Driving Automotive Middleware Towards a Secure IP-based Future

Alexandre Bouard*, Benjamin Glas†, Anke Jentzsch‡,
Alexander Kiening§, Thomas Kittel¶, Franz Stadler‖, Benjamin Weyl*

*BMW Group, Munich, Germany, {alexandre.bouard, benjamin.weyl}@bmw.de
†Robert Bosch GmbH, Stuttgart, Germany, benjamin.glas@de.bosch.com
‡Volkswagen Group, Wolfsburg, Germany, anke.jentzsch@volkswagen.de
§Fraunhofer AISEC, Garching, Germany, alexander.kiening@aisec.fraunhofer.de
¶Technische Universität München, Munich, Germany, kittel@sec.in.tum.de
‖Continental Automotive GmbH, Regensburg, Germany, franz.stadler@continental-corporation.com

***Abstract*—Today's vehicle has become a very complex good, offering performance and reliability thanks to a sophisticated network of Electronic Control Units. Each year car makers develop new automotive features for use inside and outside of the car. But improving this infrastructure each time adds new costs and overhead to the system. A promising solution resides in the use of Internet Protocol (IP) standards for both on-board and vehicle-to-X communications. With IP over Ethernet, bandwidth and system performance will increase, but so will the security risks for both passenger safety and vehicle system integrity. IP is a well-known and attack-prone technology and the car an easy prey. Communication encryption and static access controls won't be sufficient; more suitable and complex security protocols and infrastructures are required. In this paper we present a new middleware architecture for on-board security, allowing establishment of internal and external secure communication channels and secure runtime environment for automotive applications. The modularization of our architecture allows protocol independency and adaptability in terms of performance and security for automotive use cases.**

I. Introduction

In decades of technical and functional evolution automotive embedded systems have grown to a very specialized, highly reliable, effective and cost-efficient composition for a wide range of real time safety critical functions. In recent years the major share of innovations in the automotive sector, especially in the area of driver assistance, is based on software executed distributedly on networked electronic control units (ECU) in the vehicle, more recently also beginning to be supported by communication over the borders of the vehicle, e.g. to mobile devices like smartphones, to infrastructure, or to other vehicles.

Since many of the systems are built to improve and guarantee the safety of traffic participants inside and outside of the car, a failure, deficient function, or manipulation of these systems poses an implicit threat to the lives of these persons. Many steps of technical improvements were done to harden the systems and to deliver a very high level of protection against failure and malfunction. But in recent years an additional threat arose that has to be considered: direct, deliberate and potentially malicious manipulation. Apart from long ongoing discussions and awareness in research and development, several recent academic publications clearly showed the attack potential on current vehicles. This attention was also perceived by a broader public. Kosher et al. showed ([1]) some vulnerabilities of automotive bus systems assuming the attacker is able to obtain direct access. Additionally Checkoway et al. [2] showed how weaknesses in external interfaces can be leveraged to gain access and manipulate internal systems, using common weaknesses known from the computer world like bad policy checks, insecure implementations and static cryptographic material. It becomes apparent that system security has to be guaranteed in order to maintain safety especially in embedded systems and that the growth of software-based functions and external interfaces in the automotive domain rapidly increases the attack surface and therefore the need for protection against malice.

Most of the automotive systems operate at least semi-autonomously, based on a clearly defined mission, without direct interaction with the owner or user. A mostly static structure in hardware and software makes it a complex task to change or add functionality like security features, especially when a holistic design approach is necessary as it is the case when securing a system. Security functions have to be evaluated very close in their response to attacks and errors, since the system has to maintain safety requirements at all costs and security measures therefore must not affect safety functions.

With the introduction of Internet Protocol (IP) based communication and usage of standardized IP-solutions (subject of the SEIS project [3]), security becomes even more important, since knowledge about, tools for, and attacks on IP are wide-spread and common. Therefore SEIS considers security in a dedicated part project proposing an architecture for a secure communication infrastructure recommending IPsec as a security-protocol. But some problems remain, e.g. how to set up the infrastructure regulating and securing the communication channels, where to enforce security policies, where

and how to store secrets and cryptographic material securely, or where to enforce access control based on which criteria. Hence another work package inside the security part project of SEIS focuses on these questions with a middleware-oriented view, having the goal of establishing secure communications (internally and externally) and to ensure a secure runtime environment for applications with security properties tailored to the respective application, including authorization, authentication, integrity and confidentiality of data and communication. This paper presents the security architecture proposed by the latter work package.

The remainder of this paper is organized as follows. Section II provides some background information on automotive security and next generation on-board networks. We then present in section III our security architecture and concrete runtime functionality. The same section discusses error management and provides a description of our prototypes. Finally in section IV we discuss some related work and conclude in section V.

II. Scope of this work

In this section we provide background information on automotive networks and security. We present relevant threats and the adversary model considered for this work.

