Co-Authors

Prof. Dr. Martin Burgmer, Europe-University Flensburg

Dr.-Ing. Christian Knobloch, Knobloch & Gröhn GbR

Dipl.-Ing. (FH) Neofitos Arathymos, German Federation of Motor Trades and Repairs

Security Concept for an Interoperable Telematics Platform

September 2017

Professionally verified by Clemens Bergmann
c.bergmann@proventa.de
Proventa AG
Untermainkai 29
60329 Frankfurt am Main

Content

I. Introduction

Nowadays modern vehicles are connected (WAN-Connection) to various systems outside the core vehicle network like, vehicle-to-vehicle (V2V), vehicle to infrastructure (V2I) networks and/or also using the Internet for communication with different servers (service provider server, vehicle manufacturer server). Usually, these exchanges of data are controlled by a Telematics Control Unit (TCU) which serves as an information gateway in the vehicle, where it is connected to both the wireless link via a GSM module or even more sophisticated cellular communication technologies (UMTS, LTE) and internally to the vehicle's communication busses and to the physical OBD II connector or through a vehicle manufacturer's proprietary interface.

However, automotive security covers more aspects than only securing a TCU. It covers protection of all electronic control units (ECU), in-vehicle communication, and external communication against malicious encroachments by an attacker. Furthermore the implementation of third party application in a secure manner is also an important part of an overall security concept.

By providing an additional external communication to the vehicle data busses, the attack surface of the respective vehicle is enlarged. An overall security concept should address all the aspects.

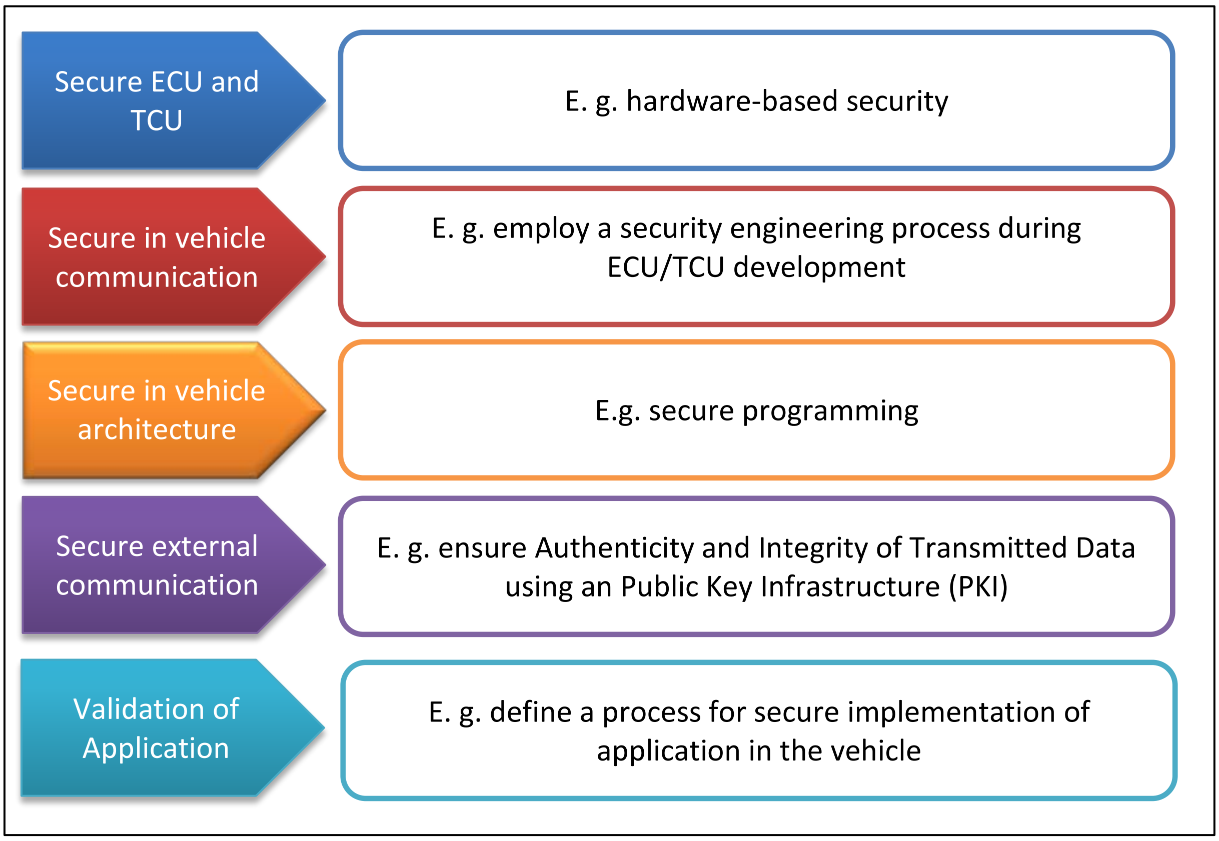
Secure ECU and TCU
E. g. hardware-based security
Secure in vehicle communication
E. g. employ a security engineering process during ECU/TCU development
Secure in vehicle architecture
E.g. secure programming
Secure external communication
E. g. ensure Authenticity and Integrity of Transmitted Data using an Public Key Infrastructure (PKI)
Validation of Application
E. g. define a process for secure implementation of application in the vehicle

