Feasibility Study for a Secure and Seamless Integration of Over the Air Software Update Capability in an Advanced Board Net Architecture

2016-01-0056
Published 04/05/2016

Bjoern Steurich
Infineon Technologies AG Munich

Klaus Scheibert
Infineon Technologies North America Corp

Axel Freiwald
Infineon Technologies AG Munich

Martin Klimke
Infineon Technologies AG

CITATION: Steurich, B., Scheibert, K., Freiwald, A., and Klimke, M., "Feasibility Study for a Secure and Seamless Integration of Over the Air Software Update Capability in an Advanced Board Net Architecture," SAE Technical Paper 2016-01-0056, 2016, doi:10.4271/2016-01-0056.

Abstract

Vehicle manufacturers are challenged by rising costs for vehicle recalls. A major part of the costs are caused by software updates. This paper describes a feasibility study on how to implement software update over the air (SOTA) in light vehicles. The differences and special challenges in the automotive environment in comparison to the cellular industry will be explained.

Three key requirements focus on the drivers' acceptance and thus are crucial for the vehicle manufacturers:

- SOTA must be protected against malicious attacks.
- SOTA shall interfere as little as possible with the availability of a vehicle. Long update processes with long vehicle downtimes or even complete fails must be avoided.
- The functional safety of the vehicle during operation may not be limited in any way

The study gives options how those objectives can be achieved. It considers the necessary security measures and describes the required adaptations of the board-net architectures both on software and hardware level.

Introduction

Updating software over the air is a commonly used practice for mobile devices in the consumer world. Main reason, why vehicle manufacturers still hesitate to do so, are various vehicle specific safety challenges, coming along with the update process.

When comparing update processes of vehicles to mobile phones for example, vehicles show significantly higher dependencies on all kinds of safety aspects or safety related side effects. Typical aspects hereby are especially risking lives, the potential of causing damage claims, or risking the brand's customer acceptance.

A critical update failure is that the update process does not execute as expected and the updated system refuses to restart. For a mobile phone the consequences are that the user needs to get the device to a repair shop and needs to grab another phone for making calls in the meantime. The potential maximum damage are the costs of a new phone of 20$ to 800$. Loss of data is typically avoided by backup tools and cloud services. For vehicles the situation is different. Vehicles might need to get towed from remote locations and might cause high costs for replacement and damages. The sum can get estimated to be roughly 100 times higher than for mobile phones. More critical is the exposure of peoples' life to vehicles. This exposure is well documented in all kinds of statistics and regulated by laws. As a result it is common sense in automotive industries that during and after the software update process it must be ensured under all circumstances that the vehicle does not get out of control. For consumer products like mobile phones the safety situation is much less critical. Only few incidents have been reported with mobile phones. For example there are a few cases where phones got on fire while being charged. Thus manufacturers have to handle significantly less risk when updating the software of a mobile phone. This risk also influences the users' acceptance of update processes.

The nature of software updates make systems unavailable at least for a short period of time. In contrast to the cellular industry vehicle manufacturers have little experience how well their customers accept this unavailability. As a consequence the requirements on update execution vary significantly between 0 and 30 minutes or more. Options of how to reduce the time are discussed in the second part of this paper.

Malicious attacks are another potential risk for vehicles' safety. As soon as the control networks of vehicles can get remotely accessed, they are also getting exposed to potential remote hacker attacks. Miller & Valasek [15] demonstrated very lively in 2015 how an entire vehicle platform can get manipulated via cellular connectivity (and this nationwide). Such a massive attack scenario is of course also conceivable for smart phones. Regardless of the resulting damage for the national economy, with vehicles the risk for the road users must be classified as being higher.

Such hopefully theoretical scenarios explain the necessity to carefully evaluate options to secure the entire SOTA process and even more stringed than in the case of mobile phones. A feasibility study of implementing protection mechanisms within vehicles by using existing devices is described in the first part of this paper.

Besides all those challenges, the prospect of continuous firmware improvements is compelling. The request goes far beyond the already existing update capabilities in infotainment systems [1]. IHS Automotive estimates that the total worldwide cost savings from software over the air (SOTA) updates will grow from $2.7 billion in 2015 to more than $35 billion in 2022.

Lower costs in case of recalls, faster response times, feature upgrades in the aftermarket and thus increased customer satisfaction are good reasons for the vehicle manufactures to introduce SOTA [1, 3].

Figure 1 gives an outlook on the expected implementation rate of OTA (over the air) software update capabilities in the automotive market. OTA updates for maps and the download of apps are state-of-the art, at least in the premium segment. A first quite prominent example for a SOTA fix of security vulnerabilities in a telematics control unit (TCU) occurred in 2015 [15].

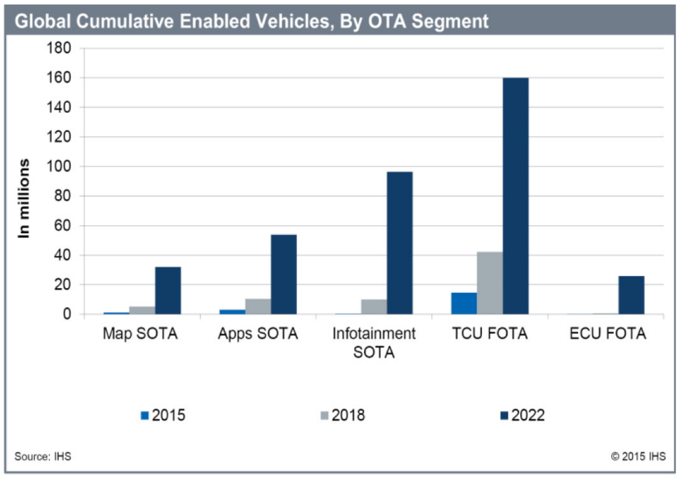


Figure 1. Global cumulative OTA enabled vehicles, by application segment

A technically challenging step will be implementation of SOTA capability on electronic control units (ECU) that are used outside the infotainment segment. Embedded Flash microcontroller (MCU) and no applications processor are typically used in those ECUs to control the real-time critical operations of the car. Also the required quality and functional safety level is normally much higher than for the application processors in the infotainment segment that are often based on consumer products.

The following study had the task to derive new requirements on the microcontroller architecture based on the expected board-net modifications. Of particular interest have been thereby potentially required modifications of the memory architecture and the necessary measures to comply with the security requirements of the SOTA process.

The first step was an investigation of the market environment prior to first round of requirement capturing at the OEMs (Original Equipment Manufacturer) to better understand their motivation. This step was followed by a detailed system analysis. As a result possible solution approaches have been deduced, including both, the necessary security measures as well as different options for the implementation on electronic control unit (ECU) level. The different implementation scenarios have been compared in terms of vehicle availability, cost, and impact on power consumption. Subsequently, further discussion rounds with OEMs followed, that will be also described in the paper. At the end is a summary and outlook

In the following chapter, the results of the market investigation are presented, followed by the results of the first discussion round with the OEMs in chapter two.

