TOWARDS A SHARED DIGITAL COMMUNICATION PLATFORM FOR VEHICLES

Jan Holle, André Groll
University of Siegen, Chair for Data Communications Systems
Hoelderlinstrasse 3, D-57076 Siegen / Germany
+49-271-740-3325, +49-271-740-2536
{jan.holle, andre.groll}@uni-siegen.de

Marko Wolf, Hakan Cankaya
escrypt GmbH – Embedded Security
Leopoldstr. 244, D-80807 Munich / Germany
+49-89-2080391-{31, 94}, +49-89-2080391-27
marko.wolf, hakan.cankaya}@escrypt.com

Cyril Grepet
TRIALOG
25 rue du Général Foy, F-75008 Paris / France
+33-144706114, +33-144700591
cyril.grepet@trialog.com

ABSTRACT

Giving directions to the driver over a Bluetooth headset, processing position data via GPS and internal vehicle driving data from the CAN bus, while receiving latest traffic data via a GSM connection in the meanwhile. What once was a futuristic vehicle scenario is today's reality. Consequently, communication is one of the key technologies for most automotive innovations. The introduction of standardized open in-vehicle IT platforms enables various (ECU) applications to make efficient, parallel use of the manifold vehicular communication interfaces by sharing available communication resources. In this article, we will discuss the benefits as well as the challenges and requirements for ensuring security and dependability of shared vehicle communication resources realized by an open and standardized platform. The article will moreover present a possible realization based on a efficient and dependable virtualization architecture.

Keywords: shared vehicle IT resources, vehicle communication, security, dependability, virtualization

INTRODUCTION AND MOTIVATION

As today's cars become more and more similar in mechanics, distinctive features and innovations have to be realized in electronics and software with external communications as new driving force behind. Moreover, as the profit margins for OEMs on the primary market stagnate already, aftermarket business models based on digital communications and software become increasingly important for OEMs, suppliers, as well as for third party developers. Especially since aftermarket businesses can cover a wide field of applications ranging from infotainment, traffic management up to enhanced safety and, of course, ITS applications. Following the great success story of today's mobile devices, a new trend in the automotive

research is the introduction of a standardized, open, multi-purpose vehicle IT platform providing multiple runtime environments in parallel by means of virtualization. The idea behind this approach is a more efficient, more flexible, more secure and less costly utilization of hardware resources by merging several individual control units into a more powerful one. Within this approach, an application is simply imported into an isolated and virtualized environment that is provided with access rights to shared resources and services, thus easing the deployment, reducing the costs and enhancing the flexibility. Nonetheless, such an approach requires the adaptation of the resources to a multi-runtime environment. Especially shared communication resources create a critical issue in such an infrastructure. While the idea is to provide flexible and efficient access to the resources, the virtualized platform has to ensure the security and dependability of shared access points.

This article will take a general look on virtualized environments for vehicles and focus especially on the communication and shared communication resources aspects of virtualized platforms. In the first part, communication resources in today's vehicles will be investigated. The next section will state the requirements for creating a secure and dependable infrastructure for sharing resources, especially communication resources, on a virtualized platform. Finally a solution for realizing a secure and dependable mechanism to share resources will be introduced.

VEHICULAR COMMUNICATION RESOURCES

From the ITS action plan (1) as well as current and foreseen applications, a number of communication resources are therefore required to be supported by upcoming vehicular platforms. GPRS/UMTS communication, positioning system, the on-going standard for dedicated ITS communication relying on the 5.9 GHz (2) and further standards have to be support for the ITS use cases (e.g. eCall). Many ITS applications will further gather information coming from in-vehicle sensors, e.g., to be sent to the environment; for instance, to provide reliable information on the traffic situation or the road condition to a traffic management center. The smart car and smart traffic concepts are more and more attractive both to the car manufacturer and the legal entity responsible of road traffic and safety.

Figure 1. Communication Single Point of Access

Besides the ITS aspect, the car manufacturers strive for providing more in-vehicle connectivity to their users and some infotainment relying on radio communication, e.g., Vehicle Area Network (VANet) and Bluetooth-like within the vehicle. The success of social networks, anytime/anywhere connectivity and the potential income for the stakeholders are great motivations.

Moreover, it is a commonly accepted assumption that infotainment applications will require increasing embedded data storage. Actually, the foreseen applications as media exchange between vehicles and location based social networking are obviously resource consuming.