Figure 1: Principle areas of the overall security concept

It should be noted that due to the development in the area of automated vehicles, the points “Secure ECU/TCU”, “Secure in vehicle communication” and “Secure in vehicle architecture” are already under development by the various vehicle manufacturers in dedicated working groups to ensure the vehicle integrity.

This document defines fundamental security requirements for an Interoperable Telematics Platform. It does not provide specific design or implementation guidelines, but an overview of the requirements which must be fulfilled by the respective vehicle manufacturer and the third party service providers in order to guarantee a specific level of security. Some fundamental requirements will be defined in the next Chapter.

II. Security Requirements (R)

In the following, we present some overall security requirements that need to be fulfilled for providing a secure Interoperable Telematics Platform.

Please note that the following requirements do not replace the need for specifying an overall security strategy. However, they can be understood as guidelines for developing a security concept.

Thus, this document offers a level of detail that ensures comprehensibility for non-experts in the complex field of IT-security whilst still citing the state-of-the-art technologies, procedures and institutions for the domain.

Especially the inclusion and referencing of Institutions like the “Bundesamt für Sicherheit in der Informationstechnik, BSI” who are known to update their security requirements constantly to reflect the fast technical evolution in the area of IT-security will keep the basic requirements and principles of this security concept always up to date.

Secure Communication between TCU and the backend servers

To secure the communication between the TCU and external interfaces, here called ‘backend servers’, the following six requirements are defined. In general, possible attacks on the communication can be conducted on two different layers:

- Communication between sensors, other ECUs and the TCU.
- Communication between the TCU and the backend.

Since the vehicle bus (and therefore the sensors and the ECUs) are standardized and well established, the definition of security requirements for the communication with the internal vehicle bus is out of scope in this report. This is in the responsibility scope of the respective vehicle manufacturer. It is obvious and well-known that similar security risks already exist in the automotive communication ecosystem which are already addressed by many vehicle manufacturers and supported by publicly funded projects [1] [2].

R1.1: Ensure Authenticity and Integrity of Transmitted Data

To prevent modification of data exchanged between the backend and the TCU or to prevent that fake data is received by the backend and subsequently processed, it is important that the integrity of the message is ensured and at the same time only an authorized message could have been sent. There are two main methods to provide integrity and authentication of a message:

- Message authentication code (MAC)
- Digital signature

Both methods share the characteristics that it is hard to generate a valid MAC or digital signature for a given message without knowing the secret token. Also, it is hard to find a so-called collision, i.e., a second message that has the same MAC or signature as an existing message with a valid MAC or signature.

A MAC is generated symmetrically, which means that the generation and the verification are both done using the same key. For constructing a MAC, either a hash-based (HMAC) or a cipher-based (CMAC) approach can be used. Since many modern automotive-dedicated controllers are equipped with an AES hardware accelerator, the AES-CMAC should be the first choice to ensure secure messages.

Digital signatures are based on asymmetric cryptography, i.e. a private key is used to generate the signature, while a public key (which is publicly available) is used to verify the signature. The advantage over a MAC is the fact that the public key must not be stored secretly which results in an easier key management. It is common to use signatures together with private-key certificates based on the X.509 standard (ISO 20828).

As a best practise for enhanced security there should either be an update/exchange-process for certificates in place and/or a possibility for the communicating parties to check the integrity of certificates online via the Online Certificate Status Protocol OCSP or related protocols like OCSP stapling/Multiple Certificate Status Request Extension.

R1.2: Ensure Confidentiality of Data

To ensure confidentiality of secret data sent exchanged with backend servers, like technical data in combination with personal data, the data stream must be encrypted with a secure encryption algorithm, such as the Advanced Encryption Standard (AES). Furthermore, a suitable mode of operation like CBC (Cipher Block Chaining) together with a MAC to ensure the integrity or an Authenticated Encryption like the GCM (Galois/Counter Mode) or CCM (Counter with CBC-MAC) must be used to authenticate and then encrypt.