Market Environment

The interest to reduce costly recalls is one of the main motivation factors for the OEMs, whose importance is expected to increase further in future. National Highway Traffic Safety Administration reported for US a record high of 74.2 million recalls in 2014 and in May 2015 the number already reached 25 million [6,7]. Although only a portion of those recalls is solely related to software issues, it can be assumed with little risk of misinterpretation that this share will increase in future. According to Morgan Stanley [8], about 10% of the value of the vehicle relates today to its embedded software. This value is expected to shift gradually towards 40% in an autonomous vehicle environment (Figure 2). The 20% content includes advanced entertainment, improved productivity and other functionality to occupy the spare time of the passengers of the self-driving vehicle (topics that prerequisite another major software portion).

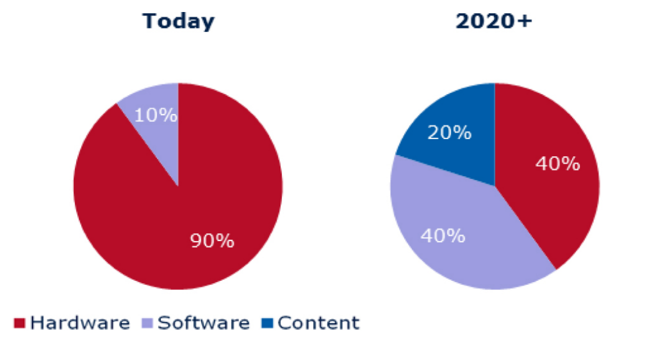
To assess value of the SOTA functionality for the OEMs the following model calculation:

Taking the US long-term recall statistic of the last 30 years as basis [9] and calculating with the vehicle sales numbers of North America in 2014 [10], results in a mean recall rate of ca. 84% (thus in average eight out of ten vehicles sold in the US will see a recall during operation lifetime)

- Assuming an average cost of 200€ for a firmware related recall (dealerships usually plan for at least two hours for ECU reprogramming; the necessary programming tools are relatively expensive and the number of available programming stations in a workshop are limited),

- An estimated cost adder of 75€ per vehicle (necessary investment in the board net architecture and secure update infrastructure),
- And by assuming that only 30% of the recalls can be fixed by SOTA,
- The potential cost savings should be in the range of 31.50€ per vehicle ((200€ - 75€)*0.3*0.84)).

This of course is only a rough estimate without any claim of completeness. The result would have been higher taking the recall rate US has seen in the first 5 months of 2014 into account (Figure 3).



The value shifts from predominantly hardware to hardware and software. A new revenue model will be introduced by being able to monetize the content.

Source: Morgan Stanley (2013)

Figure 2. Expected value shifts of automotive ECUs

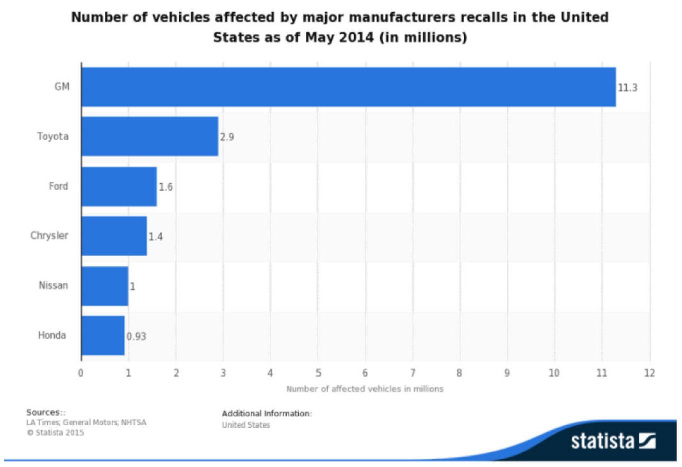


Figure 3. Number of vehicles affected by major manufacturers recalls in the United States as of May 2014

Even if 2014 was an inglorious exception, there are clear indications that the recall rate could be lastingly higher than in the decades before.

Results of the Initial Discussion Round with the OEMs

The discussion has been done in total with seven OEMs in US and Europe in order to get a wide spectrum of requirements from the industry. A first learning was the estimate for SOTA introduction on operational critical ECUs as shown in figure 1 is rather too conservative. First implementations are targeted for start of production (SOP) 2019. Also the desired number of SOTA enabled ECUs was surprisingly high. Some OEMs would like to enhance up to 50% of the ECUs till 2022 with this new software update functionality. Engine management besides applications in the body and comfort segment is one of the most desired candidates for the SOTA implementation. The desire to reduce costly software recalls was confirmed. Also the 200€ that have been used for the cost saving calculation are a consistent magnitude. The three key requirements already mentioned in the abstract (safety, availability and protection against malicious attacks) have been confirmed. The desire for availability thereby has two aspects. Availability on the one hand side it relates to the end-customer expectation. With the exception of an electrical vehicle (EV) that's not available on during the rather long lasting charging cycle a "conventional" vehicle is normally always available. The acceptable maximum time of a potential interruption of this availably is hard to judge. The second aspect that needs to be considered in conjunction with the availability or better the duration of the update process is the vehicle battery. The critically of the ECU applications excludes a change of the software version during vehicle operation. The update takes place while the vehicle is parked (with no engine is running). Thus the involved ECUs are getting solely supplied by the vehicle battery.

The last requirement that should be mentioned is the desire for a seamless integration in the existing vehicle architecture at minimum possible cost. As the current possibility for the software update by a garage tester and via the OBD interface will continue to exist, it's desirable to match this process as far as possible. Based on these findings, solution proposals have been developed.

Protection Against Malicious Attacks

The third chapter starts with the definition of the security system goals, followed by a high level overview on the update process and the regarded security architecture. Subsequently the update flow will be studied from the vehicle and software perspective. Here, in particular the security aspects will get regarded.

Security System Goals

The overall security architecture has to assure that a vehicle receives the right and integer firmware update.

To facilitate these goals, we assume the operation of a Public key infrastructure (PKI) which enables 3 functions:

1. Grant right to receive updates to the vehicle,
2. Grant right to update a vehicle to the OEM update service,
3. Assure the integrity of the firmware update.

Provisioning of rights is enabled by certificates and related private keys. Cryptography is based on standard algorithms (RSA, ECC, AES and SHA 256).

A principle block diagram of the assumed PKI is shown in figure 4.

