with the highly demanding automotive requirements.

Keywords: Security, Access Control, Middleware, Data Labelling, CE Device, Mobile Device, Automotive Application, Car-to-X Communication.

1 Introduction

Consumer electronics (CE) devices like smartphones or tablets have become more and more powerful and ubiquitous. New use cases, unimaginable a decade ago, appear everyday. A few years ago automotive manufacturers started to propose numerous on-board services directly accessible from the CE device. Music, navigation, phone calls, car status... the applications are various and connected through a plethora of interfaces such as USB, Bluetooth or GSM. But albeit numerous, the applications stayed similar and are only developed by the automotive manufacturers themselves or partner companies; security concerns and high system complexity slow down the release of new CE device accessible functions.

Originally conceived to develop IP-based solutions for automotive on-board communications, the SEIS project [1] aims at reducing the complexity of the network infrastructure and at designing a suitable security layer for both middleware and applications. We believe that the introduction of IP standards in cars will considerably simplify the integration of mobile CE devices and provide the security level needed to make on-board functions available from any external communication partner.

In this paper, we describe an automotive architecture for holistic CE device security while addressing shortcomings of traditional automotive security. We propose on-board distributed mechanisms for cooperative security evaluation and enforcement complying with the highly demanding automotive requirements. In addition, a first in-car implementation of the proposed security concepts is available for ETCH [2], an open-source middleware that we are currently extending for automotive and mobile platform use.

The remainder of this paper is organized as follows. Section 2 provides some background information on traditional and next generation automotive security and threats related to the integration of CE devices. Afterwards, we present our proxy-based security architecture in Section 3 and 4. In Section 5 we describe our prototype implementation. Section 6 discusses the advantages and disadvantages of this architecture and future work. Finally Section 7 provides a conclusion.

2 Scope and Related Work

In this section we provide background information on automotive networks and security. We explain the threats related to the introduction of CE device applications in cars. Then, we present the security requirements and attacker model considered by this work.

2.1 Automotive Network and Security

During the last decade, the car has become a very complex distributed system; a premium vehicle can include up to 70 electronic control units (ECUs) interlinked by at least 5 networking technologies using complex application gateways. We believe that future use cases will involve more resources, more inter-ECU communication and more external communication partners. As mentioned in the introduction, the SEIS project proposes to alleviate the forthcoming functional issues with the development of IP solutions for automotive middleware. But the use of IP standards is not without risk. The underlying protocols and systems are well-known standards and attacks could be potentially directly applicable to the car.

Recently, work has highlighted numerous security issues of automotive infrastructure, such as the lack of encryption and authentication of controller area network (CAN) protocols [3,4] or weaknesses at the ECU level [5]. Aware of these problems, some projects aimed at providing long-term security solutions and proposed security architecture [6,7], but didn't consider the security requirements of integrating external services. The SEIS project proposes security solutions for application and IP-based communication. In Section 4 we outline their approach and use it as basis for our CE device adapted security architecture.

Today, automotive CE integration is provided via three different technologies:

- Bluetooth: Standard methods allow communication encryption and PIN-based authentication, but the features used in-car are generally limited and not security critical, e.g. phone book, phone call functions, audio system, etc.
- GSM/3G: Communications from the CE device are routed through a back-end server acting like a firewall and delivering to the car only authorized and valid function calls. Access to some critical functions is possible (e.g door locking).
- Wired (USB) interface: authorized applications establish a secure communication channel and are equipped with a certificate from the car manufacturer defining rights for a pre-defined set of infotainment and car status functions.

Industry projects about Car-to-X (C2X) security [8] already propose security solutions for on-board C2X communication platforms and protocol standardisation. Academic works [9,10] have designed car-to-car security and privacy communication protocols, but in each case the focus was on communication and authentication at the edge of the on-board network. Few consider risk analysis and authorization management. They generally propose access control list (ACL) and firewall based systems at the network entry points, but these solutions don't provide the scalability and flexibility in a reliable and efficient way that is required for new services from untrustworthy CE devices or developed by a third-party.

2.2 CE Device Related Threats

The short life cycle of CE devices [11] and their increasing power might soon allow automotive manufacturers to consider these mobile devices as virtual automotive software/hardware upgrades [12] for new applications running on the CE device and potentially communicating with any internal automotive service.

