Cybersecurity Evaluation of Automotive E/E Architectures

Martin Ring
Bosch Engineering
Abstatt, Germany
martin.ring@de.bosch.com

Davor Frkat
Bosch Engineering
Vienna, Austria
davor.frkat@at.bosch.com

Martin Schmiedecker
Bosch Engineering
Vienna, Austria
martin.schmiedecker@at.bosch.com

ABSTRACT

The number of connectivity features of a modern car have expanded tremendously in recent times, including convenience applications over local wireless networks and back-end-connections over mobile networks. Additionally, the amount of exploited vulnerabilities in recent years has increased [16], and as such it is a paramount requirement to keep cybersecurity on pace with the additional demands. This is critical for new and upcoming features to prevent software vulnerabilities and later possibly incidents.

In this work, we present cybersecurity mechanisms in place today, and the upcoming mechanisms planned for vehicles. The results of our survey show that the industry currently lacks thorough cybersecurity concepts, and there is no common ground for cybersecurity in series vehicles. While the safeguarding of critical diagnostic functions is mandated by standards (e. g., ISO 14229 [3]), even these mechanisms are rarely implemented. Some implementations are even flawed, or the used protection mechanisms are too weak. For the E/E-Architectures and ECUs in development, the outlook is far better. Signed updates, firewalls, secure back-end communication and domain separation are features that we believe will become common for the majority of devices currently in development. Advanced features such as secure and/or authenticated boot, intrusion detection systems or encrypted and authenticated CAN messages are still rarely seen features even for cars of the near future.

CCS CONCEPTS

• **General and reference** → **Surveys and overviews**; • **Security and privacy** → *Authentication*; *Access control*; *Authorization*;

KEYWORDS

Automotive Security

ACM Reference Format:
Martin Ring, Davor Frkat, and Martin Schmiedecker. 2018. Cybersecurity Evaluation of Automotive E/E Architectures. In *Proceedings of ACM Computer Science in Cars Symposium (CSCS 2018).* ACM, New York, NY, USA, 7 pages. https://doi.org/10.475/123_4

1 INTRODUCTION

Based on recent attacks on vehicles that have received large coverage in media, in particular Charlie Miller's and Chris Valasek's attacks on a Jeep [12, 13], the need for additional security mechanisms is now clearly visible. As cars are further connected, the security requirements rise in the same way as their connectivity features. Possible pitfalls have been shown to, e. g., the BMW AG with an attack on their connected services [19], which was only possible because they had not activated encryption for the communication to their backend and the cars used identical keys across the whole brand that were stored on every ECU. Another vulnerability of a big OEM hurt Mitsubishi , they used the Wi-Fi of the car for remote features. The PSK of the Wi-Fi was too short, this allowed a fast offline computation followed by an attack on, e. g., the alarm system that was deactivated [10].

To give an extensive overview on the state of the art we will describe in the following the relevant security mechanisms which are observable in the market as of today as well as further mechanisms we deem relevant from our point of view. We further elaborate on the perceived market penetration of the presented mechanisms. We will also describe to what extent we see them for upcoming ECUs and E/E architectures currently in development.

Within the scope of this analysis is a generic vehicle, consisting of multiple ECUs, different bus systems for connectivity, and the data and wireless emissions it produces and processes per-se on a regular basis. We do not consider additional components that make up the entire vehicular ecosystems to be within the scope, such as, for example, smartphone applications, workshop equipment or external databases that process telemetric data.

2 CYBER SECURITY MECHANISMS

While PC-IT systems enjoy a multitude of protection mechanisms such as antivirus software, firewalls as well as secure update mechanisms, Table 1 gives a market overview of the available and planned security mechanisms to mitigate vulnerabilities for vehicles from the point of view of an automotive supplier.

It is important to keep in mind that one secured device (e. g., gateway or body controller) inside a car does not secure the car as a whole. Only if a complete chain/domain, or even all car components ranging from every sensor to data visualizations, are secured with similar levels a car can be referred to as a secure car!