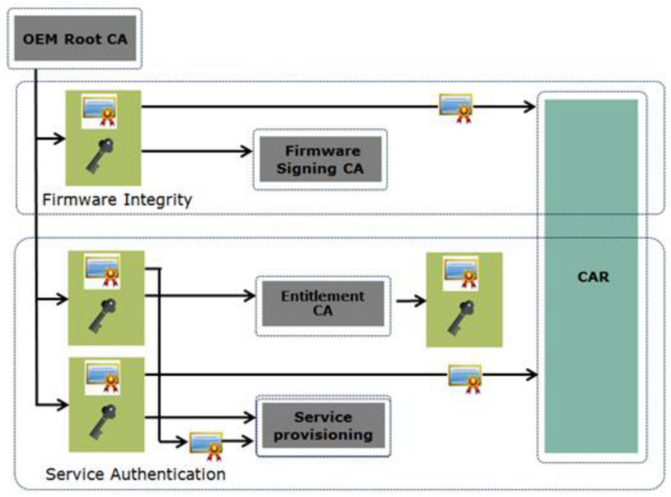


Figure 4. Sample PKI

The PKI can be separated into two parts, the provisioning of keys and certificates for the service authentication and the firmware verification. Not shown is the operation of an OEM key management system which facilitates the key provisioning to the vehicle.

High level Update Flow

The update is done in 4 basic steps:

1. The released firmware update is signed by the firmware signing certification authority (CA),
2. The vehicle, acting for security reasons as a client, is establishing a communication session with the OEM update service,
3. The vehicle and the server perform mutual authentication and establish a secure and transport encrypted channel by using TLS (Transport Layer Security),
4. The vehicle receives the firmware update, verifies it and performs the update.

In the following we will analyze these steps with respect of security focusing on the vehicle side. We will not cover recovery mechanism due to corrupted transmission or due to failed verification.

Vehicle Security Architecture

In the figure 5 building blocks to realize the over the air update are shown.

It can be separated into 3 parts:

1. The Telematics control unit. This unit is establishing the data link and is executing the service authentication. A microcontroller besides the application processor is commonly used to make the connection to the vehicle network and to implement emergency call functionality. As we will explain below, we recommend complementing the chip-set by a TPM (Trusted Platform Module) for security reasons.
2. The Central Gateway. This unit is receiving the firmware update, storing the update and is executing the first verification of the update. The ECU is acting as a central storage for all ECU firmware updates. This storage is likely to be implemented in an off chip flash memory. The OBD (On Board Diagnosis) interface remains connected to the central gateway, as it also serves today many non-SOTA related functions.
3. The Target ECU. This is the ECU that will receive the update and will execute the second verification of the update.

The control units are connected by different bus system (with date rates and communication protocols) that can have a significant impact on the update duration in dependence on the implementation model as we will describe later in this paper.

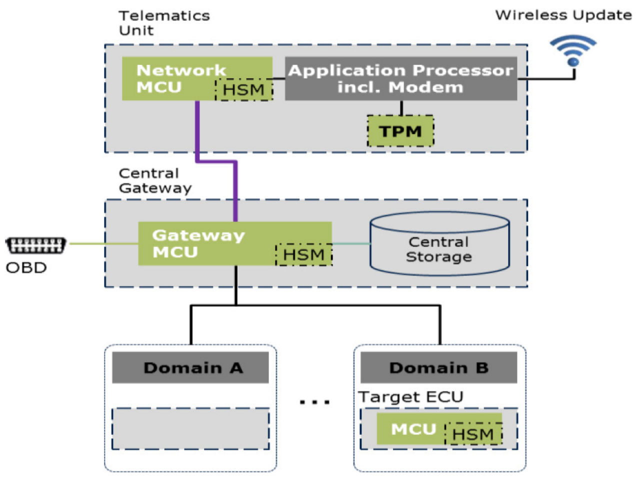


Figure 5. Draft Functional Model

A different system partitioning as shown in figure 5 is of course possible. The reason why we picked the gateway as the central entity to store and verify the update was:

- It has access to most busses of the vehicle's network and thereby may be able to transmit update simultaneously, thus speed up the update process,
- It is already today a security critical ECU because it performs other security critical tasks like bus separation and firewalling.

The communication between the gateway and the target ECU can be secured by message authentication code (MAC). It is assumed that the MAC is based on symmetric keys using e.g. CMAC and AES. Provisioning and management of those keys is outside the scope of this article. It is further recommended that all ECUs implement measures to enable their integrity.

The considerations on the proposed security flow can be broken down in three main steps (figure 6) and is explained in the sub-chapters "Service Authentication", "Firmware verification" and "Secure distribution of the update within the vehicle".

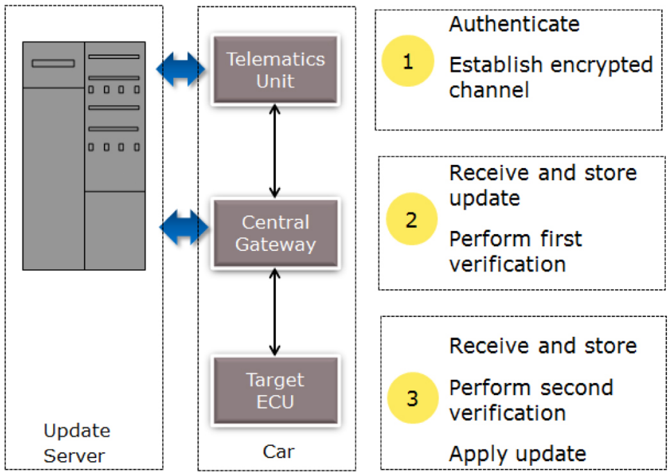


Figure 6. Overview Firmware Update

Service Authentication

The update of safety critical ECUs needs a high level of security since it may impact functions that can affect life and limb of the passengers. It is also a process that will be applied over the air not requiring physical contact to the vehicle, thereby in case of security breaches may affect many vehicles in a short amount of time. The service authentication is an essential element of the security of this process.

Thus we propose the use of a TPM as the entity to host the storage and processing of certificates and private keys. The TPM is produced and personalized in security certified manufacturing environment. It further improves the protection of keys in many aspects and also has a positive impact on the total cost of security which will be further discussed in more detail.

It is assumed that the participating MCUs have either a Secure Hardware Extension (SHE) module [11] or an EVITA [12] compliant Hardware Security Module (HSM). This will be later also justified in more detail.

Firmware Verification

After successful authentication, the server is able to identify the vehicle and further receive additional information to provide the right update.

The firmware update can be separated into two phases.

1. The reception and the intermediate storage of the firmware which can be done while the vehicle is in use (as we already explained before). In our architecture this function is performed in the gateway.
2. The update of the target ECU

Mitigation procedures due to verification errors and potential transmission errors should not impact the time the vehicle cannot be used. We, therefore, suggest that a first firmware verification should be done already on the gateway in phase one. Since the gateway is verifying the integrity of the firmware, no further measures have to be implemented to assure authentic communication between telematics unit and gateway.