Finally, interactions between vehicles and drivers have to be taken into account. As cars are becoming more and more "intelligent", the interaction with the driver has to be clear, simple and efficient. So the human machine interface (HMI) has to consider these new trends, e.g., the augmented reality solutions.

Actually, it is possible to divide the different communication resources into three categories: (i) external vehicle communication (e.g., VANet, GPRS, UMTS or ITS 5.9 GHz), (ii) passenger communication (e.g., Bluetooth, Wi-Fi or USB), (iii) in-vehicle communication (e.g., HMI, CAN).

Currently, to propose a more efficient and less costly architecture for the in-vehicle embedded system, it is necessary to reduce the number of ECUs by implementing a single access point for communication, that support all the communication resources. But communications are meaningless if they are not used by applications. To gather all the communication resources leads to consider also regrouping all the applications into the same box. But the resources needed by each application in terms of OS, real-time constraints or resources are not the same. A critical in-vehicle application dealing with safety as an emergency break has critical real-time constraints, whereas a social network application requires only update time to time.

To deal with these issues, one of the suitable solutions is to rely on virtualization.

SECURITY AND DEPENDABILITY REQUIREMENTS

The integration of several ECUs and/or applications into a single multi-purpose platform with multiple runtime environments, however, could also yield to some serious implications. To prevent, for instance, malfunction (e.g., programming errors) as well as (malicious) misuse (e.g., unauthorized access) requirements have to be specified which ensure a secure and dependable utilization of shared (communication) resources. The most important high-level requirements are shortly summarized below.

Confidentiality: Only authorized runtime environments are allowed to read transmitted and stored data that is used for utilizing shared communication resources.

Authenticity and Integrity: Data that is used as payload as well as runtime environments that utilize resources should be verifiable regarding their genuineness. We distinguish between data origin authenticity and entity authenticity. In some cases it may be also necessary to have evidences in order to guarantee non-repudiation. Data that is altered, replayed, relayed, etc. by unauthorized parties has to be recognized in a secure way.

Authorization: The system has to ensure a dependable access control for shared resources. Runtime environments should be able to access only resources they have been approved for. Access rights can be granted, modified or removed only by an authorized entity responsible for the respective resource.

Strong spatial and temporal isolation: After the allocation of a time slot and memory space to a specific runtime environment, it has to be guaranteed that the current user has exclusive access to this resource and only to these time and memory allocations.

Availability: Sharing resources between multiple runtime environments requires a reliable mechanism being able to guarantee sufficient allocation and availability on demand or depending on a fair schedule as well as minimum requirements regarding processing power, memory or communication bandwidth per user.

Hardware security anchor: For the provision of security and dependability on higher layers, it is necessary to have a strong security anchor that cannot be tampered by physical means (e.g., side-channel attacks). Additionally, this security anchor is responsible for the integrity of the system software (software protection mechanisms).

Privacy: As the ITS applications can require accurate information on a vehicle (e.g. position, speed, direction) and at the same time infotainment applications can exchange personal data, privacy is an issue for both of these application kinds and many others. Since privacy is rather a second level or sociological term than a clear technical or cryptographic property, in the following just the concrete aspects regarding privacy are covered ID privacy(sender does not want to reveal its identity without his explicit approval and never want to reveal its identity to any unauthorized entities) and location privacy (sender does not want to reveal its location without his explicit approval and never want to reveal its location to any unauthorized entities). Actually as explain in (11), this includes some security requirements (e.g. accountability or authentication), some dependability requirements (e.g. robustness) and some dedicated privacy requirements (e.g. minimum disclosure or anonymity)