User-installable applications allow customization of CE devices. This can generate substantial risks. Malicious applications can corrupt a valid automotive application running on the CE device, directly send exploits, worms or viruses to the car and leak sensitive information provided by the car. Weak security configurations (e.g. weak password, no or misconfigured security software) can increase the risk of corruption of a CE device. Attackers can take advantage of this and, for example, steal secret authentication credentials directly from the device (password, keys). Additionally, CE devices communicate over untrusted wired or wireless channels, where messages can be listened to, intercepted, altered, injected and replayed, facilitating attacks aimed at impersonating an authorized CE device or at invoking a function with an illegal input.

In other terms, integrating CE devices is challenging; the car has to enforce appropriate security mechanisms that are both adapted to the capabilities of the CE device and its operating system (OS) and additionally provide safe and secure access to in-car functions and data.

To circumvent some of these problems, mobile OSes provide libraries implementing secure communication protocols, strong authorization and isolation mechanisms, which are reliable as long as the device is well-configured and not rooted or jailbroken. For Android, academic work proposes additional solutions such as taint tracking[13], virtualization [14], behavioral analysis[15], enforcement of mandatory access control[16] or analysis of remote duplicates[17]. These approaches mostly concern internal CE device security and are not applicable to our automotive use cases. Promising work about remote attestation for mobile devices has been published [18,19], but is based on trusted hardware which is still far from reaching mass production.

Interesting approaches may reside in secure integration of mobile devices for corporate networks [20,21]. But such solutions mostly include heavy security protocols like IPSec, not suitable for mobile device purposes [22], and rely on integrity measurements provided by hardware based security mechanisms. Additionally these approaches only regulate network access and usually lack specifications for internal function calls and data handling.

2.3 Attacker Model

As attack surface we define every communication channel present in current and future vehicles that is potentially usable by CE devices for short (USB, Wi-Fi, Bluetooth) and long range communication (GSM, UMTS, LTE) able to carry IP packets. We consider as out of scope the systems for key-entry, radio-channels and other addressable channels (emergency calls, remote diagnostics).

As attacker, we define a person having physical access to the car or being able to contact any remote interface of the car and wanting to use an in-car function in an unauthorized way. Regarding her technical capabilities, we consider that she has good knowledge of the system (standard protocols, open-source technologies) but that her computation capacities are limited (no possible brute force attack of a large encryption key). However, she can compromise or steal an authorized CE device. She has no time limit for her attack, but we assume that she has no physical contact with the on-board network or ECUs (no addition or extraction of in car components).

2.4 Automotive Security Requirements for CE Integration

With regard to the security risks and challenges described above, we define the following security requirements for CE integration.

Functional Requirements: The system should provide scalability and performance, essential goals when dealing with safety-relevant use cases. Additionally, it should provide good usability and limit the system complexity both for system development and for the end-user.

Communication Requirements: Considering the untrustworthiness of external communication networks, the protocols used should enforce mutual authentication to provide proof of origin for both car and CE device, data confidentiality due to encryption methods to not disclose sensitive information, and finally data integrity to avoid unauthorized tampering with the messages during their travel.

Car Requirements: The introduction of external services should not disturb or compromise the car runtime and its integrity, therefore strong isolation techniques should be used. The car should be able to judge the health of the CE device and enforce strong access control based on authentication mechanisms and reliable authorization management based on available context information. Finally, the car should assure the continuous operation of its internal services, even while using strong security mechanisms or under attack.

3 SEIS and Car-to-CE (C2CE) Communication

As mentioned in the introduction, one of the goals of the SEIS project is to develop an automotive middleware for IP-based communication in the car. The middleware, by definition, provides abstracted interfaces and hides the network complexity. In addition, it can automate the security management with an appropriate security configuration.