A. Future Automotive Applications and Networks

During the last decade, the car became a complex IT infrastructure. In several upper class vehicles, up to 80 ECUs are interconnected thanks to different proprietary bus systems like CAN, MOST or FlexRay. These bus systems have their own requirements, protocols and are not interoperable with each other. For the moment on-board communication mostly consists of simple control signals and the coupling between two different buses is achieved through complex translation gateways. Since the communication carries no routing information, the gateways are statically configured to translate and route every packet. On the contrary, communication between two IP networks relies on simple switches or routers and a suitable packet addressing at the data link or network layer and can allow dynamic routing features, since the packet header includes both source and destination. IP-based communication can simplify the on-board infrastructure and make the integration of external communication partners easier, e.g. for interaction with a back-end server, CE device integration over WLAN or LTE.

Future applications will exchange bigger and more expressive messages, e.g. complex models for radar environment perception or high-definition navigation maps. For the next generations of automotive applications, a significant bandwidth increase will be necessary in the near future. The solution of this challenge is not possible with current automotive communication technologies. An automotive variant of a 100 Mbit Ethernet is a serious candidate allowing such performances and a Gigabit Ethernet version could be soon available as well.

IP over Ethernet is apparently a good solution for automotive on-board communication; it complies with most automotive requirements and offers standards, directly adaptable for the car purpose [4]. Though, in order to properly administrate the on-board communication, a middleware offering an efficient abstraction layer for network management and security is required.

B. Security Threats for Automotive Middleware

Automotive middleware threats are similar to the ones concerning traditional communication middleware and can be listed in three categories.

1) *Asset Corruption*: Attacks aim at modifying or destroying automotive data and services. They are usually resulting from software and implementation weaknesses (weak security mechanisms, injection of bogus packets) and allow unauthorized use of the service (discovery of wrong services, ECU memory modification, ECU flashing...).
2) *Information Disclosure*: Successful attacks result in unauthorized access to middleware assets (mostly data). Attackers aim at stealing automotive information from the driver (privacy infringement) or from the car manufacturer (industrial espionage). Both communication link and data storage units can be targeted.
3) *Service Interruption*: Here the service availability is affected. Attacks aim at making the service unusable or slower, they usually consist in resource exhaustion (jamming/flooding), unauthorized service deactivation or in producing errors and thereby delays in the system.

These threats are mainly posed by attackers or adversaries that are generally driven by motivations like theft of vehicles, obtaining features by fraud, illegal reuse of components, data manipulation or causing malfunctions.

C. Adversary Model

Adversary models are essential for security risk analyses and can be formalized. As an example, the Dolev-Yao-model [5] specifies an attacker with full control over the network who can intercept, modify and replay all messages.

Our adversary consideration bases on a mighty internal or external attacker with properties of the Dolev-Yao-model. We assume timely unrestricted access of attackers to the vehicle. We assume the attacker to be only polynomially bounded in computational power and storage, so that the current cryptographic primitives (like AES, RSA etc.) can be assumed to be secure, since there are no algorithms known to break them in polynomial time. Thus the assumed adversary cannot break strong cryptographic protocols or successfully guess random numbers. Due to the scope of our work, we restrict the attacker to software-based attacks so that he cannot physically tamper ECUs for e.g. memory content reading or flashing. Hardware-based attacks are in principle out of reach of middleware-based security mechanisms.

III. Securing the Middleware

Previous work done within the SEIS Project [6] motivated the necessity to classify automotive on-board communication regarding their characteristics and requirements. A model in

three security zones was designed, highlighting the different security risks for the on-board network. Building on this work, the following section proposes concrete solutions to comply with the mentioned model.

A. Security Requirements

With regard to the risks and challenges described above, our system should comply with the following requirements:

- *Functional requirements*: The security middleware should provide reliable on-board communication and propose performances at least similar to traditional automotive systems. Additionally, it should allow integration of external services running outside the car.
- *Communication requirements*: The security middleware should provide secure communication channels between ECUs. They should offer mutual authentication mechanisms and assurance of integrity and confidentiality for the exchanged messages depending on the application's needs. In the same way, the security middleware should adapt the secure channel when dealing with external communication.
- *Application runtime requirements*: The security middleware should assure integrity of every ECU platform. Access control to any ECU-resources shouldn't degrade the overall system performance, disclose secret information or endanger the passenger in any way.

B. Security Middleware Extension

This paper presents a security middleware extension providing modular security services for automotive on-board networks. These security services provide the basis for secure applications runtime and establishment of secure communication channels. For the rest of this paper we define the abstract representation of these security services as security modules, independent from their implementation (present on several ECUs or not). The security extension communicates with the communication middleware through the Security Abstraction Layer and relies on security Plug-Ins included within the modules. The Plug-Ins are the concrete security mechanisms and communication protocols used by the system. This architecture is designed to offer scalability, flexibility and middleware protocol independency following some principles:

- *Suitable modularization*: each extension sub-component can be adapted and configured based on specific security requirements (e.g. use cases, considered security zones and additional functional information) and therefore will be optimally designed for its target purpose. Modularization allows the definition of different versions for each module offering an adapted choice of security Plug-Ins, for example a full- versus a light-security version.
- *Abstraction of security interfaces*: for a flexible integration of security services, suitable abstracted interfaces should be defined. They simplify the enforcement of security at each level (e.g. in the application and in the middleware at transport-, network- and physical-level).
- *Configurability*: The system should allow static and dynamic updates for configuration at the end of assembly or during system runtime.