R1.3: Mutual Authentication of Entities

For a secure communication connection between the TCU and the backend, a mutual authentication must be done prior to the data transmission to ensure all other security requirements. That means that both communication partners, like the TCU and the backend server are authenticating each other at the same time. This is usually done with a challenge-response algorithm based on asymmetric cryptography, with the help of a client and a server certificate similar to what is done in the vehicle's Transport Layer Security (TLS) protocol. If the communication parties share a common secret, the mutual authentication can also be done via a symmetric algorithm like secure variants based on the Needham-Schroeder protocols (The term Needham–Schroeder protocol can refer to one of the two key transport protocols intended for use over an insecure network)

R1.4: Ensure validity of Fresh Data Messages

An easily accessible attack vector is a replay attack which means that a message is recorded and played back later by an attacker. To avoid replay attacks on the communication between the TCU and the backend, it is necessary to add sequence numbers, timestamps or another variable component (like nonces) to the messages. Note that these sequence numbers or nonces need to be integrity-protected together with the message via a MAC or a digital signature, as described in Requirement R1.1.

The generation of the timestamps or nonce(s) must also be done in a secure manner. That means that timestamps must be calculated via a Real Time Clock (RTC) which cannot be tampered and must be synchronized with the backend server.

Nonce(s) must be generated by a cryptographic secure (pseudo) random number generator.

R1.5 Ensure Perfect Forward Secrecy

To ensure perfect forward secrecy of all communication, the system must generate session keys with a non-deterministic algorithm from long-term key material for each new session. After a session is closed, the used session key must be destroyed in order to prevent a decryption of previous recorded messages if the system is later accessed by an unauthorised user and the previous keys are extracted.

Through this requirement, it is ensured that an attacker cannot decrypt messages which were recorded in the past, even if he is able to break into the system and extract key material. By using a non-deterministic key generation algorithm, e.g. by using randomness for each key generation, an attacker cannot compute future session keys from the key material, leading to a system with the Perfect Forward Secrecy (PFS) properties.

R1.6 Usage of a Suitable Protocol for the Secure Communication

To secure the network communication and to comply with the above requirements, the TCU needs to make use of existing network protocols. Two good options are the Internet Protocol Security (IPsec) protocol and the Transport Layer Security (TLS) protocol.

IPsec can be used to implement an authenticated and encrypted communication between two network hosts. IPsec itself is, strictly speaking not a protocol, but a protocol suite, as it encompasses several protocols for particular purposes: for the key exchange, the Internet Key Exchange (IKE) protocol is used. If the communication between hosts only needs to be authenticated, the Authentication Header (AH) protocol can be employed. In case that content also needs to be encrypted, it is recommended to use the Encapsulating Security Payload (ESP) protocol that allows for both encrypted and authenticated communication. In fact, the AH protocol should be avoided due to its limited use. Additionally, it is important to never use the ESP protocol in the so-called encryption-only mode, but always with authentication being enabled. Otherwise, the protocol lacks defensive measures against message manipulation and attackers can attempt to re-route messages or manipulate targeted option or content fields of the transmitted messages.

The details on the secure configuration of IPsec exceed the scope of this document. The reader is therefore referred to the technical guideline TR-02102-3 of the Federal Office for Information Security (Bundesamt für Sicherheit in der Informationstechnik, BSI) [3]. As an alternative to IPSec, the use of Open Virtual Network Protocol (OpenVPN) could be considered, offering the advantages of a Virtual Private network and an enhanced stability for frequently changing internet connections.

TLS provides security not only between network hosts, but also between Transport Control Protocol (TCP) ports, i.e. between applications. The protocol has been developed over several versions and the latest version is 1.2 (as of 2015/02). It is recommended to always use the latest protocol version exclusively, because as for all protocols, a large number of practical attacks against the TLS protocol in earlier versions are known. To address several risks through the choice of an insecure cipher suite or parameters, the BSI has published a technical guideline for the secure configuration of the TLS, namely TR-02102-2 [4]. Furthermore, the draft for TLS 1.3 [5] shows a large step towards a simpler protocol with fewer, but still secure algorithms and parameters. For example, the data stream compression functionality which can lead to the so-called entropy attacks will be removed for TLS 1.3. Also, support for all non-AEAD cipher suites, which were responsible for several attacks on TLS in the past, was removed.

For the TCU, TLS is more relevant than IPsec, because for the use cases of a TCU, the communication on the application layer needs to be secured rather than on the network layer. Therefore, Table 1 gives a recommendation for basic TLS 1.2 configuration parameters, whereby Table 2 gives recommendation for extended TLS 1.2 parameters.

These recommendations are considered safer due to a doubled symmetric AES key length and longer SHA-2 hash values. The choices of the parameters are mainly related to the used controller, the supported hardware accelerators and the targeted performance.