3.1 SEIS Security Middleware for On-board Network

Figure 1 presents the modularization adopted by SEIS for a three-layer security framework [23]. Such an architecture offers enough adaptivity to comply with highly demanding requirements. The first layer provides security decisions by means of static policies governing authorized on-board communications and application access controls (Policy Management) and by monitoring the reaction of the system both at the ECU and network level (Intrusion detection). The two remaining layers are in charge of security enforcement (e.g. protocol implementation and filtering). The bottom layer, Key management and Cryptographic service Management, may be included in the ECU hardware. Such a configuration allows an additional hardware-based protection for cryptographic keys and platform integrity.

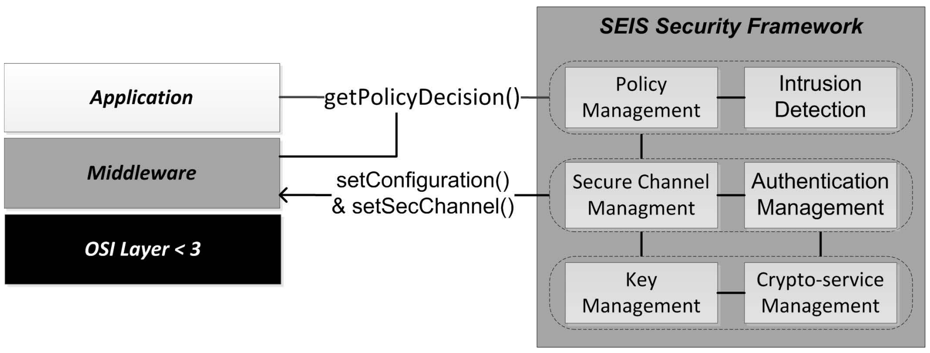

Fig. 1. SEIS Security Framework for internal IP-based communication

The automotive system is subject to drastic safety and performance requirements and generally can't afford the latency and the risk of errors induced by complex security mechanisms. Most of the configuration of the framework is therefore set up statically during vehicle assembly or during periodic system updates. This encompasses security policies and setup of security associations for on-board IPSec communication channels between ECUs.

3.2 Towards Secure Automotive Proxy-Middleware

The static automotive configuration requires a new communication infrastructure when dealing with external partners. CE devices are heterogeneous and their capabilities depend on several factors, e.g. their OS and hardware. Car manufacturers cannot restrict the C2CE connectivity to a certain class of device; therefore the architecture needs to be adaptive, as does the underlying security. The car has to be able to integrate external CE-based services communicating over a wide range of media and communication protocols (e.g. automotive middleware- or web services-based). At the same time the car has to limit its system complexity. We propose the use of a communication proxy, an entity in charge of managing access between on-board and external networks. The proxy will realize a protocol decoupling, allowing flexibility for outside communication and optimal security solutions on the inside. This approach is contrary to most corporate network solutions, where mobile devices need to provide a strict security configuration in order to be considered as an internal entity and directly access internal resources.

Security for the proxy is essential and requires a new dynamic policy engine, authentication schemes and an intrusion detection system. The protocol decoupling makes internal ECUs context-unaware and forces the proxy to enforce security for both inbound and outbound messages at the edge of the internal network. On the other hand, the introduction of IP over Ethernet as internal communication standard allowing bigger bandwidth than today's networks will raise the complexity of the software components and of the exchanged objects, e.g. processing of object models for radar environment perception or download of high resolution maps for micro navigation from different external sources. The verification of both security requirements and packet validity for every message of each external communication partner will be impossible at the proxy level alone. We propose to share the security enforcement between proxy and ECU: the proxy provides external security protocols and supports the ECU in enforcing security for applications and resources. More details about our proposed architecture are given in the next section.

4 Security for CE Adaptive Communication Proxy

In this section, we explain the concept of "Proxy-ECU Cooperation" previously motivated and present our CE adaptive security architecture for Proxy and ECU.

4.1 Proxy-ECU Cooperation for C2CE Security

Security for CE device integration aims to prevent malicious disturbance of the automotive systems and to control the release and propagation of data related to CE devices. The "Cooperation" concept is about enforcing information flow labeling between proxy and ECU in order to avoid system corruption. We define "Security Level" (SL) as a formal security description of an information flow coming from and to a CE device. It may include information about the CE device, communication link and other security requirements that car and CE device fulfill or have to fulfill. The SL is included like a tag in every CE device related on-board message.