Developing automotive applications with security in mind can be complex though. The proposed middleware extension aims at easing and automating this process: each application and communication should be mapped with a set of both functional and security-relevant requirements, that can be declared either in the middleware during service definition for static implementation or in application-level context data, transferable to the security services. The development of a security middleware allows the car manufacturer to mitigate the risk of potential security misconfiguration of the application.

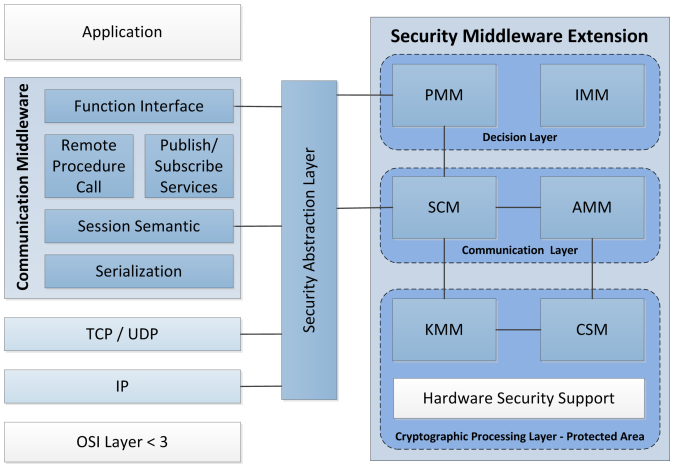

Fig. 1. Connection between Communication Middleware and Security Middleware Extension (Abbr. AMM: Authentication Management Module - CSM: Crypto-Service Module - IMM: Intrusion Management Module - KMM: Key Management Module - PMM: Policy Management Module - SCM: Secure Communication Module)

C. Security Modules

The SEIS security middleware extension consists of six modules integrated within the middleware and specialized for six different security purposes. Figure 1 shows the interactions between modules and with the functional middleware- and application-layers. A more detailed description of each module follows later in this section. The modularization is based on a security management consisting of three layers:

- A security decision management and monitoring layer, in charge of the authorization process and the detection of security violation consisting of the Policy Management Module (PMM) and the Intrusion Management Module (IMM).
- A security layer responsible for the establishment and verification of secure communication channels: the Secure Communication Module (SCM) and the Authentication Management Module (AMM). This layer controls the session semantics of the security protocol and guarantees authentication, confidentiality and integrity of the exchanged messages according to the respective needs.

- A layer in charge of trust anchoring and management of cryptographic processing. The Key Management Module (KMM) stores and administrates the keys and the Crypto Service Module (CSM) controls the process of encryption/decryption and generation/verification of authentication fields.

This modularization offers the possibility to take the hardware and software capacity of each ECU into account and allows to setup the security on both levels as necessary.

1) Crypto Service Module (CSM): Most of the security mechanisms used on different layers and by different applications are based on a couple of carefully chosen security primitives and algorithms which in turn are based on the usage of sensible, secret cryptographic data, like keys. Integrity, authenticity and confidentiality of these keys are basic assumptions for the security of the entire system so that these secrets have to be especially protected. In addition all mechanisms for security can only be as secure as the base primitives, their implementation, and the hardware and software they are executed on.

The CSM offers cryptographic services like encryption, decryption, digital signature generation and verification, processing of Message Authentication Codes (MACs), etc. based on various cryptographic primitives and algorithms. Calling entities and processes can parameterize the request to use the appropriate primitives and modes and also give data and keys via handles. Depending on the security properties (e.g. key storage, protected processing…) and processing capabilities (especially asymmetric primitives tend to be resource intensive) of the respective ECU these services can be implemented locally or centrally, in the latter case the local request is relayed over the vehicle internal network using a secured channel (see SCM in section III-C2). The SCM therefore abstracts from the actual implementation and also location of processing.

The grouping of security processing as a service has several benefits. First of all the cryptographic processing is the only service in the system that needs direct access to secret cryptographic key material. All other entities and processes, even the actual authorized owners of the material and credentials only need a key identifier/handle to point out the key to use to the CSM. So all internal communication and parameter transmission can be done with key handles instead of keys in plain text, reducing the need for confidentiality and mitigating the risk of disclosure. The actual key is only communicated between CSM and KMM (see section III-C4) using a specially protected channel. Ideally this can be done by integration of CSM and KMM in one protected environment (e.g. on an Hardware Secure Module - HSM).