SHARED COMMUNICATIONS FOR VIRTUALIZED APPLICATIONS

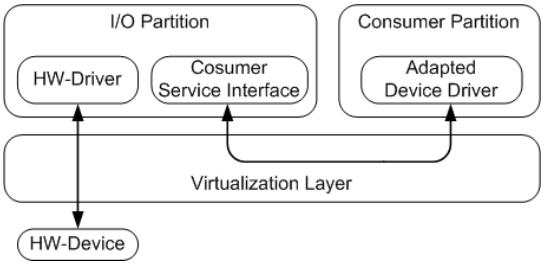

Figure 2. Shared communication resource access

The basic concept of virtualization is to split up the capacity of a resource over a number of consumers, while each consumer gets the impression that he is using the resource exclusively. Hence, hardly any changes to the consumer's code are necessary. This concept works perfect for a lot of resources; for example in the case of CPU time segmentation in several time slots that will be dedicated to specific consumers using the CPU within these slots exclusively. For communication resources connected to entities beyond the virtualized environment the implementation is even more complex, since a simple context switching could interrupt the connections. Therefore, the European project OVERSEE (3) handles external communication resources within one dedicated runtime environment (partition) exclusively and offers access to the established connections via special communication means of the used hypervisor XtratuM (4). Obviously, the consumer's code has to be adapted to cope with these interfaces, which are not identical to the underlying hardware. Hence, a system based on para-virtualization (5) is achieved. Figure 2 depicts the schematic concept for the provision of shared communication resources for infotainment and ITS applications in OVERSEE. The introduced concept with its additional abstraction layer will furthermore enable the addition of security and dependability functionality within the communication paths to fulfill the mentioned requirements. Especially, isolation and authentic communication measures offered by the hypervisor will help to enforce the appropriate security policies (e.g., in terms of confidentiality, authorization and prioritized resource access).

OVERSEE ARCHITECTURE

The following sections will describe the key components of the OVERSEE architecture, aiming to provide a shared digital communication platform for vehicles, which fulfills the given requirements in terms of security and dependability.

PARA-VIRTUALIZATION

One of the techniques to provide access to shared resources in virtualized environments is the emulation of real hardware devices by the virtualization layer. This technique is very convenient for the consumers, typically called guests, since no adaption of the code is necessary. Two of the issues of this solution are (i) the effort to emulate a real device is quite high (ii) all the complexity of the device driver emulation and management of the real device will be assigned to the virtualization layer. Thus, this solution is rather slow and the code base of the virtualization layer will be very huge. Furthermore, an error within the device drivers or device management could affect the whole system.

Concerning the drawbacks, discussed in the section above, another technique, called para-vitualization gained a lot of interest in the past years (5). Guests executed on a virtualization layer, using para-virtualization, are aware of the fact that they are executed in a virtual environment. This means that the code of the guest, typically the device drivers of an operating system, will be adapted to the subjacent virtualization layer and the provided interfaces for the virtualized resources. Thus, the interfaces of the virtualized resources in the virtualization layer could be simple and hence the virtualization layer could be unburdened to achieve a more efficient solution, including a restricted propagation of errors caused by faulty device driver implementations.

One of the goals of OVERSEE is the reusability of legacy applications, especially from the infotainment domain. Many of these applications are developed for Linux based runtime environments; hence the Linux device drivers need to be adapted to the interfaces of the virtualized resources. The Virtio driver set (6) in Linux already provides Linux drivers which actually access the interfaces of virtualized resources.

An exemplary design for the integration of IP networking capabilities - whether the real network interface is a WiFi, 2G/3G or wired module - is depicted in Figure 3.