	Key Agreement		Encryption	Mode	Hash	Use Until
TLS_	**ECDHE_ECDSA_**	**WITH_**	**AES_128_**	**CBC_ GCM_**	**SHA256**	**2020+**
	ECDHE_RSA_	**WITH_**	**AES_128_**	**CBC_ GCM_**	**SHA256**	**2020+**
	DHE_DSS_	**WITH_**	**AES_128_**	**CBC_ GCM_**	**SHA256**	**2020+**
	DHE_RSA_	**WITH_**	**AES_128_**	**CBC_ GCM_**	**SHA256**	**2020+**

Table 1: Recommended Basic TLS configuration parameters

Key Agreement			Encryption	Mode	Hash	Use Until
TLS_	ECDHE_ECDSA_	WITH_	AES_256_	CBC_ GCM_	SHA384	2020+
	ECDHE_RSA_	WITH_	AES_256_	CBC_ GCM_	SHA384	2020+
	DHE_DSS_	WITH_	AES_256_	CBC_ GCM_	SHA384	2020+
	DHE_RSA_	WITH_	AES_256_	CBC_ GCM_	SHA384	2020+

Table 2: Recommended Extended TLS configuration parameter

Security guidelines for the TCU

In the following section, we present some overall security requirements that need to be taken into account for a secure TCU.

R2: Unique Cryptographic Identities

A key aspect of a secure TCU is the injection of unique identities which are required for most of the security requirements. The usual way for this to be achieved is the employment of certificates based on strong asymmetric cryptography like RSA or ECC.

The certificates bind a public/private key pair to a specific ID which belongs to one unique device. In this way, every CCU has its own key pair and is unambiguously identifiable.

On the other hand, every TCU stores the public key of the backend server in order to perform an authentication protocol prior the communication.

R3: Hardware-based Security

Since the TCU is built into the vehicle and not isolated in a secure environment, an attacker has physical access to the device. This special aspect has to be addressed by the security design and lead to several fundamental requirements towards the hardware used.

R3.1 Key Storage

There are various ways to store key material securely on the TCU. In this report, four different methods are proposed:

- On-chip One-Time-Programmable (OTP) space
- On-chip SRAM storage spaces
- Encrypted storage of private keys

- Usage of Hardware Security Modules to protect cryptographic keys (Compare Requirement R3.3)

A distinction must be drawn between public and private keys. Whereby a public key must be stored in a tamperproof way, or at least be tamper evident, a private key must be stored in a way that an ensures that an attacker cannot extract the key.

As a result, the public key of the backend servers must be securely stored by the TCU. Since the key is publicly known and is not confidential, a possible attacker can extract the key without any harm to the overall security of the system. Nevertheless, the public key must be stored read-only, which means that the key must be secured against unauthorized modifications, which could give the attacker the possibility to inject his own public key and lead to a successful impersonation attack.

To fulfil this requirement some fuse technology is often used where the fuse can be burned with the public key and is automatically locked after the write process. Another way can be the use of a SRAM-based key store whereby the key can also be updated if the update process is started with the correct credentials. Since a SRAM chip is erased at a power loss, the memory should be buffered by an additional battery.

A higher security level is required for the secure storage of the TCU's private key. Since this key is used for all other cryptographic operations like the authentication or for the encryption of the data stream the key must be stored confidentially. One way to protect the key is to encrypt the updateable private key with a master key which is generated at production and then made unreadable by blowing some specific fuses. Another technique is to have some integrated memory which is hardened against unauthorized access and physical attacks respectively, and where it is difficult to extract the key. Such techniques are offered by some chip manufacturers to additionally secure key material from unauthorized extraction.

However, the generation - especially of asymmetric key material - is a complex task and is likely to take a significant amount of time at the end-of-line test of a production phase, which might be unacceptable for the TCU manufacturer. Therefore, a possible alternative is that the cryptographic keys are be generated by an external unit, for example a Local Registration Authority (LRA) that resides at end-of-line of production and can communicate with the TCU. After generating key material by the LRA, the LRA injects the cryptographic material in the TCU, where it is stored. The private keys have to be permanently removed from the LRA to avoid a leakage of confidential data.

R3.2 Hardware Support/Acceleration for Cryptographic Algorithms

For nearly all security related operations some cryptographic algorithms are used. Since these algorithms are costly in terms of memory and time, many automotive-specific controllers support some of the most common algorithms like the AES. Due to the fact that many cryptographic protocols require a secure random number generator, the hardware should have some built-in true random number generator (TRNG) functions. Since most of the protocols only require a deterministic random bit generator (DRBG) which can be built efficiently in software, the TRNG, which usually has a low throughput, must be used to seed the DRBG with a real random value in order to get pseudo-random numbers with sufficient entropy.