Information flow control is essential. ECUs internally exchange genuine messages and therefore only necessitate secure channels and simple access control mechanisms. External messages can not be dealt with in the same manner, inbound and outbound messages need to be tracked because they can harm the system and disclose private or secret information. A good SL definition should provide appropriate expressiveness of the CE device communication situation and be efficiently transmittable and interpretable. Figure 2 shows the life cycle of SL tags. We differentiate two types of SL: the SL_{CE} generated by the proxy and the SL_{ECU} provided by applications on the ECU.

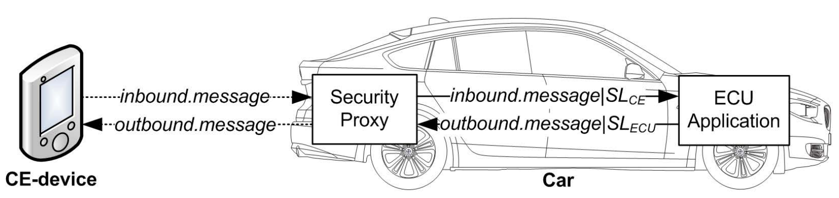

Fig. 2. Cooperative data tagging between proxy and ECU

The SL_{CE} describes the contamination risk presented by the data and the CE device security exposure, it includes information about the device, its state of health and about the security present on the communication link. Derived from a continuous reevaluation of the CE device security context, this tag is transmitted from the proxy to the ECU middleware, which adapts the message treatment and policy decision in consequence. The security mechanisms induced after interpretation of the SL allow reducing the security risk to an acceptable level in order to be passed to the application.

On the other hand, the SL_{ECU} or secrecy tag characterizes the risk of privacy infringement or industrial secret disclosure and concerns outbound messages only. Supplied by the ECU, this tag includes the user or class of user allowed to receive this data, as well as the security requirements that the user and the communication link have to fulfill. To release the message, the proxy verifies if the tag matches the concerned CE device.

The SL metric defines qualitative levels obtained from quantitative security parameters. The metric maps abstract security concepts and requirements to concrete protocols and mechanisms (cf. Section 5.2). The quantitative part of the metric is easily updated, because it is located only in the proxy and follows the evolution of the security techniques during the car's lifetime. Our system follows strict mandatory access control. The CE device adaption is supported by use case adapted security engines providing the testing and non-harmfulness verification of inbound and outbound messages.

4.2 CE Adaptive Security Architecture

Figure 3 presents a more concrete view of the proxy and ECU infrastructure. They both rely on a secure middleware like the one mentioned in Section 3 to establish secure communication channels. For more flexibility and independence, the proxy has its own C2CE Authentication Manager, adapted to store and verify security credentials of CE devices (password or certificates). Policy decisions from the C2CE Policy Manager and protocol decoupling are enforced in the Secure network access (SNA) module. Inbound messages are authorized based on the accessed domain (e.g. for infotainment, driving assistance...) grouping several ECUs, whereas outbound messages are released after verification of the SL_{ECU}. Secure Proxy Middleware and C2CE SL Evaluator are in charge of the tag management.