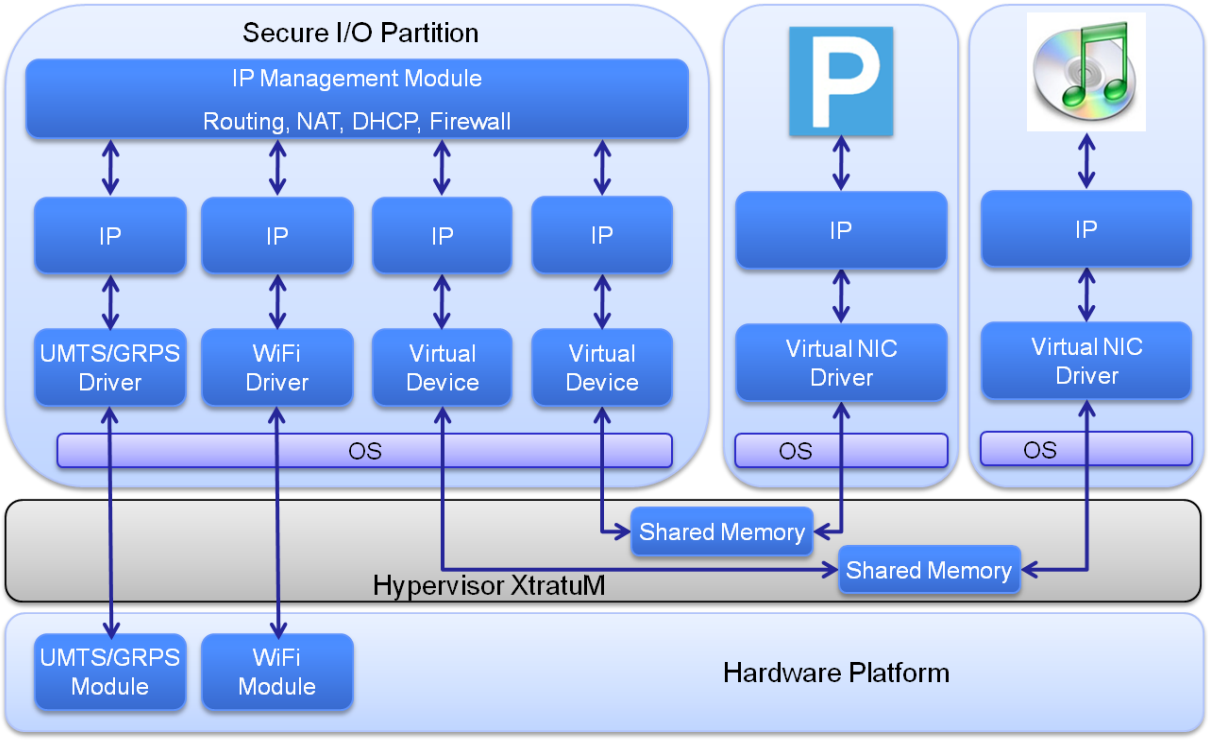

Figure 3. IP integration in OVERSEE

As shown in Figure 3, the device drivers for the real hardware devices will be executed in the so called secure I/O partition. The virtualization layer only provides generic communication means for interpartition communication, in this example areas of shared memory, and exclusive access rights to hardware resources to dedicated partitions. The device drivers, used to provide virtual devices in the runtime environments, will access the generic communication means of the virtualization layer, instead of the real hardware.

SECURITY ARCHITECTURE

The virtualized architecture of OVERSEE enables different runtime environments to run parallel on the same platform with different levels of trustworthiness. As the access to the various resources of the system is restricted by the OVERSEE architecture the need for integrity and trustworthiness can be limited to a minimal number of modules of the overall system. Based on such a trusted base further enhanced security services can be built upon. In this section of the paper the general architecture of the OVERSEE security concept is explained and the possibilities to ensure system integrity with this architecture is discussed. Furthermore some of the individual services which can be built upon this security concept are summarized.