Table 1 shows how far security features are already used in cars today and how they evolve until 2023 according to OEM requests we, as an automotive supplier, received. We classified the implemented and planned security mechanisms of OEM requests into four groups that illustrate four levels of usage, in ascending order:

- None: there is hardly any / no device inside cars across all / almost all OEMs

- Seldom: there is at least one device inside car across a minority of OEMs
- Many: there are many devices inside cars across the majority of OEMs
- Common: almost all devices inside cars across most OEMs uses this feature

2.1 Domain Separation

Domain separation was until recently mainly a focus for European manufacturers and primarily driven by the limited data rates of the used bus systems. With ever increasing numbers of connected components per bus, adding additional busses to increase the overall bandwidth was a viable solution. However, this also had the beneficial side-effect of enabling effective security features, when the domains are properly separated and the cross domain traffic is monitored and filtered. It should be noted that this feature is only as strong as its weakest link, when one ECU allows the spanning of multiple networks without filtering, the added security is lost.

2.2 Public-Key-Infrastructure (PKI)

A PKI is used to build a chain of trust that allows the different stakeholders such as manufacturers and suppliers to closely monitor and restrict the possibilities of interactions of ECUs with each other, a tester or the backend. A PKI is also the foundation for a secure authentication of hosts and initial integrity checks of their communication. Core component is a trusted storage, to store the root certificates that are allowed to update software or communicate securely. Depending on the scope of the PKI it could be for all components in a car, for manufacturers and all their cars, for an administrative domain of different cars, as well as regional or otherwise groupable sum of devices. Considered current-best-practice for remote authentication, the capabilities of a PKI can be very flexible regarding the revocation of access rights, granularity of permissions and their scope.

However, the downside of operating a PKI is that they are costly, and the flexibility comes with the possible issue of increased complexity. Trust anchors such as root certificates have to be already in place before signatures can be verified, processes are needed to revoke certificates and to check their validity, and if the private key of a root certificate becomes compromised one needs mechanisms in place to change them in all affected devices. Especially the aspect of verifying validities should be referred as critical because without having a trustful time source on every device which cannot be manipulated certificates will lose a lot of benefits. Standard protocols for time synchronization, e. g., NTP, GSM, GPS or similar are not feasible for this task form a security point of view [7]. Without the option to check validity of crtificates aspects like time-limited feature activation or avoiding the execution of outdated software is not possible.

2.3 Hardware Trust Anchor – Hardware Security Module

Multiple types of hardware trust anchors are available, with different names and a wide range of functionality. Usually they should have a dedicated area in hardware as software can be modified at runtime, and a security enclave that protects sensitive material like cryptographic operations or stored keys. There is however no norm or formal definition on the specifics for the terms HTA and HSM. Every supplier can basically define its own silicon as HSM, as soon as any kind of cryptographic operation is running on the chip. That means during the vendor selection phase, an OEM has to verify every technical statement of the possible suppliers in detail. This lack of standardizations often leads to misleading interpretations from vendors if a customer asks, for example, if an HSM is inside. Occasionally we have already encountered HSM functionalities that are insecure and useless, e. g., having a secret key stored openly in RAM during runtime, or only allowing insecure modes of operation for the AES algorithm.

Additionally, OEMs should ask suppliers if features from advertisements and marketing material supported by the hardware are already useable, or more precisely useable for the OEM. Often we have had cases where, e. g., elliptic curve cryptography was openly advertised, but in detail it turned out that the curves used were referred to as broken, or additional licenses to use specific secure curves were needed that were still under negotiation between the OEM and the rights owner.

After all, an HSM is also just a piece of embedded software running on (dedicated) hardware with limited access rights, with the additional benefit that it is protected from inference and has only very few interaction features. These are usually sensitive in nature such as *encrypt*, *verify* or *create key*. OEMs should also verify that the software to use the HSM functionality is already fully implemented, usable, and tested. Test reports or even a certification or confirmations about, e. g., fulfilled NIST standards in specific parts such as for AES or other building blocks are very useful. Hence we strongly recommend that OEMs should involve not only their purchasing department during vendor selection phase, but also their security experts.

HSMs today only make sense in the context of the whole car, i. e., as soon as a sensor, a minor device or similar is not protected by HSM trust anchor features such as message authentication or secure boot, attackers can compromise the unprotected device and use it as an entry point or stepping stone for further attacks. This was used for example in the Jeep hack by Charlie Miller and Chris Valasek [12]. That's also the reason that currently fully-featured HSM are seldom in use, because OEM have to redesign the whole sensor - actor path with more secure (which means more expensive) hardware and software.