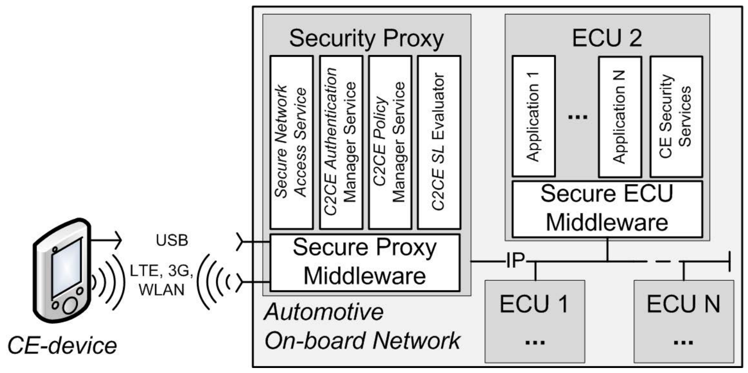

Fig. 3. CE adaptive security architecture

After reception of an SL_{CE} tagged message, the ECU middleware extracts the tag and decides whether the security information contained by the tag is sufficient to allow an appropriate access control or whether, based on the tag, the incoming data requires specific security treatment in order to be processed. In case of a complex function receiving a critical object as argument, like executable code, from a CE device which does not qualify for complete trust, the middleware can invoke specific CE Security Services for data "decontamination". Like a quarantine zone, the decontamination services perform some tests in an isolated part of the system and allow avoiding or detecting potential application corruption by verifying the data's non-harmfulness (e.g. syntax check, execution in a virtualized environment). These tests are adapted for each use case depending on its requirements. Additionally, the tag can help to prevent the waste of ECU resources, for example by verifying before processing the data if the CE device is authorized to get a response potentially containing sensitive information.

The SL_{ECU} are managed by the middleware, they are first statically defined at compilation time and can later evolve according to authorization and intrusion detection based policies. We previously said that the SL_{CE} can help to enforce a control on the ECU output. However in certain cases where the ECU isn't aware of a "fresh" SL_{CE}, the addition of a new tag, the SL_{ECU} may be necessary, either because the ECU instantiates the communication or because the packet might get forwarded outside without its knowledge e.g. in case of publish/subscribe services where the ECU is a publisher. Additionally, like the inbound message case, security engines adapted for outbound messages are supported by CE Security Services. They enforce data anonymization when possible and may even exclude critical message sections in order to prevent unauthorized disclosure of sensitive information.

5 Prototype Implementation

In order to evaluate our concepts, we set up two realistic scenarios for CE devices integration. The first use case, called "Social Flight Mode", provides a way to release private information. An on-board application provides the CE device with a video stream of the front camera and a real-time instrument cluster. The second use case, called "Remote Window Control", concerns controlling internal automotive functions. The user opens and closes the four windows directly from her CE device. Access to the driver windows is subject to credential verification at the application level. Based on these use cases, we developed two applications running on an Android 3.2 tablet.

5.1 ETCH Security Tagging Service

As prototype basis for an automotive IP-based middleware we use the remote procedure call framework ETCH [2]. ETCH is an open-source software developed as an Apache Incubator project under the Apache 2.0 licence. It benefits from a modular architecture, offering efficient message serialization and flexibility to develop new security features. The prototype is implemented in Java. The proxy runs on a Windows PC and the ECU application/middleware on another Windows PC connected to the car CAN bus.

Figure 4 shows the architecture of the ETCH middleware. We included in the Transport Handler the capability to serialize and deserialize the SL tags from the ETCH packet. Additionally we customized this layer with some other features of the SEIS security framework presented in Section 3 for authorization

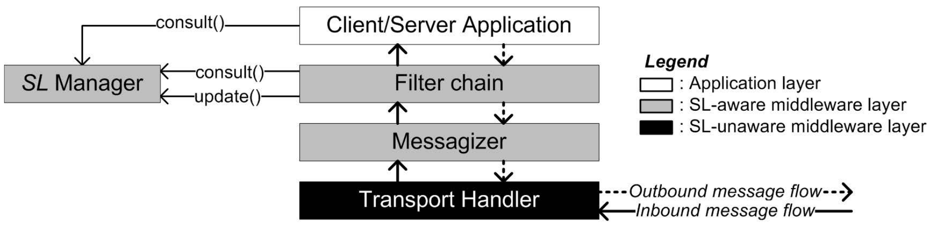

Fig. 4. ETCH architecture and SL tag management

and establishment of secure communication channels. The SL Manager stores SL_{ECU} and SL_{CE} in a hash table and provides support to enforce policies both at the application- and middleware-level, in the Filter Chain, a native module of ETCH that we adapted for our tag management. The Messagizer is in charge of dispatching the packet received from the Transport Handler to the right application and vice versa.

The security interface "consult()" for SL support of both application and middleware is motivated by complex applications willing to enforce a more granular access control. For example, the application controlling the four windows verifies specific CE device accreditation included in the tag for the driver window.

Regarding the definition of the SL_{CE}, we added the possibility to directly declare a vector representing the minimum SL in the declaration of the interface description language (IDL) before compilation of the service. This feature allows the application developer to potentially remain security-unaware.