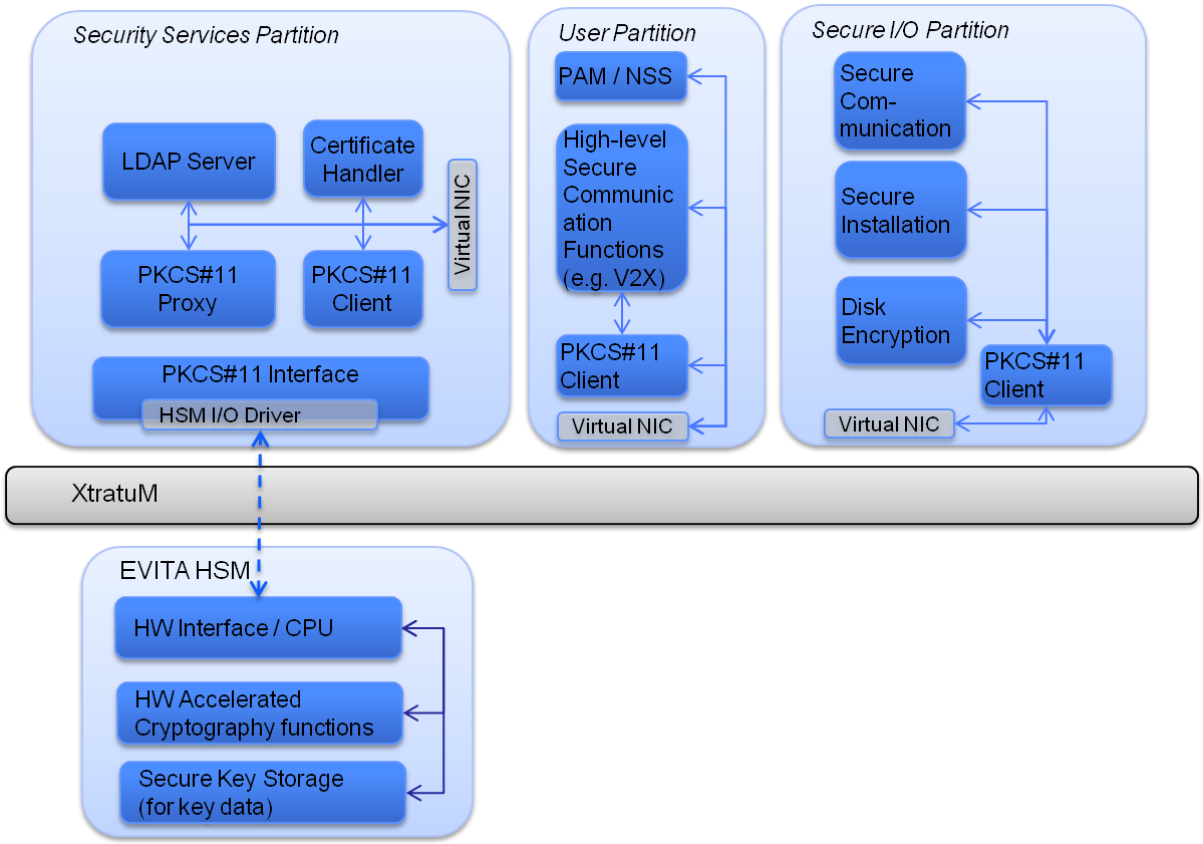

Figure 4 - OVERSEE Security Architecture

As seen in Figure 4 the OVERSEE architecture involves a hardware security module. In the first version of the OVERSEE architecture the implementation is based on the EVITA HSM (Hardware Security Module) (12), which provides many essential services and features serving as a base for the security concept of OVERSEE. The HSM is logically coupled to the security service partition of the OVERSEE platform which provides a secure and isolated runtime environment for security services in general that can be requested by the other partitions through secured communication channels.

As mentioned before a trusted configuration of the system is essential to build the security concept upon. To assure this a secure boot process assisted with the HSM is integrated in the OVERSEE architecture. The EVITA HSM adheres to a large extend to the TPM specification (7) and enhances this specification with further features, creating an essential part of the secure boot functionality.

The secure boot process starts with the authentication of the XtratuM hypervisor and the two essential partitions, the secure I/O partition and the security services partition. The hash values of the current configuration of these parts of the OVERSEE architecture are compared with the known configuration values of the platform in the HSM. Based on this comparison many actions can be taken and/or enforced by the architecture. One action would be to restrict access to cryptographic keys by coupling the correct hash value of the system configuration to cryptographic keys stored in the HSM. Furthermore the HSM can provide signed attestations of the system configuration which can be used by external entities to remotely validate the integrity of the system. The EVITA HSM provides multiple such configuration hash registers which may be used to detail the integrity validation of the system. This can be used to validate

only the integrity of one specific runtime environment (or only a part of it) and take actions only for this part of the OVERSEE platform.

The parallel execution of the runtime environments in an integrated architecture enables also other approaches. A known issue with today's systems is that cold start of systems is not very common anymore; the systems start usually from a stand-by or hibernated state which excludes the whole secure boot process. Furthermore the fact that harmful changes can only be recognized in the next boot process already can cause severe security flaws. The parallel execution of the security service partition can react on such situation more dynamically. The integrity of specific memory areas or stored data can be validated cyclic on-the-run or after specific actions like warm-start or software installation. The actions upon recognition of non-expected configuration can vary from just notifying related modules to stopping the tampered partition from running.

The security of stored data in the system is assured by storage encryption and individual services for encrypting individual files. The key material for the encryption is stored in the HSM, providing a sealed storage. The file system of the user partitions are provided by the secure I/O partition using Virtio, thus enabling central encryption/decryption of the file systems and a seamless integration of the encryption/decryption for the user partitions.

OVERSEE architecture provides a central point for handling software installations. This central instance can be used for a variety of use-cases and business models providing a mandatory verification procedure in means of authorization, integrity, authenticity, compatibility and dependencies. The software package structure is based on a standardized format, in our case the Debian package (8). The central handler serves as a proxy between the partitions and the repositories and validates the packages before forwarding to the destination and initiating an installation. It also provides functionality for updating whole partition images or the hypervisor.

The direct services provided by the security service partition can be summarized as follow:

- A controlled interface to cryptographic functions and key material hosted by the HSM.
- Central handling of authentication and authorization information.
- Certificate handling.