As one of many HSM examples the EVITA project [5] defined three levels for their HSM interpretation, light, medium and full, and distinguishes them by their capability to compute asymmetric ciphers, random number generation, available counter modules and if they have a dedicated programmable ECU. An overview of the different components, which are usually implemented in software with corresponding hardware acceleration, can be seen in Figure 1. Similar approaches of classification exist for most vendors of HSMs.

With the recent attacks in 2018 against Intel and other CPUs using side-channel attacks named *Spectre* and *Meltdown* [9], it has to be said that HSMs are no panacea in themselves.

Table 1: Cybersecurity Mechanisms for Vehicles

Feature	2018	2023
Domain separation (2.1)	Seldom	Common
PKI (2.2)	Seldom	Many
Hardware Trust Anchor and/or HSM (2.3)	Seldom	Many
AUTOSAR (2.4)	Seldom	Many
Signed SW Updates (2.5)	Seldom	Many - Common
Secure Diagnostic Services (2.6)	Seldom (with weaknesses)	Many
Secure Boot (2.7)	Seldom	Many
Authenticated Boot (2.7)	Seldom	Many
Secure Communication with Backend (2.8)	Many	Common
Secure Car Internal Communication (2.9)	None	Many
Firewall (2.10)	Many	Common
IDS/IPS (2.11)	None	Seldom-Many
Wi-Fi/Bluetooth Security (2.12)	Common (with weaknesses)	Common

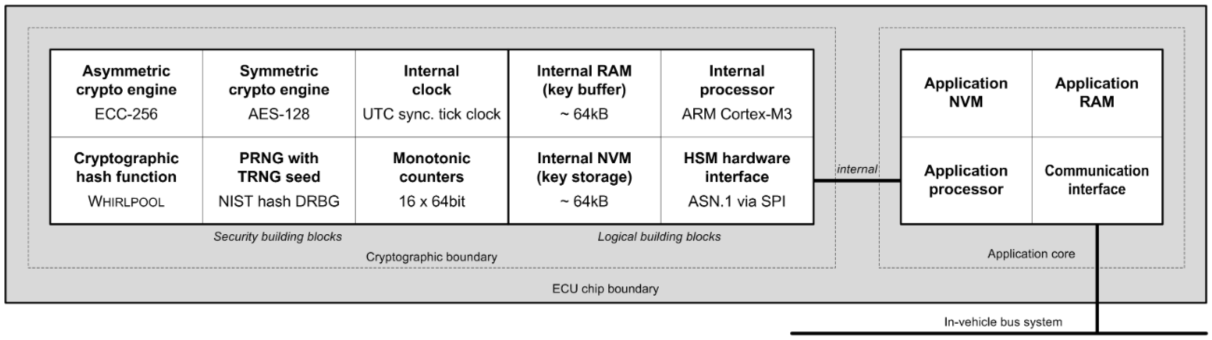

Figure 1: Exemplary system architecture of an HSM [23]

2.4 AUTOSAR

AUTOSAR is a collaboration of OEMs and suppliers to standardize and establishe a common software architecture for ECUs. The layered architecture, as shown in Figure 2, improves the reusability of software. Since version 4.2, AUTOSAR has defined all necessary security modules for Secure Onboard Communication (SecOC) – authentic communication integrated in the AUTOSAR-Stack, as well as the Crypto Service Manager (CSM) and the Crypto Abstraction Library (CAL) [1]. The offered documents by AUTOSAR are only definitions how these modules shall interact and include, they don't contain, e. g., the implementations of cryptographic primitives. Figure 3 shows the levels of abstraction for the AUTOSAR crypto interface specification.

Cryptography as such becomes part of the basic software, and allows for a diverse set of security mechanisms. Both hard- and software implementations of cryptographic algorithms are feasible. The upcoming release of adaptive AUTOSAR 18-10 (expected to be October 2018) will also include security properties (the final specification of the *Security Manager*) for high-performance computing and faster communication.

2.5 Signed Updates

To hinder an attacker from installing and executing unauthorized code on vehicular components such as ECUs and VCUs, updates should be digitally signed using internationally standardized cryptographic algorithms. This allows the verification of the authenticity and integrity of software updates. An exemplary signed update scheme is shown in Figure 4. To verify that the software is coming from an authorized source the component verifies the software update using public-key cryptography, e. g., RSA or ECC against a signature. This prevents an attacker from installing a modified software update, changing functionality, or reverting firmware versions. This is common practice for all major PC-based operating systems such as Windows, Linux and macOS, as well as smartphones and modern gaming consoles.