R3.3 Usage of Hardware Security Modules

The most secure approach to protect cryptographic keys is to employ a dedicated security controller (Hardware Security Module) which provides protection against physical extraction of cryptographic keys. HSMs are integrated circuits specifically developed and designed for security use-cases. Typically, implementations range from smart cards used for identification and authentication purposes, such as, national identification cards, to Trusted Platform Modules which are Hardware Security Modules for personal computers. HSMs typically consist of a CPU core, different types of data storage (e.g., RAM, ROM, Flash), a memory protection unit, a memory encryption unit, sensors, cryptographic accelerators, and further peripheral components. Most HSMs employ sophisticated countermeasures against physical attacks, such as active sensors to detect fault and glitch attacks, and also employ cryptographic implementations which are protection against side channel attacks.

R4: Use Strong Cryptography with Sufficient Key Lengths

To implement the security mechanisms, it is necessary to use only strong and well-standardized algorithms with common recommended key lengths. One widely accepted recommendation of algorithms and key lengths is the technical guideline TR-02102-3 of the Federal Office for Information Security (Bundesamt für Sicherheit in der Informationstechnik, BSI [6].

This guideline defines methods and parameters which have to be used for a specific security level and specific lifetime of a device or system respectively. For example, the encryption algorithm AES is the only recommended block cipher and should be used with the Galois-Counter Mode (GCM), Cipher-Block Chaining (CBC), or Counter Mode (CTR). For asymmetric cryptographic, RSA with at least 2048 bits or ECC with the brainpool curve P224r1 (standard curves in elliptic curve cryptography) or larger is recommended.

For the development of a new TCU the recommendation for algorithms and key sizes must be used in order to design a secure system.

R5: A robust TCU Operating System

The operating system (OS), usually a Linux-based OS, used on the TCU should be robust against typical vulnerabilities. Two recommended kernel modules which introduce some advanced security policies like Mandatory Access Control (MAC) are SELinux, and Apparmor. Additionally, a hypervisor-based approach shall be implemented to isolate security operations from other task on the controller (A hypervisor is a function which abstracts i.e. isolates, operating systems and applications from the underlying computer hardware. This abstraction allows the underlying host machine hardware to independently operate one or more virtual machines as guests, allowing multiple guest virtual machines to effectively share the system's physical compute resources, such as processor cycles, memory space, network bandwidth etc.).

Like any other security & safety relevant software component, also the OS of the TCU should be updateable with security patches to counter detected security issues.

Given the fact that these patches affect the safety & security of the car on the road and thus are related to type approval, it is recommended to have a neutral authority nominated by the type approval authority crosscheck the patch prior to it's installation.

R6: Employ a Security Engineering Process during ECU/TCU Development

During all phases of the design and development of the TCU, security aspects have to be considered. In detail, a security engineering process has to be introduced in order to ensure that all security critical architecture decisions are well evaluated and are correctly implemented.

This includes regular meetings and discussions with representatives of all the development departments involved in the security development as well as regular reviews of the architecture and the implementation itself.

R7: Perform Penetration Tests of the TCU

Even though a well-documented design process with a focus on the security and mandatory reviews is applied during the development of the TCU, some critical flaws and weak spots can appear in the specific device. Especially in the production phase of the device, which is usually from an external manufacturer or supplier, vulnerabilities like open debug ports or insecure debug routines can appear. To close the gap between the designed security architecture and the manufactured device, a rigorous penetration test of the TCU must be performed. Ideally, this test should be done by independent and experienced security experts to ensure a good level of security.

R8: Control communication to the Vehicle Bus

To inspect and control messages generated by the TCU for the vehicle bus networks, every message which is generated needs to be controlled and validated, i.e. basic firewalling mechanisms/Hypervisor should be employed. This includes message inspection based on a whitelist which only allows messages which are explicitly listed in this list. The remaining messages are dropped and not transmitted to the vehicle bus.

A more sophisticated approach is the implementation of a Stateful Package Inspection (SPI) which also evaluates the time and state aspect of the messages and will either allow or drop messages based on the current protocol state.

To this end, before a message is passed to the device driver which delivers the message to the OBD-bus, it needs to be inspected by a dedicated security component. This security component could be realized as an own kernel component or module, or as a dedicated security application which is able to inspect every message delivered to the OBD interface.

However, a more secure solution is the realization of the firewall component on a separate compliant microprocessor which is isolated from the remaining application processor where the Linux-OS is executed. This means that the different networks, namely the in-vehicle bus networks like CAN, FlexRay or Ethernet are connected to the separate microprocessor which communicates with the application processor via inter node communication. As a result, the in-vehicle network is clearly isolated and communication to the external network using e.g. WiFi, GSM or LTE is achieved via dedicated firewall components.

R9: Secure Boot

To ensure a trustworthy and secure system, the boot process of the firmware must be secured. Usually, secure boot is ensured by a chain of trust whereby the integrity of each component is validated by the previous component. The validation of the firmware is done by a cryptographic hash value of the code which is compared to a pre-configured stored value.

The initial component works as a core root of trust for measurement and must be stored in a tamperproof way inside the chip, ideally inside a specialized HSM core of the controller. This root of trust is triggering the bootloader and initiating the boot process. Through this measure, the actual firmware and application is validated and run in a trustworthy state.