Firmware verification is done on the gateway MCU using the HSM. Since firmware verification is only using public certificates and no secret keys, security requirements are lower than in the authentication process. Additionally the performance of the verification will benefit from high performance crypto accelerators and fast communication busses of the HSM.

For the time being we assume that the firmware update is signed and verified as a whole but it is also possible to separate the firmware in smaller blocks which are each signed individually. This would also offer the possibility to repeat the authentication process and by this create a new session key. Thereby, the time a session key is used is shortened. This is a barrier against side channel attacks requiring the repetitive use of the same key over a longer period of time.

Only after this phase is successfully finished, phase two (the update of the target ECU) should be initiated.

Secure Distribution of the Service Pack within the Vehicle

A service pack is the collection of all data needed to execute a complete firmware update of a single ECU. The service pack is transmitted over the air and stored in the central storage. Depending on the number of ECUs which participate in the update one service update might consist of several individual service packs.

Via secure and authentic communication, the gateway signals to the target ECU, that the individual service pack can be transferred and the transfer starts after confirmation of the target ECU.

Transfer of the individual service pack via the internal busses requires no further authentication because the software (SW) update package itself is signed by the issuing OEM and the integrity of the update is again verified by the target ECU.

As already discussed above the target ECU verifies the individual service pack for a second time by using the HSM. The secure FLASH Boot Loader is responsible for receiving and verifying the update. The Secure FLASH Boot Loader will be explained next.

Secure FLASH Boot Loader

Within each ECU the Secure FLASH Boot Loader (SFBL) handles all required firmware update functions in a secure way. The difference between a secure and a non-secure FLASH Boot Loader is mainly the usage of additional crypto algorithms. Thus also ECUs without HSM can participate in a SOTA process resulting in reduced security.

As the complete re-flashing process has to be handled out of the limited SRAM sizes of the microcontroller, service packs consist of smaller blocks. Each of these blocks can be transferred and handled by the SFBL individually. This may include the decryption with a vehicle series specific or ECU specific symmetrical AES128 key, which is used for the IP protection of the service pack SW during the transport from the OEM server to the target ECU. Furthermore, OEM individual certificates clinched to the small service pack blocks can be used to guarantee individual block trust level. When a complete ECU service pack is stored in the target program FLASH, the process integrity can be checked by a cryptographic CRC (cyclic redundancy check), such as e.g. applying a SHA256.

Aside the security, the SFBL has to handle the entire programming of the Flash which depends on the memory architecture of the used microcontroller.

Most embedded FLASH architectures today have some known limitations on the one hand side for the amount of permissible overall erase/program cycles and on the other hand in the segment erase granularity which is gross grain in the range of 16KB to 512KB per logical FLASH sector. The necessary amount of reprogramming cycles of the program FLASH must be of course also reviewed in the context of SOTA. In case of environed disruptions (e.g. loss of power) of the SOTA process critical situations can arise. A 'Brain Dead' situation is classified by an ECU being no longer able to reboot correctly after restart. Most critical are situations when an ECU is unable to communicate to the vehicle's network. In case of SOTA this would lead to a nonoperational vehicle in a remote location. The most reliable prevention measure to avoid such undesired situations is the usage of several instances of TIER1 and OEM persistent ECU Boot Loaders. Boot Loaders are typically not changed in the field and shall be excluded from the standard SOTA update procedure. As a general recommendation all updates of critical software which might affect basic operation, security related tasks (HSM) and availability critical routines should not be part of a SOTA operation. Critical software shall be only updated in the protected environment of a service station.

The section below is devoted to consideration on IP (intellectual property) protection, before coming to the resulting component requirements at the end of this chapter.

IP Protection

The request of the OEMs for IP protection is based on two main motivation factors. On the one hand side it's let by the desire to preserve competitiveness, on the other side to by the desire to open up new sales opportunities in the aftermarket (thus using SOTA to sell functional upgrades). IP Protection can be achieved by encrypting the firmware update while it is being sent via external and internal busses and must also be applied when storing the update in off chip memory.

Since the transfer from the OEM update server to the vehicle is encrypted by a symmetric session key, there is already a measure in place to protect this channel. Further security hardening can be achieved by using ECU specific keys.

On the vehicle, a good approach could be to exchange an encryption key between the receiving ECU and the Gateway and use this key to encrypt the firmware. This would both protect the transmission from the Gateway to the ECU and the storage in the off chip memory.

Now we come to the considerations on the hardware components. The focus is given first step to the security requirements. A partitioning proposal is done for the critical crypto functions on two essential hardware trust anchors (including a detailed justification for the recommended components). The discourse is supplemented by considerations about the total cost of security ownership.

Functional Partitioning Between TPM and Application Processor

The TPM is able to execute the related cryptographic algorithms which are the basis for the authentication. It therefore stores long lasting certificates and related private keys in a strongly protected security certified execution environment. TPM 2.0 is supporting the most recent standard algorithms like ECC, RSA, AES and SHA 256 and provides flexibility in the choice of algorithms.

The TPM can be cryptographically bound to the application processor. By this the information flow between the TPM and application processor shall be encrypted, thus protected against 'man in the middle' attacks. For transport encryption, TLS is proposed. In TLS after successful authentication, a session key is negotiated between the two parties and data then is encrypted using symmetric encryption. The information to compute the session key is transferred from the TPM to the application processor and the bulk encryption is then finally executed on the application processor. It is foreseeable that the authentication capabilities of the telematics unit will be used in future for other services which will be granted, revoked and authenticated by different certificates and private keys. A driver for separating services may also be privacy protection and need to operate related services separately on the OEM backend. The related key store needs to support this scenario. TPM key storage is highly scalable and can be securely off loaded to external memory of the application processor. This enables the OEM to store further certificates for authentication of other services. Additionally the TPM provides a secure and standardized key management interface that enables the provisioning and revocation of keys end- to- end secured.

Thereby, the increased security of the TPM can be used for all authentication use cases of the OEM and also the OEM can benefit from a high secure authentication stack for all of his services.

In the following the protection measures for the ECUs will be regarded. These considerations also include the central gateway.

Security Hardening of ECUs

In general security attacks are performed by operating a system in a non-specified mode. A certain function is manipulated to perform a task which it is not supposed to do.

A general concept to mitigate these attacks is to separate the system in various security domains which are isolated from each other. The security efforts and measures to harden these security domains then depend on the criticality of the attack scenario.