To guarantee this security the base for the verification (keys/certificates) has to be secure. The public keys/certificates on ECUs have to be tamperproof, because if the public key can be exchanged for the key of an attacker the verification process is compromised. The same holds true if the private key for the signature generation is compromised/stolen. If possible, a certificate-based approach

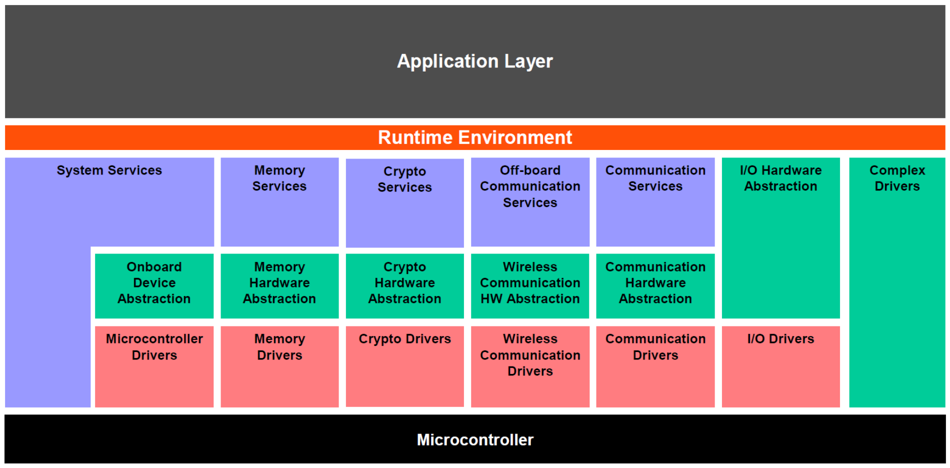

Figure 2: AUTOSAR Architecture [1]

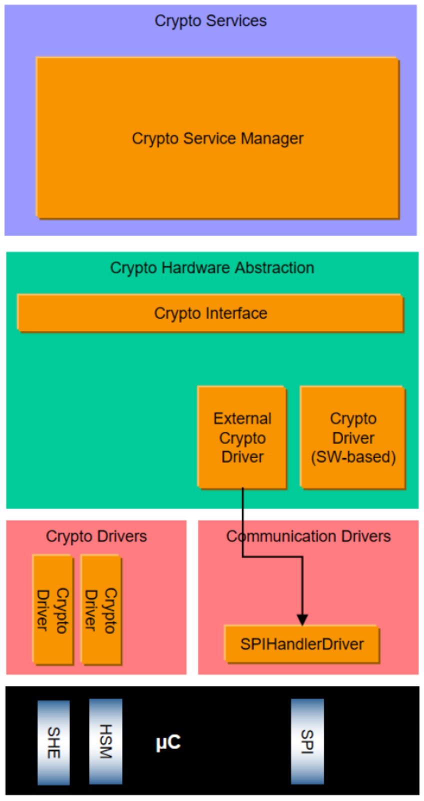

Figure 3: AUTOSAR Abstraction [2]

should be used, this allows the legitimate owner the exchange of certificates when the private key is compromised. A common standard for certificates is X.509 [6].

The processing of all attributes of an X.509v3 certificate generates lots of overheads. In addition, a PKI as well as a revocation list needs to be maintained over the product lifecycle incl. disposal.

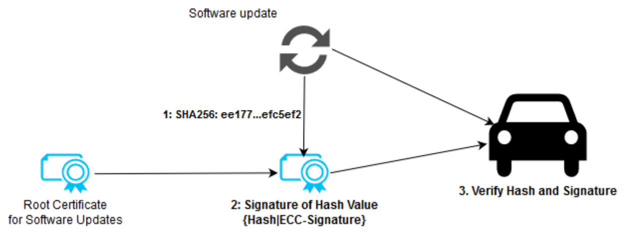
Software update
1: SHA256: ee177...efc5ef2
Root Certificate for Software Updates
2: Signature of Hash Value {Hash|ECC-Signature}
3. Verify Hash and Signature

Figure 4: Signing software updates

There is no common way to decide pro or con digital certificates. Every project has to evaluate the effort-benefit-ratio at the start of the project. Generally, the less security features a hardware has the more difficult it is to protect SW update verification keys/certificates and hence the lower the benefit of certificates will be in comparison to simple RSA/ECC signatures.