A second benefit is that implementation of cryptographic algorithms can be pooled. So it is easier to maintain and enforce secure programming guidelines (e.g. also for protection against side-channel attacks) to make sure correct and secure processing, locate the implementation to suitable platforms and make use of hardware support or acceleration if available. As the entity responsible for crypto services the CSM is a crucial part of the trust anchoring of the overall system. Also maintenance of algorithms and adding of new primitives can be done more centrally for a better Crypto-Agility. In addition the processing shall be done in a secure, isolated environment. Approaches may e.g. include secure virtualization with hypervisors and strong seperation of memory. If further protection mechanisms like a dedicated HSM [7], [8] are available, security can be improved by using dedicated, access restricted processing units only for secure processing. For resource-intensive primitives like asymmetric algorithms (e.g. RSA [9] or ECC [10], [11]) and especially for real-time environments hardware acceleration may be necessary and also part of the HSM.

2) Secure Communication Module (SCM): In current and even more in future vehicular networks, new functionality is often realized by interconnecting functions that are distributed across numerous ECUs. Communication relationships are being formed to facilitate information exchange and remote function invocation. These relationships have use case specific requirements of communication reliability, bandwidth and also security. In order to avoid applications implementing their own custom communication mechanisms, the ECUs' middleware often provides the necessary infrastructure. Traditionally, these mechanisms seldomly cover the applications' security requirements and leave this task to custom application layer solutions. As implementing security mechanisms is non-trivial and thus error-prone, this should be avoided.

It is the purpose of the SCM to solve this problem at the middleware layer and provide transparent access to communication services that meet the applications' specific security requirements. The SCM hides as much complexity from the application as possible by offering an API that is based on the standard POSIX [12] socket API. If an application for example wants to establish a secure communication relationship, it invokes a function called *openSecConnection* and passes the desired security requirements as parameters. These requirements encompass connection properties such as whether the transmission has to be authentic, integrity-protected, or confidential, as well as the security credentials to be used. Additionally, a set of actual security mechanisms that are to be used can be set. The specification of the security requirements is optional. If nothing is specified by the application, the SCM contacts the PMM to query for the configured standard security policy for such a connection.

In order to create a holistic security solution, the SCM integrates with the middleware's other security modules. This includes mainly the modules AMM to facilitate the authentication process while establishing a secure connection, the CSM to perform the applied cryptographic calculations and the KMM to access the necessary cryptographic keys. In order to comply with the ECU's configured security policies, the SCM cooperates with the PMM. Section III-D depicts the internal message flow among the involved security modules while sending a message.

3) Authentication Management Module (AMM): Responsible for the authentication process of every internal and external

entity, the AMM tasks can be separated in two parts: Accountant and Enforcer. Implemented on every ECU establishing authenticated communication, its main role is to support the SCM with the communication security management.

First as accountant, the module maintains a mapping of every entity's authentication, i.e. status of the process, associated protocol and security token ID. Its role also includes revoking and managing the generation of new tokens (certificates or authentication tickets). Besides, it can support the IMM with analysis of patterns related to the authentication process. This module can be extended for vehicle-to-X (V2X) communication with privacy mechanisms (for pseudonym management) and modules for hardware based remote attestation.

For all the considered protocols (IPsec, TLS/SSL, DTLS) authentication plays an essential role to authenticate the peer and to give proof of origin for every exchanged data frame. As enforcer, the AMM's task is synchronization of KMM and CSM for appropriate generation and verification of the authentication fields (e.g. digital signature or HMAC).

4) Key Management Module (KMM): The KMM provides access to cryptographic keys that are necessary for executing cryptographic operations. To guarantee a high level of security, the KMM is tightly integrated with the CSM - both together forming a special trusted zone as depicted in figure 2. Cryptographic keys are to be used in this zone exclusively and are not permitted to leave this zone. If applications or modules of the middleware request access to keys, the KMM only provides indirect access via handles. These handles can be used to invoke cryptographic functions of the CSM. As the CSM is part of this trusted zone, it can use the provided handles to retrieve the necessary keys from the KMM and perform the requested operations.

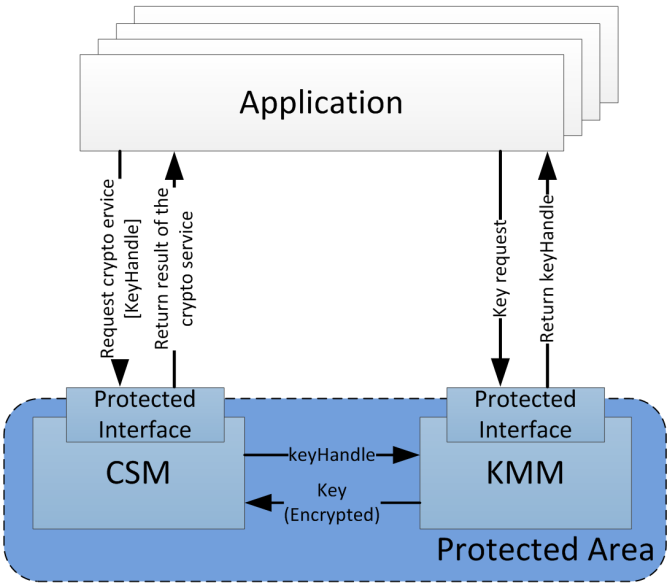

Fig. 2. Collaboration of CSM and KMM in secure environment. Keys are never communicated in plaintext outside this protected area.