5.2 Mirroring Proxy Middleware

We developed a mirroring service for communication protocol decoupling in the proxy. The Management Service in the proxy informs the CE device about which interface of the Mirror Service to contact. The Mirror Service provides sockets accessible from the CE device and a naming similar to the actual internal service. The CE device application has the impression of directly communicating with the internal service. For simplicity, the CE device supports the ETCH middleware and communicates over the ETCH protocol. The mirror service adds or extracts SL tags and enforces access control rules as mentioned in Section 4.1. The rules for SL_{CE} evaluation are defined within the mirror service. The tag consists of functional parameters describing the CE user (e.g. driver-, owner-status, ID...) and three security parameters, each of them evaluated with a four-level scale, describing the strength of the protocol encryption, its integrity and its authentication scheme.

5.3 Performance Overhead

In this section, we present an experimental evaluation of the performance impact of our proxy for communications between CE device and automotive application.

Table 1. Performance overhead of the ETCH-proxy

Configuration Decoupling	- *SL* Tag	- TLS	1)Throughput ([Call+Response]/s)	Penalty	2) Channel Establishment Latency (ms)
no	no	no	351	-	n/a
yes	no	no	336	4,3%	10
yes	yes	no	195	44,4%	15
yes	yes	yes	190	45,9%	45

We measure 1) the throughput of message processing for a simple service and 2) the latency resulting from the communication establishment between CE device and proxy in order to generate the mirroring services. For these experiments we deploy a simple case: an application on a CE device sends a function call message to an ECU service behind our ETCH proxy and receives a boolean as an answer. Our experiments are conducted on an Intel Core i7 2Ghz machine with 6GB RAM running Linux for the proxy and an Intel Core 2 Duo 2,4GHz with 4GB RAM running Windows XP for the ECU. The CE device is a Motorola Xoom with a Nvidia Tegra 2 chip. The CE device and the proxy are communicating over a 54 Mbit WLAN network, proxy and automotive application over a Gigabit Ethernet link. Function call and response messages are IP packets with a payload of 30 and 50 bytes respectively. In order to compare the middleware throughput and the communication establishment latency, we vary the following parameters:

– Communication Decoupling, the decoupling is enforced by the proxy, for the case "no decoupling" the proxy is replaced by a simple packet forwarder.
– *SL* Tag Evaluation, on top of decoupling the communication the proxy evaluates the *SL* tag and enforces adapted filter rules.
– External network security, the link between CE device and proxy is secured using the Transport Layer Security (TLS) protocol providing mutual authentication and data encryption.

Table 1 shows the average throughput for message processing (1) and the latency resulting from the communication establishment (2). This experiment does not produce much application processing; it mostly stresses the middleware and network mechanisms. In our set up the ETCH middleware and the communication decoupling decrease the throughput by 4,3%, with the evaluation of the security tag by 44,4%. The lower performance of the system is a consequence of the user and kernel context switching due to the network inputs/outputs. The process of encryption and decryption of the TLS protocol does not cause a visible performance loss when added to the decoupling and *SL* evaluation. The channel establishment latency results from the service discovery process and the generation of the mirroring services. Without any security feature enabled this process lasts 10 ms, with the *SL* tag evaluation 15 ms and with the TLS feature 45 ms. The first latency increase is caused by the context evaluation and the tag generation, the second one by the TLS authentication handshake. We believe that the overhead of our system becomes less significant for realistic and more complex automotive applications requiring more application processing.

6 Discussion and Future Work

In this section, we offer a brief evaluation of the security architecture, based on the requirements and threats defined in Section 2.

Functional requirements: Protocol decoupling offers several advantages. First the CE device application developer can chose the communication protocol. As long as the proxy provides an adapted translation plug-in, the car adapts its security levels in consequence. Second, internal communications can be run over a car-wide strong security solution like IPSec. However, due to potential heavy traffic (still insignificant in comparison to the volume of exchanged messages between ECUs) caused by numerous external communication partners, the proxy might become a bottleneck. Our prototype presents a throughput penalty of 44% with the security enabled, a value still allowing time demanding use cases and the possibility to have several simultaneous communication partners while maintaining quality of service. Though further investigations and tests need to be done, for example concerning external communications over different application or middleware protocols, e.g. HTTP, even if it would require a translation layer and induce more delay. Additionally, our implementation and benchmark were done in Java on powerful computers, more realistic scenarios would involve smaller ECU with an implementation in C code, but should not suffer from a considerable performance degradation [24].