2.6 Diagnostic Service Security (UDS, XCP, . . .)

Diagnostic Services have to be secured when they have consequences for safety, security or the quality of exhaust gases, according to ISO 14229 [3]. The used security mechanisms in the field today however are not secure. The used mechanisms are based on symmetric algorithms – no cryptographic algorithms – with seed lengths for the SecurityAccess of 1-4 byte length. For upcoming developments, a length of not less than 8 byte is used.

The communication is up until now, after the initial authentication, only partially secured for updates and not for every instance. Among the many other issues in automotive components security-wise, this is one of the rather pressing ones.

2.7 Secure Boot

Secure boot is used to verify the integrity of all software during start up using digital signatures. It is necessary to guarantee that the different runtime stages written in software, such as the bootloader, the operating system or even possibly applications, were not altered by unauthorized persons. While the bootloader and

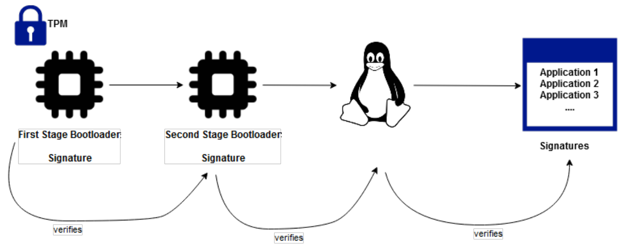

Figure 5: Secure Boot Schematic

operating system are usually under control of the manufacturer or Tier-1, custom applications can possibly come from official app stores or even third party vendors, which can make the publishing process more complicated. As common for most soft security terms the term *secure boot* is not standardized, however there is a common interpretation of that term on the market: the integrity of software is calculated in multiple stages. Every stage in the boot process is only started if the previous stage could verify its integrity successfully – otherwise the start-up halts or the ECU reboots.

To securely store the key for signature generation, a TPM or hardware trust anchor is required. It assures that the public keys or certificates which act as trust anchors are stored inside non-volatile memory and cannot be manipulated or deleted. Such a trust anchor can be publicly readable, as usually RSA or ECC public keys are used as well as their root certificates in terms of certificate based secure boot approaches. They do not need to be secret, or kept confidential. If such data is *one-time-programmable*, i. e., it can only be written once and never again (e. g., PROM). Care has to be taken as in consequence in case of security issues (e. g., stolen private keys) such a trusted anchor cannot be easily replaced or simply revoked. An example for a secure boot scheme is depicted in Figure 5.

If it is a requirement to have that trusted data to be updateable over the whole product lifecycle, a very secure access method is required to allow such a replacement or revocation. ECU producers should always inquire at silicon vendors if it is possible and allowed for their customers to replace / revoke data from trusted anchor locations. It is highly recommended not to trust in security marketing statements but to ask for technical details confirmed in writing.

This overall mechanism of secure boot is widely used, for example as an available feature in Windows 10 (UEFI [21]), or by default in the iOS operating system on Apples iPhones [4] to prevent jailbreaks and software modifications. Common time constraints are that safety critical features (rear-view camera picture) of the ECU have to be up and running in well below one second.

A subtype of secure boot is authenticated boot, which is a mechanism that can verify the integrity of the started software as well but the necessary checks are calculated in parallel, or even after start-up. This means that halting the startup-process is not possible, and the execution of untrusted software cannot be prevented. This is the trade-off for a faster boot time, but considered less secure. A manufacturer can use it to remotely assess the software running in the vehicle, but is only able to detect modifications without effectively mitigating them without additional means. For detecting attackers though, this is a feasible path.

2.8 Backend Communication (TLS & hardened TLS)

Some cars feature communication modules to interact with a backend system that allows users to remotely control car features and request information about their car. Current best-practices for encrypting and authenticating communication contents on the network layer are achieved using the TLS protocol suite [8]. While it is currently the protocol most commonly used world-wide with billions of connections every day, it can be tricky to achieve a strong level of security by configuring every option correctly.