One of the purposes of the KMM is to hide the actual key storage mechanism from the requesting party. As cryptographic keys are to be considered critical information, they have to be properly secured. However, implementation depends on the actual capabilities of the underlying ECU. For example, if an ECU is physically equipped with a Hardware Security Module (HSM), this can (and should) be used to properly encrypt the key storage and keep the storage encryption key secure. If this is not the case, the KMM has to implement other measures to protect the keys in the ECU's flash memory. It is clear that these alternatives are less secure and just increase the effort an attacker has to take.

As the automotive industry is very cost driven, the equipment with HSMs is expected to be kept to a minimum. If an ECU is equipped with an HSM, it can offer specialized services based on the HSM to other ECUs. One of these can be to provide a secure storage for cryptographic keys. Such a key server can store cryptographic keys that are too critical to be stored locally in potential insecure flash memory. This key server can be transparently integrated into the local KMM to hide the additional complexity of remotely requesting access to cryptographic keys.

5) Policy Management Module (PMM): As pointed out in the name, this module is in charge of the policy decision process for every application and security service. Its role is mostly consultative; the requester has to enforce the decision itself. However with support of the IMM, the enforcement can be monitored and appropriate reactions (e.g. process isolation, recovery process) can be taken in case of a security issue. The PMM is implemented on every ECU requiring the enforcement of a security decision. It should locally dispose of all the necessary policies and context information in order to minimize the risk of error and latency. A central interface is responsible to deploy the policy and security configuration (firewall rules, virus signature database) at the end of assembly and update these policies during functional runtime with localized policy updates while the car is stopped.

Designing policy management models for complex and time-critical systems like cars can be challenging, trade-offs between policy expressiveness and processing performance have to be made. In order to limit the latency and risk of errors we designed two policy formats [13]. The first one at the middleware-level defines the authorized communication between pre-defined internal services and over specific security protocols, whereas the second, more expressive, provides constraints evaluation possibilities (time, location, user ID) and is particularly adapted for not-time-critical applications with user interaction or integration of external services (CE device, back-end server).

6) Intrusion Management Module (IMM): The IMM aims to defend the system against external tampering or targeted attacks on single or multiple ECUs within the car. It is a distributed module, in which the different implementations of the IMM inside the middleware of different ECUs work together. The different ECUs do not have to run the same implementation of an IMM. It is even suggested to use different implementations on different types of ECUs. The cooperation can be seen as a distributed intrusion detection (or prevention)

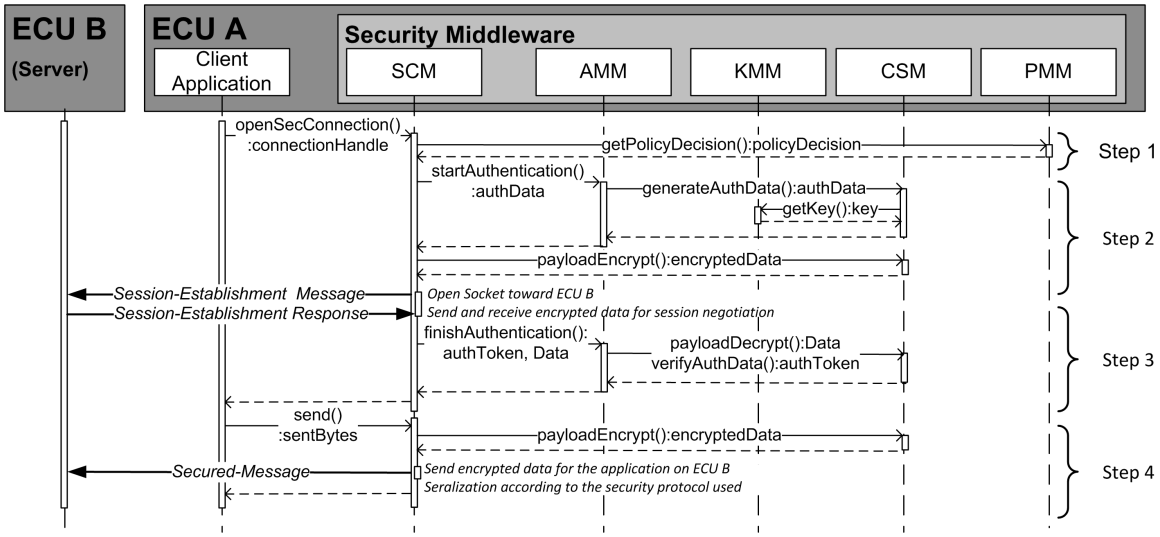

Fig. 3. Use-case: Open connection & data sending

system within the automotive middleware. All collected data can be aggregated on central nodes, where further analysis is performed and further decisions on countermeasures are considered. As a countermeasure the system can, for example, decide to send warnings to the driver or notify the car manufacturer about a distributed attack [14].