Communication requirements: The difference in computing power between ECUs and their inability to dynamically adapt their security configuration motivated our choice of decoupling external communications. By trusting the security proxy's integrity and its CE device security evaluation, the ECU is able to make an adapted security decision. Further investigation needs to be done to precisely define the SL metric in order to provide a holistic security understanding of CE devices. Several use cases, such as software download and firmware update require end-to-end security solutions. For such cases, the proxy has to provide a secure tunnel; external entity and capable ECUs (e.g. Head Unit) negotiate the secure channel through this intermediary.

Car requirements: The ECUs rely on the proxy's integrity for delivery of a valid and accurate SL_{CE}. A potential attack would be to corrupt the proxy and tamper the tagging process. The malicious message would be handled with a lower security treatment and would have access to more functions. Our proxy is a single point of failure, therefore a security resistant architecture is necessary. Weyl et al. [6] propose a secure hardware architecture, which offers several advantages such as physical protection of encryption keys and secure boot. The second advantage, assuring proxy integrity, relies on hardware-based integrity measurements that can be performed only during the ECU boot, e.g. when the car starts. This solution couldn't therefore detect a corruption happening after boot. More promising approaches reside in isolation and monitoring techniques.

Technologies like hypervisor and microkernel allow a separation of the message treatment and the tagging process: Each CE device communication is treated by one isolation cell and can not interfere with its neighbor. Further investigations need to be done in order to determine the suitability of these concepts.

7 Conclusion

The customizable and non-regulated nature of CE devices raises several automotive security concerns. In this paper, we have presented a flexible security architecture aimed at mitigating this risk. We have proposed a design for an automotive security proxy enforcing the communication decoupling between internal and external networks. It allows the car to communicate over a wide range of protocols with the outside while internally keeping an optimal security protocol and limiting the increase of complexity of the inside. Our approach proposes CE adaptive security mechanisms relying on cooperation between a Security Proxy and ECUs, enabled by an in-band signaling protocol managed by the middleware. This architecture integrates various technologies to secure external communication and evaluate trust between CE device and car. The prototype of our architecture has been implemented and integrated in car and offers the performance required for automotive use cases. We are not aware of other research projects designing and implementing CE secure adaptive architecture for distributed systems with high functional requirements like cars.

Acknowledgments. The authors would like to thank Benjamin Weyl for valuable discussions about future on-board automotive security and the anonymous reviewers for their constructive comments.

Some of the research presented here, took place within the project SEIS - Security in Embedded IP-based Systems. The research project explores the usage of the Internet Protocol as a common and secure communication basis for electronic control units in vehicles. The project is partially funded by the German Federal Ministry of Education and Research (support codes 01BV0900 - 01BV0917). We would like to thank all SEIS partners directly or indirectly involved in our research.

References

1. Glass, M., Herrscher, D., Meier, H., Piastowski, M., Shoo, P.: SEIS - security in embedded ip-based systems. ATZelektronik worldwide, 2010-01, 36–40 (2010)
2. Apache ETCH homepage, `http://incubator.apache.org/etch/`
3. Hoppe, T., Kiltz, S., Dittmann, J.: Security Threats to Automotive CAN Networks – Practical Examples and Selected Short-Term Countermeasures. In: Harrison, M.D., Sujan, M.-A. (eds.) SAFECOMP 2008. LNCS, vol. 5219, pp. 235–248. Springer, Heidelberg (2008)