OVERSEE aims to rely on standardized interfaces and provide a modular design. Therefore the interface to the security module is based on the PKCS#11 (9) specification which is supported by most of the security modules (e.g. smart cards, hardware security modules etc.). This enables to also use other security hardware instead of the EVITA HSM with minimal effort. The PKCS#11 interface is tunneled to the other partitions by a proxy and a PKCS#11 client driver hosted at the other partitions. The OVERSEE PKCS11#-proxy design enables parallel access to the cryptographic functions and objects of security hardware and also provides a layer for restricting access of the individual services for each partition sending a request over the PKCS11#-proxy. The layer can filter the requests for a various number of parameters as the requested cryptographic object, requesting partition, authentication or authorization status of a specific user. This architecture enables a very flexible and modular design to fit the system to the needs of the platform designer.

The central handling of authentication and authorization data is essential for the security and flexibility of the system. To enable such a service an LDAP (Lightweight Directory Access

Protocol) server (e.g. openLDAP (10)) is provided by the security service partition. This server can be invoked by the other partitions by NSS (Name Switch Service) or LDAP based PAMs (Pluggable Authentication Modules). Also direct usage of the LDAP server through look-up services can be used to retrieve data to validate information as authorization or roles of a specific user, partition or any other entity. Further functionalities like single sign-on are built upon this infrastructure.

The security service partition also provides services for handling certificates and security tokens like certificate validation and creation. Furthermore services for importing security attributes or objects (e.g. cryptographic keys, new users) are provided by the certificate handler. The secure key storage of the HSM enables the storage of trusted public keys (e.g. OEM public key) at the beginning of the vehicle lifecycle, enabling a chain of trust.

Most of the concepts listed here are adaptations of existing and standardized solutions to the OVERSEE architecture. Consequently many of the approaches are based on UNIX systems. To avoid blocking the integration of non-Unix systems into the OVERSEE platform a general approach is taken in the OVERSEE design. The XtratuM hypervisor provides secure channels between partitions which can be configured individually. A non-conform communication standard can always be built upon these channels interfacing with the services in the security partition. For instance, this would be a solution to provide the cryptographic services to partitions not supporting a PKCS#11-driver. The PKCS#11-client driver would be hosted by the security service partition in such a case and listen to the dedicated XtratuM channel for the specific requests.

SECURE I/O PARTITION

The secure I/O partition hosts all of the device drivers for the real hardware devices; therefore the partition executes a small Linux distribution with security enhancements. In general, each communication resource will be managed by its own management component, which will provide access to the real hardware communication interfaces through the hypervisors generic communication means. Thus, novel communication interfaces could be added to the OVERSEE architecture easily.

Some of the communication interfaces are represented by a similar virtual interface driver in the partitions, serving the applications, e.g., IP networking capabilities introduced in the section above. While for some other interfaces an additional abstraction will be applied. Figure 5 depicts the integration of access to the vehicle internal networks as an example for this type of interfaces.