For one, configurations have to be adaptable both for algorithmic primitives and for changes in current best-practices which can happen swiftly. Numerous publicly discussed instances document cases where the communication was insufficiently protected, such as for example in E-Mail protocols [11]. While usually the exchanged certificates are tested against validity, mechanisms can be employed that further harden the TLS connection. This could be certificate pinning, meaning that only a specific certification authority is allowed to authenticate endpoints, TLS tunneling for VPN-like connections as well as TLS-based VPN, and enforcing TLS connections. In the near future TLS 1.3 will be released, and while the previous versions were standardized allowing mechanisms to reduce the security of end-to-end encryption, the upcoming version is expected to be both resilient against attacks and using sane default primitives.

Entire books have been written about hardening of TLS, while the one by Ivan Ristic *Bulletproof SSL and TLS* is at the time of writing the most comprehensive resource [17]. Future use cases could include predictive maintenance, or real-time geolocation, and many more. Currently only a very limited set of cars with yearly subscriptions allow for this. This is expected to rise, due to the versatility of TLS and existing experiences from large number of users and websites that employ TLS.

2.9 Secure Car Internal Communication

The in-vehicular communication has, up until now with only a few minor exceptions, no security features at all. Currently only the CRC features of the CAN or other simple test values that offer no effective security are used, the focus was and mostly still is primarily on safety. The reason for this is that with the short payload of CAN messages and the restricted datarates it is not viable to add security as this would increase the overhead in proportions that are not feasible. Newer bus systems, e. g., CAN FD, will help mitigating this problem with increased payload sizes, but that means that all nodes have to support this new features.

The focus on safety will be shifting, as the SecOC AUTOSAR module demonstrates the intentions to protect the authenticity and integrity of data in-transit. Capabilities of automotive components however still struggle with the performance overhead and there is no best-practice available, although the presented HSM can help with an asynchronous computation on an additional logic core.

2.10 Firewall for On- & Off-Board Communication

Firewalls are used to filter and block data streams at network perimeters, internal network hubs based on hosts. The filters are based on a multitude of properties, starting with trivial filtering by point of origin or destination, ramping up to deep inspection of the payload. Firewalls for individual ECUs are commonly used to filter incoming messages by their CAN-IDs. For high-volume channels such as gateways, these firewalls as well as routing logic usually have performance implications. The same filter mechanisms are used for wireless connections with backend-server, so only certain services are received. The firewall features in both ECUs and VCUs usually have no logging mechanisms, but this is likely to change with the upcoming connection features where the entire car is online. An upcoming issue will be the rule updates, as in-place component upgrades with newer component generation could violate the existing rules or service oriented architectures with a dynamic allocation of services to ECUs. The same problem exists for cars with high configuration variability, as the configuration management and rule maintenance will become complex.

The first firewalls in cars which can report irregularities of internal car communication will soon become available on the market. Due to performance implications, this kind of observation will most likely be shifted to a high-performance cluster within the car. For more complex analysis aggregated data will be sent to an IT backend/cloud service in a secure manner. We also find it likely that an artificial intelligence approach will be used to have self-learning firewalls, but it has to be seen what the implications are on regular usage, how they are trained, and what happens if messages are falsely classified. For Ethernet connections, plenty of existing knowledge from the PC world can be used, but for automotive applications there is still plenty of learning to be done.

In general, a firewall as it is will not protect any system – the filtering rules as well as the possibility to update these rules very fast will make a firewall powerful. As long as artificial intelligence will be trained by data designed by humans, it will be likely the case that real intelligence of such a firewall is still to be derived.

2.11 Intrusion Detection / Prevention Systems

IDSs/IPSs are systems like firewalls but in contrast to firewalls are not located on network perimeters but inside the networks that shall be secured by monitoring/controlling the network traffic. Intrusion Detection Systems monitor the system/network and send logs to a backend system for further analysis. The detection mechanisms are primarily based on rules that have to be kept up to date and can be circumvented [20]. This makes continued updates and thus a backend connection necessary. Reaction times of IDSs for rule updates can be extensive, spanning up to multiple days or more in case of slow connectivity. An IPS further allows to directly influence the network or host by, e. g., prohibiting specific connections that match a known bad communication signature.

ID/IP systems can be distinguished in host and network based systems, where they safeguard individual ECUs, or the network domain. They can be realized as individual hardware component but also in software on an existing host. IDSs/IPSs are rather effective and until 2023+ expected by us to become a common security countermeasure to mitigate security risks. Nevertheless, it has to be mentioned that an IPS can have negative impacts on safety related functions. There is the posibility for the IPS to detect a false-positive, e. g., many brake signals are detected just before a crash which might be interpreted by an IPS as an *attack*. This aspect will become more critical in terms of future machine learning IPS algorithms. That means whenever an IPS will be used a backup path is necessary to supervise all IPS interpretations which will have performance impacts as well as liability aspects. An IDS in general is less critical because detection rules will be pre-defined by humans and only these rules will be applied. Of course there is still the possibility for human errors when defining such IDS rules but these can be minimized using a well-defined update process and rigorous testing.