Example of security domains are:

1. Hardware security domains which isolate the cryptographic processing and key storage from the rest of system
2. Different privilege level in operating systems
3. The isolation between domains within the vehicle by a gateway incorporating firewalling

To create security domains soft- and hardware support is required.

With Infineon's product portfolio a differentiated approach with respect of security hardening of various domains can be implemented. An example is the aforementioned TPM hosting the asymmetric key storage for long lasting, security credentials in tamper resistant component.

Also within the MCU various hardware measures are in place to isolated security domains. In the AURIX^{TM} microcontrollers, the dedicated HSM domain can isolate security functions from the application domain.

Infineon saw this value early and therefore pushed the availability of HSM also covering the lower ends of the product portfolio. This already enables customers today to benefit from of HSM in applications which require a lower application performance. A typical example would be the airbag ECU requiring authentic communication support by HSM. Additionally customers can leverage their investment into security technology over a broad range of applications.

Currently the main functions of the HSM are to act as a trusted execution environment for storing and processing keys. In the context of SOTA, HSM can be of additional value by performing an on demand integrity check. In our example, both the telematics unit and the gateway exchange their integrity status securely and only then start the update process. Same can be applied to the target ECU before starting phase two of the software update. In addition, the target ECU itself may increase the security level by the implemented 'secure boot' feature from HSM. Before allowing the application software to execute its code after reboot, the HSM computes a secure seed (e.g. AES128 MAC or SHA256) and by this double-checks the trust level of the application software itself. Even on-the-fly code checks of the application software (e.g. for the secure Flash Boot Loader) can be easily applied in case of SOTA. This capability is one of the significant improvements of the HSM compared to the before mentioned SHE module. With the upcoming next generation of AURIX^{TM} family Infineon will further support this concept and the deployment of HSM will cover 100% of all new derivatives.

Since hardware security initially incurs additional charges the following considerations on the total cost of security ownership.

Total Cost of Security Ownership Considerations

Hardware trust anchors are additional components which increase the bill of material (BOM). But cost for security is not only related to component cost but to overall cost installing and maintaining security processes. Here hardware trust anchors can help to reduce those costs and simultaneously increase security. Hardware trust anchors provide security measures against physical attacks (e.g. side channel attacks). Keys injected in hardware trust anchors are thereby strongly protected not only during the operation of the vehicle but also during various shipment processes of vehicle components and also the shipment of the hardware trust anchors as such.

The TPM is personalized, meaning it already contains a so called endorsement certificate and related private key that provides the following benefits:

- It can be used to establish a secure, encrypted channel to the OEM personalization system in manufacturing to inject further keys into the ECU and further stored end- to- end secured into the TPM.
- This relieves the OEM from the costs having to establish and maintain security measures in its manufacturing to handle keys unencrypted.
- Additionally the endorsement certificate is signed by Infineon; thereby the OEM can inexpensively verify the genuineness of this security critical component by simply verifying the endorsement certificate.
- Additionally hardware trust anchors are produced, developed and personalized in security certified environments which are regularly audited by 3rd parties under governmental control. This relieves the OEM from performing those audits thus lowering the total cost of ownership.

The following fourth chapter of the paper is about finding methods to integrate SOTA with minimum possible interference to the existing system architecture and assuring maximum availability of the car. The focus is on considerations on the available board-net architecture and the ECU specific requirements.

Proposals Concerning Maximum Availability and Minimal Intrusion into the Existing Board Net Architecture

The availability of the car is particular critical when an emergency situations triggers the urgent desire to crank the engine and to operate the vehicle. As we will explain in more detail in the following, the final sequence of the update process on ECU level will always result in a certain non-availability of the ECUs for normal operation. There are of course technical means to reduce this non-availability down to few milliseconds. A delay that could be most probably added with some adaptations to the normal power-down or -up cycle of most vehicles, but the necessary add-on cost might not be easily justifiable, especially when targeting the lower-end market. As alternative, it has been proposed by the OEMs to extend the power-down cycle (after engine off) by the last sequence of the SOTA process. Since the engine is already turned off at this moment, the participating ECUs and (at least part of) the board-net have to be supplied solely by the vehicle battery. By doing so it's however critical to conserve the minimum battery capacity needed for next re-cranking cycle of the engine and that under worst case conditions (cold start after an extended parking period). As today's software updates are normally done at the service station where the board-net can be additionally supplied from supported, extra measures to delimit the energy consumption (as e.g. sub net operation) are not common in the automotive industry. For this reason, now follows an estimate of the maximum tolerable duration under consideration of the usual battery capacities.

Maximum Possible ECU Update Duration by Sole Supply of the Vehicle Battery

Even with no active electrical loads and at engine off, a realistic assumption for the entire board-net consumption is in the range of 8A for small vehicles ($\leq$ 40ECUs) and up to 25A for premium vehicles ($\geq$ 100 ECUs). The MBN LV 124-1 norm [13] requires 6 cranking cycles of 10s each under cold start conditions. 400A can be assumed for the cranking current, resulting in an overall battery capacity requirement of ca. 7Ah to comply to the MBN LV 124-1 test cycle.

The battery capacity of a light vehicle usually ranges between 36 and 110Ah. The state of charge (SoC) of the vehicle battery normally ranges between 70% and 100%. It is important however, to also to consider corner cases like weak batteries and cranking at cold weather conditions. This allows us deriving the following recommendations for the last sequence of SOTA process:

- No execution at low ambient temperature,
- No execution under 70% state of charge (SoC),
- And an upper limit of the maximal usable battery capacity for SOTA of max. 20%

Battery sensors that are already available for the most start-stop systems can be used to determine the 'State of Charge' (SoC) and 'State of Life' (SoL) of the battery.

By allowing 20% of the battery capacity for SOTA and based on the typical values of the board-net consumption as mentioned before, we can derive as maximum tolerable update duration 45 minutes.

The board-net consumption could be of course reduced by additional measures like partial or pretended network operation [14]. But again, those measures are not widely implemented today due to cost

The next chapter examines options to comply with the OEMs' desire of a seamless integration in the existing board-net architecture.

Seamless Integration in the Existing Board-Net Architecture

It's about adapting to the already existing service station software update process and optimum usage of existing system resources. Although surcharges can't be completely avoided, they should be limited to a minimum.

In today's vehicles, the re-programming of single ECUs (or the update of the entire car) is done at a service station. Such updates are typically performed by a diagnosis tool which gets attached to the OBD (On Board Diagnosis) connector. The diagnosis tool handles all functions to control the update flow e.g. the download of the service pack, the distribution to the target ECUs and the final verification. For the supplementary SOTA functionality, the OEMs understandably desire to keep as far as possible to the existing update mechanisms (incl. Secure FLASH Boot Loader etc.). For SOTA it is therefore the need to transfer the functionality of the diagnosis tool to a central location in the board-net architecture and to add functions that handle the additional OTA flow (incl. the additional security functionality we described before). It is assumed that these functions are executed inside the gateway ECU (other solutions are possible, but are not considered in this paper).