In order to enable a flexible secure boot (e.g. for security updates), a secure reference update mechanism (e.g. based on a shared secret or a public key scheme) is required, which allows firmware updates in the field.

R10: Secure programming

To enable the possibility of updates of the TCU's firmware later during the vehicle's life, a secure programming process based on the type-approval requirements (Euro 5- and Euro 6) and the standard ISO 18541 with the additional requirements for anti-theft related repair and maintenance information shall be implemented.

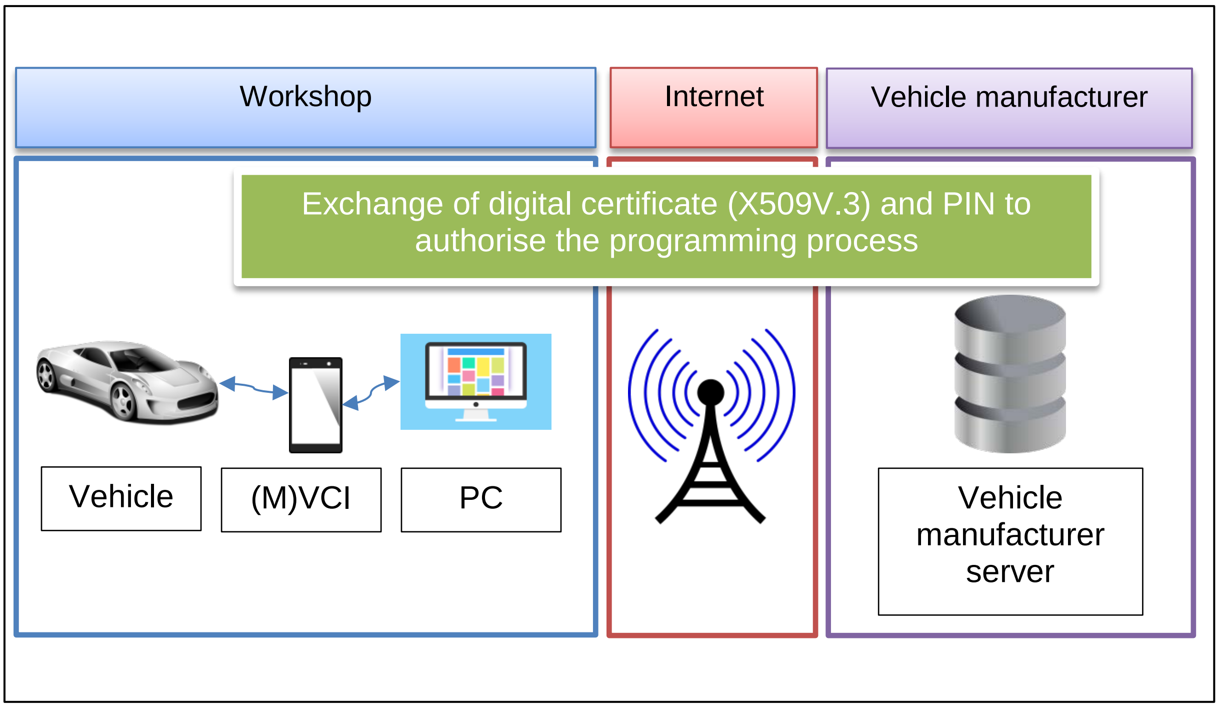
Workshop
Internet
Vehicle manufacturer
Exchange of digital certificate (X509V.3) and PIN to authorise the programming process
Vehicle
(M)VCI
PC
Vehicle manufacturer server

Figure 2: Security access process for secure programming

The integrity and authenticity of a new firmware image must be validated by strong cryptographic mechanisms like a digital signature based on RSA or ECC. Furthermore, the secure boot process must be updated which means that the stored hash value of the firmware and application code must be updated inside the protected memory area.

Particular cautions must be taken if the new image is not validated successful during the update process. In this case, the system must boot the last secure firmware, which means that the new firmware image has to be stored in an empty flash area on the chip and if (and only if) validated successfully the bootloader must be updated with the address of the new image. That means that the chip must be equipped with enough memory for storing two times the firmware concurrently.

R11: Secure Confidential and Private Data on the CCU

To ensure the privacy of the driver and secure confidential data, data shall be stored encrypted on the TCU. This could be some geo-information data, cellphone identification data or also some buffered application data, like messages from the infotainment domain. Furthermore, unused temporary data should be erased as soon as possible.

R12: Secure Physical Access to the CCU

Many publicly known attacks on embedded devices were done (or at least with the help of) open debug ports like JTAG or UART [7] or poorly protected external interfaces and memories. Therefore, these attack vectors should be carefully considered and must be evaluated by a mandatory penetration test described in Requirement R7.