2.12 Wi-Fi Security (Client & Back End)

For connecting client devices such as smartphones with the car, communication should be secured by known good measures, i. e., WPA2 for Wi-Fi that is spawned by the car, using strong passwords. These should be random and auto-generated if sufficient entropy is available, for example at the final assembly at the factory. For WPA2, the length of the password is of importance as there exist dictionary-based brute force attacks using, e. g., the tool hashcat [14]. As long as WPA3 is not yet fully specified and rolled out, additional policies can make sense such as regular password rotation or physical switches for connectivity features. As soon as WPA3 or higher will be available on the market, customers and OEMs should have a long-term update strategy to always support only the latest communication protocols.

Additional security measures could be placed on top of the network layer, such as an app that encrypts communication content on the application layer as well as using either a pinned TLS connection, or a shared secret which is only know to the car and the device after securely pairing them. This would yield three layers of encryption, which would make impersonation or replay attacks very difficult.

To allow cars to automatically connect to Wi-Fi networks with known SSIDs, further efforts are required to prevent connections with rogue access points. Examples could include specific workshop equipment workshop equipment or at gas stations. This is for example used in Tesla's Model S, which automatically connects to *Tesla Service* and *Tesla Guest* at their supercharger stations to provide connectivity and remote diagnostics [18]. Since the VCU will be constantly scanning for these known SSIDs in the background during runtime, it is very easy for an attacker to figure them out. Also MAC address filters are insufficient, as the MAC address can be modified easily.

For backend connections, every connection should be explicitly whitelisted, encrypted and authenticated, in particular if sensitive information like diagnostic codes or private data is transmitted. Additionally, every connection attempt should be logged and analyzed to detect brute-forcing attacks in a timely matter. It is beneficial to include these metrics into intrusion detection systems since wireless signals are much harder to confine locally.

3 CONCLUSION AND FUTURE WORK

With this paper we wanted to give an overview of the state-of-the-art in automotive security. The first part of the overview is a listing of security mechanisms available for the automotive industry and their market penetration as of today and as we see them for 2023 and onwards. In the second part of this paper we described the security mechanisms in detail and linked them to PC-IT systems.

The automotive development can benefit immensely from the accumulated knowledge and well-established standards for security in PC-IT systems. For future systems the methods and measures described in this paper have to be combined to achieve a layered defense according to a defense-in-depth approach. It is beneficial to have different security concepts working together, usually every security measure counters a dedicated threat, but many security measures not only prevent the sum of these threats, but also hinder the implementation of new and upcoming threats, as an attacker has to circumvent or deactivate them. One of the rising threats is the manipulation of devices on a hardware level with voltage and electromagnetic glitching [15]. Effective measures have to be developed and tested. First approaches for software based approaches were already presented [22]. Consequences can further be mitigated by device individual keys. Based on our research, the automotive sector has recognized the problems that arise from missing security and is on track to develop safe and secure systems, although not for all challenges fitting solutions for the automotive sector are available.

REFERENCES

[1] AUTOSAR. 2014. Layered Software Architecture. (2014). https://www.autosar.org/fileadmin/user_upload/standards/classic/4-3/AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf

[2] AUTOSAR. 2017. Specification of Crypto Interface. (2017). https://www.autosar.org/fileadmin/user_upload/standards/classic/4-3/AUTOSAR_SWS_CryptoInterface.pdf

[3] International Organization for Standardization (ISO). 2013. *ISO 14229: Road vehicles – Unified diagnostic services (UDS)*. Standard. International Organization for Standardization, Geneva, CH.