Those additional SOTA functions include vehicle network protocol extensions, which may initiate and control in the target ECU the required update services. A small list of required functions can be found below:

- Report current SW version
- Change vehicle status to update state
- Controlled transfer of SW blocks e.g. handshake based
- Verify update
- Reboot network
- A/B SWAP (if applicable)
- Transfer and caching in local ECU storage (if applicable)
- …
  (A/B SWAP and local caching will be also explained in the following)

Some of these functions are already standardized within UDS (Universal Diagnosis System) and ISO 14229-1:2013.

While doing software update at the service station the vehicle is typically in a predicted environment and out of operation for a longer time. For SOTA the situation is different. The location of the vehicle can be anywhere, the environment can vary and even the time while the vehicle cannot be used is directly influencing the owner's acceptance the update process.

Since the data needs to be transferred from the central storage to the target ECUs, the network topology needs to be examined. This differs sometimes significantly from OEM to OEM. Transportation speeds of different vehicle network protocols and specific bus implementations have been recognized as the main bottleneck.

In the following approaches are presented that consider the impact of the network topology. The characteristics of different solutions can by roughly distinguished by

1. the distribution of service packs from central data storage to the specific ECUs,
2. how many ECUs are able to change software in parallel,
3. and as a result how long the vehicle is out of operation.

Subsequent three different approaches are discussed without interfering with specific solutions of the different OEMs:

1. Classic approach: Update of individual ECUs by loading and reprogramming individual service packs over the vehicle network to the embedded FLASH in one step.
2. A/B Swap: The main idea of the approach is, to have two blocks, A and B, of FLASH memory for code execution within the microcontroller. The SW download from the central storage to the target ECU and reprogramming of the "spare" memory can get executed in the background and as slow as necessary to ensure full operation of the vehicle.
3. Additional local storage memory on ECU level: service packs are distributed to this local memory in the background (during vehicle operation). The participating ECUs are than updating their software out of the local storage either at power-down or -up (thus out of the safety critical operation)

First the classical approach.

Classic Approach

In this paper the solution is called classic, because all existing ECUs which can get a software update via OBD can also get used with this approach (provided the already described extended functionality of the gateway). It does not require any changes of hardware within the ECUs. The drawback is, that especially with slower busses like CAN, the bus speed is mainly defining the duration of the update process.

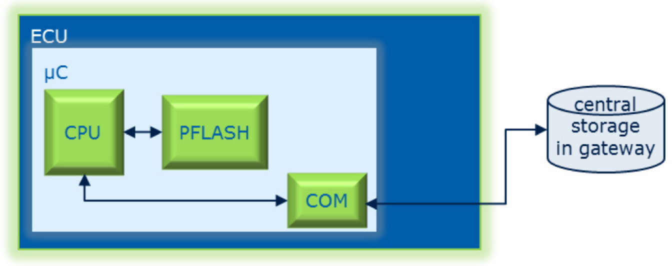


Figure 7. Classic approach, transfer data from central storage during update

The table below shows examples for a 4MByte-update using different bus standards under realistic operation conditions. The realistic raw data rate (payload) of the respective communication protocol is determinate by the protocol overhead, the nominal data rate and application specifics.

Table 1. Examples for transferring 4MBytes via various bus types

Protocol nominal data rate	Realistic Transfer Rate	Time for transfer of 4MBytes
CAN 500kbits/s	16.8kbytes/s 50% bus load	250s
CAN-FD 2Mbits/s	88.5kbytes/s 50% bus load	47.4s
FlexRay 10Mbits/s	300kbytes/s 50% dynamic segment usage	13.6s
Ethernet 100baseT 100Mbits/s	5 to 10 Mbytes/s (UCP or TCP/IP)	0.8s to 1.6s

In addition to the time to transfer the data there is the time needed for data and communication handling, decryption, erasing, programming and verification of the embedded FLASH of the used microcontrollers. The issue of the classic approach is obvious: when using the most common implementation of CAN with 500kbits/s and updating one ECU it takes around 5 minutes. In case of a service update of 20 MCUs (with 4 MByte each), the vehicle could not be used for 1h40min. This is beyond the 45 min we estimated before.

Nevertheless, the approach would allow to transfer update 36 MByte of software. Or at least 18 MByte in case the involved ECUs have to "fall-back" to the previous software versions due to a fail of the final software verification.

This time window can be further optimized by additional measures on network level. As an example, if the ECUs are clustered to different CAN bus sub-domains (usual practice today according functional groups and application segments) with a gateway in center (incl. central SOTA storage), it is possible to transfer service packages to those sub-domains in parallel. Another meaningful approach to decrease the data foot print (for the cellular and the following bus transfer) is to use data compression.

The classical approach is regarded as beneficial as the car owner spares the time bringing the vehicle to the garage for any pure software update and as it allows to implement a very cost sensitive solution especially in the lower car segments. On the other side there might be enough peer pressure for the OEMs to look for quicker solutions especially to attract their customers in the upper car segments. And this even if it results in some additional effort and cost for the resulting solution.

A/B Swap

A more elegant, but costly solution is the so called A/B SWAP method (Figure 8). The main idea of A/B SWAP is, to have two blocks of FLASH memory for code execution within each microcontroller of the single ECUs [5]. Block A is used for executing the actual code. Block B is a spare FLASH block. The new software is programmed into block B in the background (when driving). After all ECUs have finished the "pre-storing" process, a command of the central SOTA controller (the gateway in our case) swaps code execution from block A to block B. The SWAP is completed with a final restart of the ECUs. The significant advantage of this approach is the fact that there is almost zero downtime of the vehicle. A restart takes less than a few milliseconds and can be added to the power-up or -down cycle as mentioned before. In case something went wrong within the vehicles network there is a fallback solution available within block A. If the new software image of any ECU turns out to be corrupted there is always the option to switch back to block A of all involved ECUs of the actual service update within a few milliseconds.

The drawback of the solution is that the size of embedded program FLASH is doubling and that the mechanisms of swapping are not available for most microcontrollers today. In this context it's worth mentioning that most automotive applications (with a few exceptions in the infotainment domain) are operated out of embedded Flash microcontrollers. This is mainly due to the high demands on system availability and functional safety of the vehicle (one can assume that there will be still real-time capable "satellite" ECUs required in combination with powerful sensor fusion processors for the autonomous car).

The demand for a seamless memory swap mechanism is a particular challenge for any microcontroller supplier. Apart of the necessary availability of components with the required memory sizes, the potential impact on the entire system architecture has to be carefully considered in order to come to a robust implementation of the swap mechanism. The potential impact of the overlay mechanism on the functional safety and security must be analyzed in very detail and this for each particular application.