The IMM has no specific external dependencies to collect its data. However its senses can be extended with information forwarded by other modules within the security middleware. The AMM could, for example, inform the IMM about authentication errors, the PMM could forward policy infringements and the SCM could forward attempts to open insecure connections.

Like in the world of customer electronics there are multiple ways of implementing intrusion detection systems. For a detailed description see [15]. To adapt the needs of the ECU (data intensive multimedia service or safety-critical sensor) the IMM is able to implement different mechanisms to detect tampering with the system. Signature-based detection or remote attestation on a power constrainted ECU. A routing ECU, like the cars security gateway, could implement its IMM by inspecting network traffic.

In addition to the common types of host-based and network-based intrusion detection, a further type of intrusion detection is interesting in the automotive environment: Introspection-based Intrusion Detection [16]. An introspection-based IDS is able to inspect the state of an ECU from an external point of view. It is particularly interesting with virtualized ECUs. The IDS on the physical ECU is then able to use information from the hypervisor to make assumptions about the monitored ECUs internal state.

Within the proposed IMM architecture a host-based or introspection-based intrusion detection module should be placed on every ECU, whereas important ECUs also should be integrated into the attestation process by requiring periodic remote attestations [17]. Furthermore the architecture suggests to apply network-based intrusion detection to all routing ECUs within the system.

7) Security Abstraction Layer (SAL): The SAL provides a logical interface linking applications and communication middleware to its security extension. A generic API [13] accessible from the application and middleware allows the developers to query access control decisions or to transmit requirements (abstract or not) relative to the communication security. The SAL is in charge of resolving the dependencies relative to the security modules, the communication middleware and their different versions.

Inversely, the SAL provides the security services with interfaces on several layers of the ISO-OSI model to allow an efficient integration of the security plug-ins at both application- and middleware-level.

D. Functional Use Case for the Security Extension Management

As we described the proposed security extension in the previous section, we will now demonstrate its usage by explaining a functional use case. We consider the connection establishment process and sending of data to a server through a secure connection. The connection established is bidirectional, although it is used unidirectionally in the following example. We assume that the server is already configured to receive the intended connection and the server socket is already opened using the corresponding function of the security middleware. The server functionality is analog to the client side flow path shown in this example.

The process of secure connection establishment is also shown in figure 3. In order to establish a secure connection the application initiates the security s *openSecConnection* function which initiates a process consisting of four different steps. An application optionally can provide the desired connection properties as parameters to the function call. If the application does not provide connection properties, these are requested

from the PMM (default parameters are provided to prevent connection misconfiguration).

In the first step, the SCM requests a policy decision from the PMM whether the establishment of the intended connection is permitted. In case the connection is denied, the PMM informs the IMM about the possible intrusion.

After the SCM assured that the connection is permitted, the authentication phase starts in the second step with a call to the AMM, which prepares the authentication handshake by calling the CSM. The CSM then directly requests the corresponding key material from the KMM. The SCM is then able to request encryption of payload from the CSM.

Now the SCM establishes a connection to the server and requests the AMM to finish the authentication phase in the third step. The AMM handles the necessary handshake used in the underlying authentication protocol. At the end of step three the authentication is finished and a secure connection is established between client and server. The application is now able to send data through the secure connection in step four. The payload is sent by the communication middleware using the properties and keys specified within the connection previously established.

This example presents a simple three-way authentication handshake, but more complex handshakes are supported as well. The SCM processes the additional exchanged messages in a same way in step two and three. Considering the time demanding requirements for start-up and during the runtime, the connection establishment and exchange of session keys can already be accomplished at the assembly line.

E. Error Management

Adding security into cars should not degrade performance or produce faults within the application or middleware. We consider as fault any interruption of the normal operativeness of a subsystem or of any of its sub-components. This includes both accidental events and intentional ones (e.g. caused by attacks). The latter ones are difficult to detect, differentiate and isolate. When a fault occurs, fault detection mechanisms should be able to identify the fault and create awareness, so that in a second step the system is able to restore the disturbed service.

This section discusses a reaction scenario for an automotive *fail safe mode*. This process occurs if the system is unable to restore security services for a sufficient level of functionality. The measures can be summarized as two steps. First the system should be able to shutdown the disturbed services completely or partly and second adapt other uncompromised services to reduce the impact of the error and the surface of attack (e.g. deactivation of external communication means, limitation of the driving speed or the engines revolutions per minute, audio or visual signaling of the error status). A complete system deactivation is infeasible since for obvious safety reasons the car needs to be movable even while in fail safe mode. The fail safe mode has to be used as a last resort since attacks could aim at leveraging from starting this mode in order to either take advantage of the partial security disabling or simply to limit the car performance and functionality.

This security architecture is aimed at limiting errors and their propagation. Our focus is to limit presence of single points of failure (SPOF) inducing latency and system blocking. Every ECU is independent in term of security decisions and communication channel establishment and does not rely on any central entity. Though in order to avoid too much redundancy in the system, centralized interfaces are provided in the architecture for cryptographic key or policy distribution.

F. SEIS Prototypes

To highlight the concepts mentioned above, two prototypes were developed. They both present use cases including on-board and V2X communications.