R12.1: Secure Debug

Embedded processors in general and in particular TCUs, have several accesses and debug ports and routines. These are essential during the development and production phase where the correct chip behaviour and functionality has to be evaluated. However, at the same time, these debug ports can give an attacker confidential information or in the worst case, full access to the chip and the running application. Therefore, all interfaces which are not required after production, e.g., JTAG, SPI or UART, must be deactivated and the corresponding pins shall be physically disconnected. If such a debug port is required, for example if a controller has to be replaced and the error must be examined by the manufacturer, this interface must be protected by strong cryptography.

R12.2: Authentication of External Interfaces

All access from external interfaces, such as GSM, UMTS or LTE should be authenticated by a mutual authentication scheme between the TCU and the backend as described in Requirement R1.3.

R12.3 Securing the Memory

To restrict the reading of the flash memory, the whole flash should be encrypted with a symmetric block cipher which is usually supported by current automotive specific controllers. Since the flash memory is a permanent memory, the attacker can use various methods to extract the data on the flash chip like invasive attacks which destroy the chip. In contrast, the RAM is automatically erased at power-off. Therefore, it can be sufficient to deactivate all debug accesses like JTAG to protect the RAM, since the attacker can use only attack vectors which leave the chip and the power connection intact. Nevertheless, if a RAM encryption is supported by the used controller, this feature should be used to prevent more sophisticated attacks like the cold boot attack [8].

Security guidelines for the backend servers

In the following, we present some overall security requirements that need to be taken into account the backend servers.

R13: Employ mechanisms in the backend servers to Isolate Malicious TCUs

Beside the in-vehicle TCU itself, the backend servers must be protected against potential attacks as well.

One major requirement for the backend must be the isolation and the secure handling of a malicious TCU. In detail, this should include a logging of any potential attack on the network and backend as well as active measures to force the malicious TCU into a fail-safe state like a forced firmware update, a TCU reset or a deactivation of the TCU with a message to the driver or infotainment system respectively that the TCU has been compromised and should be examined in a workshop. At least, the TCU should be isolated from taking part in the communication by revoking any certificates that are potentially being used.

R14: Establish a Private Communication Network

A private communication network should be established by assigning private IP addresses to the TCU. This means that the TCUs are not directly reachable from the Internet but rather only connected to the isolated backend system. This could be done by using Network Address Translation (NAT) and by providing a private Access Point Name (APN) which is used by the integrated SIM.

In contrast to a public APN of a mobile communication provider, which usually allows direct access to the Internet, the private APN allocate the TCU's SIM to a private network which is separated by the Internet. By using NAT, the backend maintainer assigns only private IP addresses for the TCUs which are non-routable in the global Internet. Additionally, the router which is connected to the TCU must be configured in a way that the particular TCUs are isolated and cannot send packets directly to each other. Thereby, a TCU has just one single logical connection directly to the backend server which can allow further connections like a link to a music streaming service for the infotainment system or a link to the navigation service provider in order to allow map updates.

The use of NAT was a common technology in the era of IPV4 and provided along with an improved use of the increasingly rare IPV4 address space an enhanced security. This makes this approach useful even for low cost TCUs that currently only support IPV4.

However, with new and more powerful TCUs entering the market, a strict use of IPV6 throughout the vehicle is envisioned that together with a sound and solid firewall-implementation will offer an even higher level of security and driver safety.

It is recommended to rely as much as possible on encrypted communication over the "public" Internet and to restrict the VPN-traffic -which should nevertheless be encrypted too - to the really security relevant communication.

III. Process for secure implementation of application in the vehicle

In the following, we present some overall security requirements that need to be fulfilled for providing a secure Interoperable Telematics Platform.

Please note that the following requirements do not replace the need for specifying an overall security strategy. However, they can be understood as guidelines for developing a concept for checking and implementing also third party applications into the vehicle-Telematics Control Unit.

It should be reiterated that security is not a state, but – especially with the long lifetime of a vehicle on the streets in mind – a process.

Thus, every software component (from the applications over the platform components like hypervisors or firewall down to OS of vehicle TCUs must be enabled to receive security updates/patches over the lifetime of the car to counter known vulnerabilities. By contracts, app developers as well as OEMs, manufacturers of TCUs and vendors of other security relevant components etc. must be held accountable to deliver security patches as soon as possible.

In the same way, certificates must be made updateable/exchangeable via a defined and secure process and it might even be necessary that security relevant hardware components might need a periodical replacement e.g. during periodical type inspection so that an improved vehicle hardware can run improved security software (e.g. with higher key lengths) to counter the attacks made possible by improved server hardware used by attackers.

Because all of this measures affect the vehicle's road safety, the involvement of neutral institutes involved in type approval is recommended to verify every replacement/update prior to installation in the car.