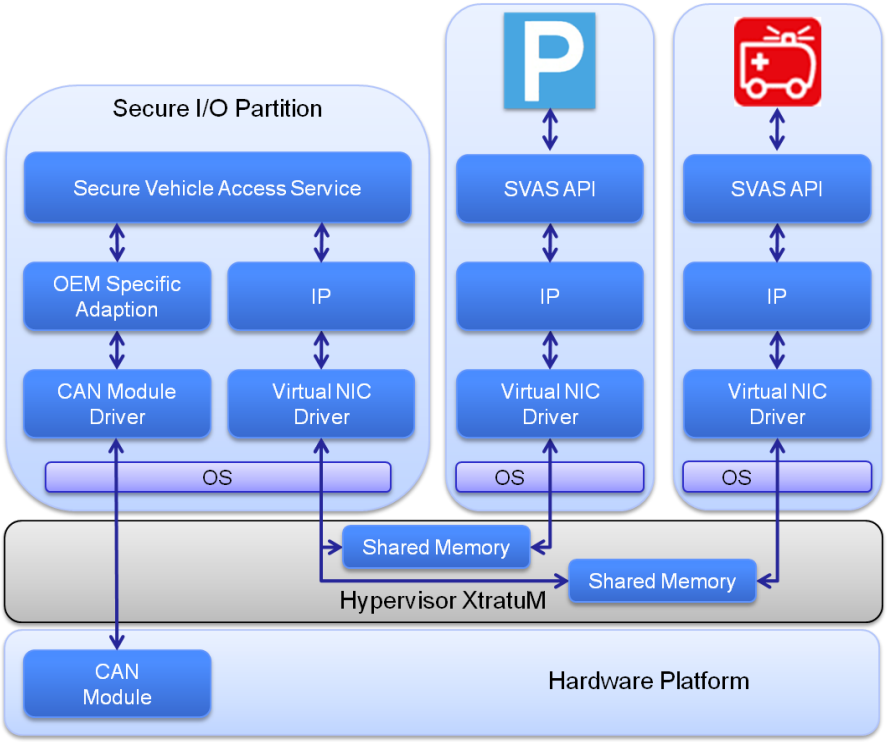

Figure 5. Secure Vehicle Access Service in OVERSEE

Access to the vehicle internal networks is highly critical, from a safety point of view; hence this leads to high needs in terms of security. Within the SVAS (Secure Vehicle Access Service) vehicle information will be represented by so called data objects. All data objects have a vehicle independent name. This additional abstraction leads to (i) the opportunity to build vehicle independent software; although using information from the vehicle internal networks (ii) the possibility to add further policies to specify access rights on the level of data objects on a per partition scheme. Thus, all data exchange between the vehicle internal network and the applications executed on OVERSEE will be filtered. However, the vehicle manufacture could still decide to introduce an additional gateway, between the OVERSEE CAN module and the vehicle network, to enforce communication policies in hardware.

CONCLUSION AND OUTLOOK

The introduction of multi-purpose platforms for vehicles in order to integrate and execute several control units and/or applications in parallel is one of the key technologies for upcoming innovations. There multiple runtime environments are able to share available hardware and communication resources in a very efficient manner. Since the proposed sharing approach of course has to be secure and dependable, we have specified a set of requirements to prevent unintended malfunction as well as (malicious) manipulations. These requirements can serve as evaluation criteria for possible solutions.

One solution to meet the requirements is analyzed and prototyped by the European research project OVERSEE that uses a para-virtualization approach with multiple coexisting and isolated runtimes. A functional demonstrator of the OVERSEE platform that shows the secure and dependable shared access to communication resources will be available in 2012.

ACKNOWLEDGEMENT

The presented paper is part of the ongoing work in the OVERSEE project, a European research project funded within the 7th Framework Programme of the European Union (Project ID FP7-ICT-248333). Therefore, we would like to thank the European Union for funding and all other partners of the project for their jointly contribution.

REFERENCES

(1) Commission of the European communities, "Action Plan for the Deployment of Intelligent Transport Systems in Europe", December 2008

(2) ETSI, "European Profile Standard for the Physical and Medium Access Control Layer of Intelligent Transport Systems Operating in the 5Ghz Frequency Band", ES 202 663 V110

(3) Open Vehicular Secure Platform (OVERSEE) Project, http://www.oversee-project.com

(4) XtratuM Hypervisor designed for real-time embedded systems, http://www.xtratum.org

(5) Whitaker A. and Cox R.S. and Shaw M. and Gribble S.D, "Rethinking the Design of Virtual Machine Monitors", *Computer vol.38 no.5*, IEEE, May 2005, pp. 57-62

(6) Russell R., "virtio: Towards a De-Facto Standard For Virtual I/O Devices", *ACM SIGOPS Operating Systems Review - Research and developments in the Linux kernel - Volume 42 Issue 5*, ACM, July 2008Weyl Benyamin, Wolf Marko et.al., "Deliverable D3.2: Secure On-board Architecture Specification", 21. December 2010,

(7) "TPM Main Specification Level 2 Version 1.2, Revision 116". Available at http://www.trustedcomputinggroup.org/resources/tpm_main_specification

(8) "Basics of the Debian Package Management System". Available at http://www.debian.org/doc/FAQ/ch-pkg_basics.en.html

(9) RSA Laboratories, "PKCS #11 v2.20: Cryptographic Token Interface Standard", 28 June 2004

(10) "OpenLDAP", Available at http://www.openldap.org/

(11) Schaub F. and Ma Z. and Kargel F., "Privacy Requirements in Vehicular Communication Systems", *International Conference on Computational Science and Engineering (Vancouver, 2009)*, IEEE

(12) E-safety vehicle intrusion protected applications (EVITA) Project, http://www.evita-project.org