1) Automotive authentication with central trust-anchor for secure software download: With the ECUs' functionalities being increasingly integrated via software, the creation of new software-based functionality is taking up more and more of the development effort. As a result, the life cycle of automotive software is changing and approaching that of the regular software industry. The increasing integration of Internet connectivity is presenting new challenges in the way of an increased need for regular security updates but also offers novel ways of how to solve this problem. It is the goal of this demonstrator to show a cost-effective but secure way to implement remote software updates using wireless connections, such as the driver's domestic wireless LAN or cell phone. Furthermore, we also enable the driver to install additional features via an online marketplace.

In this demonstrator, we implement a proof of concept that facilitates secure software updates. To support software maintenance we continuously inspect the ECUs' software using software attestation techniques and thus create a verifiable software platform. If an ECU's firmware check yields to the result that it has been manipulated or just out of date, appropriate options can be presented.

2) Security proxy architecture for secure CE device integration: Consumer electronics (CE) devices are now belonging to our daily life and propose very innovative use cases when combined with a car. Their integration can prevent from expansive software and hardware updates. But to do so, more secure and deeper integration is necessary. Mobility and potential misconfiguration make them a big security threat. Due to safety and security reasons, direct communication between CE device and ECU is not allowed. Such a communication requires protocol decoupling at the network access level: a communication proxy. The proxy is in charge of integrating the CE device and enforcing the necessary security measures to maintain a sufficient security level.

This prototype proposes a CE adaptive security architecture enforcing decoupling of security mechanisms between inside and outside with performances suitable for infotainment use cases [18]. The proxy relies on the security middleware presented earlier. It adapts its security services based on the security capacities and the integrity measurement provided

by the CE device and provides cooperation mechanisms with ECUs for optimal security enforcement thanks to a in-band signaling protocol, supported by the middleware.

IV. Related Work

A. Security Communication Middleware

Traditional middlewares already provide security, e.g. message encryption and access control mechanisms [19]. But most of them do not fulfill the necessary functional and security requirements of automotive applications [20] at the same time. Besides, they usually do not propose suitable modularization in order to provide software/hardware-based security or different versions offering different security levels.

In recent years the avionics industry has been confronted with the same problems and is increasingly willing to intensify communication with the outside (e.g. ground-based communication partners) and to tether infotainment services to customers' CE devices. Despite few available research work the main approaches tend towards firewalls and physical network separation [21] and strong security protocols [22].

B. Automotive Security Projects

A few years ago, car makers and the research community realized the importance of automotive security and launched several projects for both in-car and V2X security. This section does not provide an exhaustive list of projects but presents some major ones. The SeVeCom project [23] addressed the security of future vehicle communication network and designed security mechanisms for encryption and authentication at the edge of the car. sim^{TD} [24] developed the associated V2X communication platform, but both of them did not consider on-board security otherwise.

OVERSEE [25] proposed a reduction of the number of ECUs to increase security and suggested a virtualization architecture; Super-ECUs host smaller application ECUs and virtualize the on-board network. EVITA [7] like SEIS aimed at securing the on-board network. It proposed a modular security framework: all ECUs are managed by a *Security Master* in charge of communication security and runtime monitoring. Additionally, this project developed its own versions of secure hardware platforms providing secure boot and secure storage. Albeit aiming to secure the on-board network like SEIS, these two projects (OVERSEE & EVITA) do not consider IP-based communications and rely on centralized architectures presenting a risk of SPOF.

V. Conclusion and Future Work

Automotive security has become essential, especially with the arrival of IP standards in car. In this paper we presented a security middleware extension which is independant from the middleware and which offers variable and adaptable level of security for on-board components and communications. Our approach proposes concrete mechanisms and runtime functionality to enforce in-car security for IP-based communication networks. The features of this security architecture have been implemented in two prototypes, presenting realistic and security-relevant automotive use cases.

In future work, we want to keep on specifying the security middleware management to integrate existing and car adapted standards for policy update, re-keying process or initial key distribution. In addition, we want to define a security driven migration strategy to allow the cohabitation of secure IP-based middleware with traditional bus technologies, offering a lower security level (CAN, MOST) and to potentially and partly integrate our architecture with existing non-IP-based middleware such as AUTOSAR.

Acknowledgment

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 (IP) 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] Karl Koscher, Alexei Czeskis, Franziska Roesner, Shwetak Patel, Tadayoshi Kohno, Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, and Stefan Savage: Experimental Security Analysis of a Modern Automobile. In Proceedings of IEEE Symposium on Security and Privacy, pp 447–462, IEEE Computer Society, 2010.

[2] Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, Stefan Savage, Karl Koscher, Alexei Czeskis, Franziska Roesner, and Tadayoshi Kohno: Comprehensive Experimental Analyses of Automotive Attack Surfaces. In Proceedings of the 20th USENIX conference on Security, pp 6–6, USENIX Association, 2011.