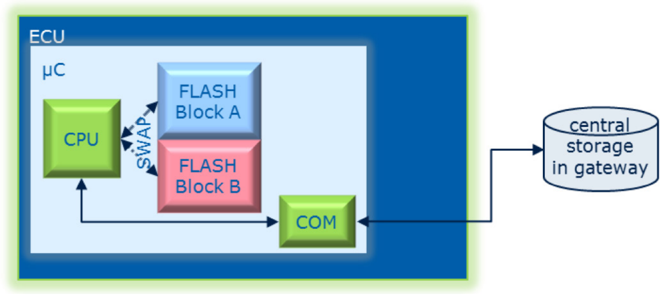


Figure 8. A/B SWAP

Doubling the FLASH of a microcontroller from e.g. 4MB to 8MB obviously increases the cost. This adder can be quite significant as microcontrollers - in contrast to application processors or stand-alone

Flash memories - are based on an embedded technology (thus a combination of a memory and CMOS technology with corresponding yield implications). The challenge of the A/B Swap approach is even graver for components at the upper memory limit of the actual technology note. Today the biggest memory sizes offered are in the range of 8 to 16MB. The situation is aggravated by the fact that those ECUs in the car with the biggest embedded FLASH MCUs are the once that are the most prone for potential software bugs or the most desired candidates for in the-field feature enhancements (a typical example is the engine control unit).

Therefore, below a third approach, which tries to combine the advantages of the first two methods?

Local Storage

The third approach is based on the fact that modern microcontrollers can erase and reprogram the program FLASH much faster than it was the case in the past. For example a 4 MByte FLASH can get erased and reprogrammed within 8s. Such speed improvements are key for the following approach that also allows avoiding long vehicle down as been described for the first approach.

The proposal is to add a local memory buffer to each target ECUs in form of an additional serial FLASH. The resulting overall update time for the selected 4MByte reference stays well below 9s despite the comparable slow serial connection (incl. the necessary communication overhead).

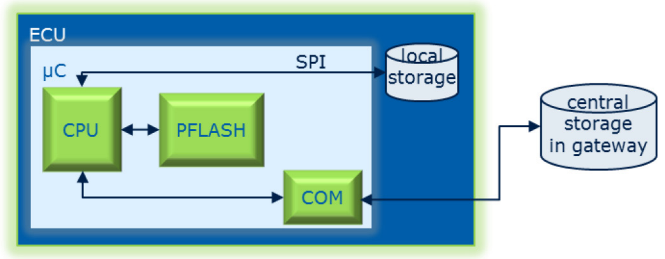


Figure 9. Updating the embedded FLASH from a local buffer

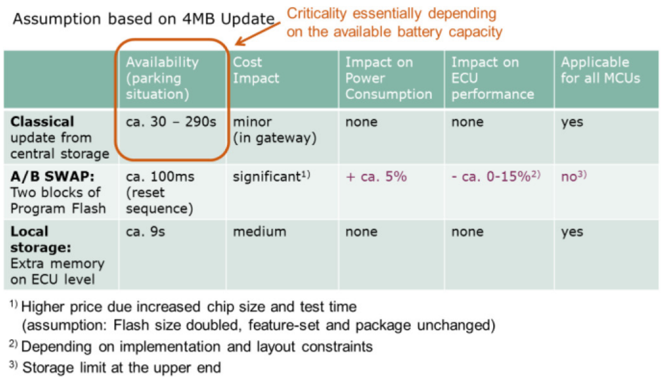
The decisive advantages of the local memory solution in comparison to the A/B SWAB approach can be summarized as follows:

- Minimal interference to the existing system design and therefor manageable risk (incl. system safety & security, development effort and time for re-qualification),
- And this in combination with relative small extra cost (significantly lower production cost compared to embedded Flash and small form factor of the extra component)

The conclusion of the chapter is done by a direct comparison of the three approaches.

Comparison of the Three Approaches

Figure 10 gives an overview on the advantages and disadvantages of the different solutions.



Assumption based on 4MB Update

Criticality essentially depending on the available battery capacity

	Availability (parking situation)	Cost Impact	Impact on Power Consumption	Impact on ECU performance	Applicable for all MCUs
Classical update from central storage	ca. 30 – 290s	minor (in gateway)	none	none	yes
A/B SWAP: Two blocks of Program Flash	ca. 100ms (reset sequence)	significant[1)]	+ ca. 5%	- ca. 0-15%[2)]	no[3)]
Local storage: Extra memory on ECU level	ca. 9s	medium	none	none	yes

[1)] Higher price due increased chip size and test time (assumption: Flash size doubled, feature-set and package unchanged)
[2)] Depending on implementation and layout constraints
[3)] Storage limit at the upper end

Figure 10. Overview on the advantages and disadvantages of the different solutions.

As the benefits of the third method outweigh the penalty of the slightly increased non-availability form the author's perspective, a summary of the key advantages of the local storage in comparison to the first two methods is given:

- Download in back ground: Like with A/B SWAP downloading of service packs can be done in the background during run time without interfering with normal operation.
- Parallel update: Once the distribution is complete all ECUs can execute the update task in parallel. This is a big advantage against the classical approach.
- Low extra cost: The advantage is that serial FLASH devices are available for less cost and with bigger sizes in automotive quality than integrated FLASH (and this at comparatively small from factors). Even the biggest sizes of code can get handled with this approach. Second it might be considered that the size of the central storage can get reduced. Inside of the local memory all necessary integrity checks can get executed in the same way as in the central storage, too.
- Low risk and development effort: Much smaller modifications to an existing system design in comparison to A/B SWAP approach
- Devices available from multiple suppliers: In contrast to the A/B SWAP the third solution can get implemented with existing standard components (e.g. serial NOR-FLASH types) which are already available in a wide variation range today.

It should be here pointed out again that all considerations have been based on the specific requirements of microcontrollers that rely on embedded FLASH. The A/B Swap might be beneficial for any processor operating from external memories. Nevertheless, the safety and security requirements must not be disregarded.

Subsequently, the results of the second discussion round of with the OEMs.

OEM Feedback

All OEMs we met agreed that A/B SWAP approach is the most elegant solution; nevertheless due to the high cost and the missing end-customer experience the majority of OEMs consider starting with the classical approach (whereby the proposed local storage is regarded as a good compromise). The discussions are further on-going. The final implementation will depend individually on each OEM's network architecture that is evolving in parallel and closely related to introduction of new system bus as Ethernet. A large unknown is further the end-customer acceptance of a potential non-availability of the vehicle.