4. 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: 31st IEEE Symposium on Security and Privacy, pp. 447–462. IEEE Computer Society, Washington, DC (2010)
5. 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: 20th USENIX Security Symposium, p. 6. USENIX Association, Berkeley (2011)
6. Weyl, B., et al.: EVITA Project, D3.2 - Secure On-board Architecture Specification. Technical Report (2010), `http://evity-project.org/`
7. Wolf, M., Weimerskirch, A., Paar, C.: Security in Automotive Bus Systems. In: 2nd Workshop on Embedded Security in Cars (ESCAR 2004) (2004)
8. Bißmeyer, N., et al.: simTD Security Architecture: Deployment of a Security and Privacy Architecture in Field Operational Tests. In: 7th Workshop on Embedded Security in Cars (ESCAR 2009) (2009)
9. Raya, M., Hubaux, J.-P.: Securing Vehicular Ad hoc Networks. J. Comput. Secur. 15, 39–68 (2007)
10. Plößl, K., Federrath, H.: A Privacy aware and Efficient Security Infrastructure for Vehicular Ad hoc Networks. J. Comput. Stand. Interfaces 30, 390–397 (2008)
11. Ferreira, A.: Android OS changes smartphone life cycle (2011), `http://www.theusdvista.com/mobile/business/android-os-changes-smartphone-life-cycle-1.2000033`
12. Endt, H., Weckemann, K.: Remote Utilization of OpenCL for Flexible Computation Offloading Using Embedded ECUs, CE Devices and Cloud Servers. In: International Conference on Parallel Computing. IOS Press, Amsterdam (2011)
13. Enck, W., Gilbert, P., Chun, B., Cox, L., Jung, J., McDaniel, P., Sheth, A.: TaintDroid: an Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones. In: 9th USENIX Conference on Operating Systems Design and Implementation, pp. 1–6. USENIX Association, Berkeley (2010)
14. Lange, M., Liebergeld, S., Lackorzynski, A., Warg, A., Peter, M.: L4Android: A Generic Operating System Framework for Secure Smartphones. In: 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices (SPSM 2011), pp. 39–50. ACM, New York (2011)
15. Xie, L., Zhang, X., Seifert, J.-P., Zhu, S.: pBMDS: a Behavior-based Malware Detection System for Cellphone Devices. In: 3rd ACM Conference on Wireless Network Security (WiSec 2010), pp. 37–48. ACM, New York (2010)
16. Muthukumaran, D., Sawani, A., Schiffman, J., Jung, B.M., Jaeger, T.: Measuring Integrity on Mobile Phone Systems. In: 13th ACM Symposium on Access Control Models and Technologies (SACMAT 2008), pp. 155–164. ACM, New York (2008)
17. Portokalidis, G., et al.: Paranoid Android: Versatile Protection for Smartphones. In: 26th Annual Computer Security Applications Conference (ACSAC 2010), pp. 347–356. ACM, New York (2010)
18. Nauman, M., Khan, S., Zhang, X., Seifert, J.-P.: Beyond Kernel-Level Integrity Measurement: Enabling Remote Attestation for the Android Platform. In: Acquisti, A., Smith, S.W., Sadeghi, A.-R. (eds.) TRUST 2010. LNCS, vol. 6101, pp. 1–15. Springer, Heidelberg (2010)
19. Bente, I., Dreo, G., Hellmann, B., Heuser, S., Vieweg, J., von Helden, J., Westhuis, J.: Towards Permission-Based Attestation for the Android Platform. In: McCune, J.M., Balacheff, B., Perrig, A., Sadeghi, A.-R., Sasse, A., Beres, Y. (eds.) Trust 2011. LNCS, vol. 6740, pp. 108–115. Springer, Heidelberg (2011)

20. VOGUE Project homepage, http://www.vogue-project.de/
21. Cisco NAC appliance - Clean Access Manager Installation and Configuration Guide, Release 4.9, http://www.cisco.com
22. Arjona, R.: An Introduction to IPsec VPNs on Mobile Phones (2009), http://msdn.microsoft.com/en-us/magazine/ee412260.aspx
23. Bouard, A.: SEIS Projekt, AP4.3, Security der Middleware für IP-basierte Bordnetzarchitekturen (2011), http://www.strategiekreis-elektromobilitaet.de/public/projekte/seis/das-sichere-ip-basierte-fahrzeugbordnetz/pdfs/TP4_Vortrag2.pdf
24. Weckemann, K., Satzger, F., Stolz, L., Herrscher, D., Linnhoff-Popien, C.: Lessons from a Minimal Middleware for IP-based In-Car Communication. In: Proceedings of the Intelligent Vehicles Symposium (IV), pp. 686–691. IEEE (2012)