The security concept consists of three parts.

1. The first part is, that the vehicle manufacturer has to provide all necessary information to all service providers in order to develop robust applications.
2. The second part is, that a vehicle-manufacturer specific validation process shall be performed to secure that applications follow certain rules.
3. Finally a secure implementation method to the TCU is needed.

In the following table the above mentioned parts are described.

PART 1	
INFORMATION	
DEFINITION	Manufacturers shall provide all necessary information for the development of applications
DESCRIPTION	The access to the required information to develop the different applications and the business to business contract with the respective service provider shall be published in one of the official websites of the respective manufacturer, e. g. the website according to ISO 18541. Following information shall be made available on the website: • Variable VIN-based data list • Available vehicle (standard) API-description • Developer-Guidelines • Validation-Guidelines • Set of test data

PART 2	
VALIDATION	
DEFINITION	Manufacturers shall provide a validation procedure for the development of applications from service providers
DESCRIPTION	The validation process of applications from third parties shall be done only by the vehicle manufacturer in order to ensure manufacturer internal security and safety requirements. The validation process and the business to business contract shall be published in one of the official websites of the respective manufacturer, e. g. the website according to ISO 18541. The validation process shall be done by using existing ISO/CEN standards or state of the art methods. The manufacturers shall at least distinguish between three security categories:

	Category 1: High safety level Applications need time critical/high available data and/or information coming from the different powertrain electronic control units and/or radar/camera systems. As an example the following applications may fall under this category: Application for car-to-car-communication Application for CO_2 monitoring Application for programming of one or many electronic control units in the vehicle Application for pay how you drive Category 2: Medium safety level Applications need data and/or information coming from all electronic control units in the vehicle. As an example the following applications may fall under this category: Application for diagnostic of the vehicle’s electronic control units Application for pay as you drive Category 3: Low safety level Applications do not need data and/or information coming from all electronic control units in the vehicle. As an example the following applications may fall under this category: Application in the area of media Application in the area of navigation The definition of the safety category shall be done by the respective manufacturer. After the finalisation of the validation process - regardless whether tests have been passed or not - the respective manufacturer shall send a detailed report to the service provider. Watchdog functionalities shall be included in all applications.

The implementation process shall follow R 10.

IV. Abbreviations

AES	Advanced Encryption Standard
AES-CMAC	Advanced Encryption Standard Cipher-based Message Authentication Code
APN	Access Point Name
CBC	Cipher Block Chaining
CBC-MAC	Cipher Block Chaining Message Authentication Code
CCM	Counter with CBC-MAC
CCU	Communication Control Unit
CTR	Counter (mode)
DRBG	Deterministic Random Bit Generator
ECU	Electronic Control Unit
GCM	Galois/Counter Mode
GSM	Global System for Mobile communication
HSM	Hardware Security Module
LTE	Long-Term Evolution
NAT	Network Address Translation
OBD II	On-Board Diagnostics II
SIM	Subscriber Identity Module
SPI	Stateful Package Inspection
TCU	Telematics Control Unit
TRNG	True Random Number Generator
UMTS	Universal Mobile Telecommunications Service
V2I	Vehicle to Infrastructure
V2V	Vehicle to Vehicle
WAN	Wide Area Network

V. Bibliography

[1]
K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham und S. Savage, „Experimental Security Analysis of a Modern Automobile,“ in IEEE Symposium on Security and Privacy, Oakland, 2010.

[2]
S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner und T. Kohno, „Comprehensive Experimental Analyses of Automotive Attack Surfaces,“ in SEC'11 Proceedings of the 20th USENIX conference on Security, Berkeley, 2011.

[3]
BSI, „TR-02102-3 Kryptographische Verfahren: Empfehlungen und Schlüssellängen. Teil 3 - Verwendung von Internet Protocol Security (IPsec) und Internet Key Exchange (IKEv2),“ 2014.

[4]
BSI, „TR-02102-2 Kryptographische Verfahren: Empfehlungen und Schlüssellängen. Teil 2 - Verwendung von Transport Layer Security (TLS),“ 2014.

[5]
E. R. T. Dierks, „The Transport Layer Security (TLS) Protocol Version 1.3,“ 2015.

[6]
BSI, „TR-02102-1 Kryptografische Verfahren: Empfehlungen und Schlüssellängen,“ 2014.

[7]
B. Jack, „Exploting Embedded Systems,“ in Blackhat Amsterdam 2006, Amsterdam, 2006.

[8]
J. A. Halderman, S. D. Schoen, N. Heninger, W. Clarkson, W. Paul, J. A. Calandrino, A. J. Feldman, J. Appelbaum und E. W. Felten, „Lest We Remember: Cold Boot Attacks on Encryption Keys,“ in Proceedings 17th USENIX Security Symposium (Sec ’08), San Jose, 2008.