Summary/Conclusions

Evolutionary steps and reusing existing methods and hardware as far as possible is the best way of seamlessly introducing SOTA updates.

The objectives of the study have been met. A list of requirements to the hardware was created. The requirements for short-term and seamless introduction of the SOTA functionality can be met with the existing TPM and AURIX™ components. Three different methodologies have been discussed:

- Introducing SOTA update with the classic approach is possible as soon as a telematics unit, a central storage, and update control functions (e.g. SOTA car network protocol) are implemented in the vehicle. The downtime of a vehicle is mainly defined by the number of ECUs to be updated. A typical single ECU update is less than 4MBytes and thus the downtime can be less than 5 minutes. Already by this approach a lot of recalls for software updates can get avoided by the OEMs. A few minutes is likely acceptable by the owner as the downtime of his vehicle is definitely shorter than bringing it to the service station. In addition, the vehicle owner could get the option to select the time of the update according to make it most convenient. The limitations are coming when updating multiple ECUs, not just as explained above with longer down times but also in respect to battery issues explained later in this paper.
- A/B SWAP is not the best choice of approach. Exactly those ECUs with the biggest Flash memories do not have appropriate products in the market for technology reasons. The coexistence of ECUs with SWAP and with classical approach makes the flow complicated and possibly unsafe. In addition the SWAP memory cannot be used well as data storage for others. It contains either the new version in case the ECU is updated itself or the fall back software of the previous operation.
- Adding a small serial Flash memory has most advantages not only for the biggest microcontrollers. An external extended Flash via SPI can get added to most designs without changing the ECUs architecture completely. The size of the Flash can get adapted to the needs within the ECU and within the domain without significantly changing the footprint on the PCB (a wide range of serial NOR FLASHs is available in automotive quality and temperature and in tiny small package variants e.g. SOP-8). Thus within a vehicle step by step ECUs can get added seamlessly such hardware capabilities without revolutionary brakes in the network concept.

Finally we can conclude that a cost sensitive introduction of SOTA should be possible for ECUs by either the first or the third methodology. The benefits of the A/B SWAP method can't be of course neglected. Nevertheless, smarter overlay mechanism will be required to avoid a complete re-assessment of the functional safety concept.

References

1. IHS Inc., "Over-the-air Software Updates to Create Boom for Automotive Market", http://press.ihs.com, Sep. 2015.
2. ADAC e.V. "ADAC deckt IT-Sicherheitslücke bei BMW auf: Autos elektronisch geknackt", https://presse.adac.de, Jan. 2015.
3. Computerworld, Inc,, "Over-the-air software coming soon to your next car", http://www.computerworld.com, Feb. 2015.
4. WEKA FACHMEDIEN GmbH., "Warum die Autoindustrie neue Software Updates braucht", http://www.electronicnet.de, Mar. 2015
5. Lobdell, M., "Robust Over-the-Air Firmware Updates Using Program FLASH Memory Swap on Kinetis Microcontrollers", Freescale Application Note, AN4533, Rev. 0, Jun., 2012.
6. Forbes Inc., "Automakers With The Lowest (And Highest) Recall Rates", http://www.forbes.com, Mar. 2014.
7. CNN., "100 million car recalls since the start of 2014", http://money.cnn.com, May 2015.
8. Shanker, R., Jonas, A., Devitt, S., Huberty, K. et al., "Autonomous Cars: Self-Driving the New Auto Industry Paradigm," Morgan Stanley Blue Paper, Morgan Stanley & Co. LLC, Nov. 6, 2013.
9. Statista Inc., "Car manufacturers' recall rate in the United States from 1985 to March 2014", http://www.statista.com, Sep. 2015.
10. "IHS Automotive Light Vehicle Engine Forecast: Engine Production", IHS Database, Nov., 2014.
11. Escherich, R., Ledendecker, I., Schmal, C., Kuhls, B. et al., "SHE -Secure Hardware Extension - Functional Specification", Version 1.1, Hersteller Initiative Software (HIS) AK Security, Oct. 16, 2009.
12. Weyl, B.; Wolf, M.; Zweers, F.; Gendrullis, T. et al.; "Secure on-board architecture specification", EVITA Deliverable D3.2, Aug., 2011.
13. Mercedes-Benz, "Electrical and electronic components in passenger cars up to 3.5t - General requirements, test conditions and tests - Part 1: Electrical requirements", MBN LV 124-1, Rev. Oct. 2010.
14. Liebetrau, T., Kelling, U., Otter, T., Hell, M., "Energy Saving in Automotive E/E Architectures", Infineon White Paper, Dec., 2012.
15. Miller C., Valasek C., "Remote Exploitation of an Unaltered Passenger Vehicle", Black Hat, Aug., 2015.
16. "BMW Group ConnectedDrive erhöht Datensicherheit. Auf Hinweise des ADAC schnell reagiert.", BMW Group Press Release, Jan. 2015

Content Information

Dipl.-Ing. Axel Freiwald
Infineon Technologies AG
85579 Neubiberg, Germany
axel.freiwald@infineon.com

Dipl.-Phys. Martin Klimke
Infineon Technologies AG
85579 Neubiberg, Germany
martin.klime@infineon.com

Dipl.-Ing. Klaus Scheibert
Infineon Technologies AG
85579 Neubiberg, Germany
klaus.scheibert@infineon.com

Dipl.-Ing. Björn Steurich
Infineon Technologies AG
85579 Neubiberg, Germany
bjoern.steurich@infineon.com

Definitions/Abbreviations

AES - Advanced Encryption Standard

BOM - Bill Of Material

CA - Certification Authority

CAN - Controller Area Network

CMAC - Cipher-based Message Authentication Code

CPU - Central Processing Unit

CRC - Cyclic Redundancy Check

ECC - Elliptic Curve Cryptography

ECU - Electronic Control Unit

eMMC - Embedded Multi Media Card

EV - Electrical Vehicle

FAR - Failure Analysis Return

HSM - Hardware Security Module

HW - Hardware

IP - Intellectual Property

MAC - Message Authentication Code

MCU - Micro Control Unit

OBD - On Board Diagnosis

OEM - Original Equipment Manufacturer

OTA - Over The Air

PCB - Printed Circuit Board

PPM - Parts Per Million

PKI - Public Key Infrastructure

RSA - Rivest-Shamir-Adleman cryptosystem

SFBL - Secure FLASH Boot Loader

SHA - Secure Hash Algorithm

SHE - Secure Hardware Extension

SOP - Start Of Production

SOTA - Software Over The Air

SoL - State of Life

SW - Software

TCU - Telematics Control Unit

TLS - Transport Layer Security

TPM - Trusted Platform Module

UDS - Universal Diagnosis System

ISSN 0148-7191

http://papers.sae.org/2016-01-0056