[3] Michael Glass, Daniel Herrscher, Herbert Meier, and Martin Piastowski: SEIS - Security in Embedded IP-based Systems. In ATZelektronik worldwide, vol. 2010-01, pp 36–40, 2010.

[4] Maier, A.: Ethernet - The standard for In-car Communication. In 2nd Ethernet & IP @ Automotive Technology Day, Regensburg, 2012 http://www.ethernettechnologyday.com/downloads/18_Alexander_Maier_-_BMW.pdf

[5] Danny Dolev and Andrew C. Yao: On the Security of Public Key Protocols. In Foundations of Computer Science, 22nd Annual Symposium on Computer Science, pp. 350 –357, Stanford University, 1981.

[6] Roland Bless, Gerrit Grotewold, Christian Haas, Boris Hackstein, Stefan Hofmann, Anke Jentzsch, Alexander Kiening, Christoph Krauß, Julian Lamberty, Michael Müter, Peter Schoo, Lars Völker, and Christoph Werle: A Security Model for Future Vehicular Electronic Infrastructures. In 8th Embedded Security in Cars, 2010.

[7] Benjamin Weyl, Marko Wolf, Frank Zweers, Timo Gendrullis, Muhammad Sabir Idrees, Hendrik Schweppe, and Yves Roudier: Secure On-board Architecture Specifications. Technical Report, EVITA Project, 2010.

[8] Oliver Bubeck, Jens Gramm, Markus Ihle, Jamshid Shokrollahi, Robert Szerwinski, and Martin Emele: A Hardware Security Module for Engine Control Units. In 9th Embedded Security in Cars Conference, 2011.

[9] R. L. Rivest, A. Shamir, and L. Adleman: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. In Communications of the ACM, vol. 21(2), pp 120–126, ACM, 1978.

[10] Neal Koblitz: Elliptic Curve Cryptosystems. In Mathematics of Computation, vol. 48(177), pp 203–209, 1987.

[11] Victor S. Miller: Use of Elliptic Curves in Cryptography. In Proceedings of Advances in Cryptology, pp 417–426, 1985.

[12] IEEE: IEEE 1003.1-2001 Standard for IEEE Information Technology - Portable Operating System Interface (POSIX(R)), http://standards.ieee.org/findstds/standard/1003.1-2001.html, 2001.

[13] Alexandre Bouard, Benjamin Glas, Anke Jentzsch, Alexander Kiening, Thomas Kittel, Franz Stadler, and Benjamin Weyl: Sicherheit auf Ebene der Applikation und ihrer Middleware, AP4.3., soon available in the TUM online library, 2012.
[14] Tobias Hoppe, Stefan Kiltz, and Jana Dittmann: Adaptive Dynamic Reaction to Automotive IT Security Incidents Using Multimedia Car Environment. In Proceedings of the 2008 The Fourth International Conference on Information Assurance and Security, pp 295–298, IEEE Computer Society, 2008.
[15] Stefan Axelsson: Intrusion Detection Systems - A Survey and Taxonomy. Technical Report, 2000.
[16] Tal Garfinkel and Mendel Rosenblum: A Virtual Machine Introspection Based Architecture for Intrusion Detection. In Proceedings of Network and Distributed Systems Security Symposium, 2003.
[17] Christoph Krauß, Frederic Stumpf, and Claudia Eckert: Detecting Node Compromise in Hybrid Wireless Sensor Networks Using Attestation Techniques. In Proceedings of the Fourth European Workshop on Security and Privacy in Ad hoc and Sensor Networks, vol. 4572 of Lecture Notes in Computer Science, pp 203–217, Springer-Verlag, 2007.
[18] Bouard, A., Schanda, J., Herrscher, D., Eckert, C.: Automotive Proxy-based Security Architecture for CE Device Integration. In 5th International Conference on Mobile Wireless Middleware, Operating Systems, and Applications, to appear, Springer, 2012.
[19] Jameela Al-Jaroodi and Alyaziyah Al-Dhaheri: Security Issues of Service-Oriented Middleware. In IJCSNS International Journal of Computer Science and Network Security, Vol. 11(1), pp 153–160, 2011.
[20] Douglas C. Schmidt, Aniruddha Gokhale, Richard E. Schantz, and Joseph P. Loyall: Middleware R&D Challenges for Distributed Real-Time and Embedded Systems. In SIGBED Rev., vol. 1(1), pp 6–12, 2004.
[21] Krishna Sampigethaya, Radha Poovendran, and Linda Bushnell: Security of Future E-enabled Aircraft Ad-hoc Networks. White Paper, American Institute of Aeronautics and Astronautics, 2008.
[22] N. Thanthry, M.S. Ali, and R. Pendse: Security, internet connectivity and aircraft data networks. In Aerospace and Electronic Systems Magazine, vol. 21, pp 12–16, IEEE, 2006.
[23] SeVeCom project - Homepage of the SeVeCom project: http://www.sevecom.org/.
[24] sim^{TD} project - Homepage of the sim^{TD} project: http://www.simtd.org/.
[25] OVERSEE project - Homepage of the OVERSEE project: https://www.oversee-project.com/.