[4] Cedric Halbronn and Jean Sigwald. 2010. iPhone Security Model & Vulnerabilities. (2010). http://esec-lab.sogeti.com/static/publications/10-hitbkl-iphone.pdf

[5] Olaf Henniger, Alastair Ruddle, Hervé Seudié, Benjamin Weyl, Marko Wolf, and Thomas Wollinger. 2009. Securing vehicular on-board it systems: The evita project. In *VDI/VW Automotive Security Conference*. VDI-Verlag, Wolfsburg, D. https://www.evita-project.org/Publications/HRSW09.pdf

[6] Internet Engineering Task Force (IETF). 2008. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. (2008). https://tools.ietf.org/html/rfc5280

[7] Internet Engineering Task Force (IETF). 2014. Security Requirements of Time Protocols in Packet Switched Networks. (2014). https://tools.ietf.org/html/rfc7384

[8] Internet Engineering Task Force (IETF). 2018. The Transport Layer Security (TLS) Protocol Version 1.3. (2018). https://tools.ietf.org/html/draft-ietf-tls-tls13-28

[9] Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Stefan Mangard, Paul Kocher, Daniel Genkin, Yuval Yarom, and Mike Hamburg. 2018. Meltdown. *CoRR* abs/1801.01207 (2018). arXiv:1801.01207 http://arxiv.org/abs/1801.01207

[10] David Lodge. 2016. Hacking the Mitsubishi Outlander PHEV hybrid. (2016). https://www.pentestpartners.com/blog/hacking-the-mitsubishi-outlander-phev-hybrid-suv/

[11] Wilfried Mayer, Aaron Zauner, Martin Schmiedecker, and Markus Huber. 2016. No Need for Black Chambers: Testing TLS in the E-Mail Ecosystem at Large. In *11th International Conference on Availability, Reliability and Security (ARES)*. IEEE, IEEE, Salzburg, A, 10–20.

[12] Charlie Miller and Chris Valasek. 2015. Remote Exploitation of an Unaltered Passenger Vehicle. *Black Hat USA* 2015 (2015), 91. http://illmatics.com/RemoteCarHacking.pdf

[13] Charlie Miller and Chris Valasek. 2016. CAN Message Injection – OG Dynamite Edition. *Black Hat USA* 2016 (2016). http://illmatics.com/canmessageinjection.pdf

[14] Omar Nakhila, Afraa Attiah, Yier Jinz, and Cliff Zoux. 2015. Parallel Active Dictionary Attack on WPA2-PSK Wi-Fi Networks. In *Military Communications Conference, MILCOM 2015-2015 IEEE*. IEEE, IEEE, Tampa, FL, USA, 665–670.

[15] Ramiro Pareja, Santiago Cordoba Pellicer, and Niek Timmers. 2018. Fault Injection on Automotive Diagnostic Protocols. In *2018 escar USA*. escar USA.

[16] Martin Ring, Jürgen Dürrwang, Florian Sommer, and Reiner Kriesten. 2015. Survey on Vehicular Attacks – Building a Vulnerability Database. In *Vehicular Electronics and Safety (ICVES), 2015 IEEE International Conference on*. IEEE, 208–212.

[17] Ivan Ristic. 2013. *Bulletproof SSL and TLS: Understanding and Deploying SSL/TLS and PKI to Secure Servers and Web Applications*. Feisty Duck, London, UK.

[18] Marc Rogers and Kevin Mahaffey. 2015. How to Hack a Tesla Model S. (2015). https://www.youtube.com/watch?v=KX_0c9R4Fng

[19] Dieter Spaar. 2015. Sicherheitslücken bei BMWs ConnectedDrive. (2 2015). https://www.heise.de/ct/ausgabe/2015-5-Sicherheitsluecken-bei-BMWs-ConnectedDrive-2536384.html

[20] Kevin Timm. 2002. IDS Evasion Techniques and Tactics. *SecurityFocus (Infocus)* 7 (2002).

[21] Richard Wilkins and Brian Richardson. 2013. UEFI Secure Boot in Modern Computer Security Solutions. (2013). https://media.kasperskycontenthub.com/wp-content/uploads/sites/63/2014/06/21032725/UEFI_Secure_Boot_in_Modern_Computer_Security_Solutions_2013.pdf

[22] Marc Witteman and Martijn Oostdijk. 2008. Secure Application Programming in the Presence of Side Channel Attacks. In *RSA Conference*. Springer.

[23] Marko Wolf and Timo Gendrullis. 2012. Design, Implementation, and Evaluation of a Vehicular Hardware Security Module. In *Proceedings of the 14th International Conference on Information Security and Cryptology (ICISC'11)*. Springer-Verlag, Seoul, Korea, 302–318. https://doi.org/10.1007/978-3-642-31912-9_20