An Efficient and Secure Automotive Wireless Software Update Framework

Marco Steger, Carlo Alberto Boano, *Member, IEEE,* Thomas Niedermayr, Michael Karner, Joachim Hillebrand, Kay Roemer, Werner Rom

***Abstract*—Future vehicles will be wirelessly connected to nearby vehicles, to the road infrastructure, and to the Internet, thereby becoming an integral part of the Internet of Things. New comfort features, safety functions, and a number of new vehicle-specific services will be integrated in future smart vehicles. These include a fast, secure, and reliable way to diagnose and reconfigure a vehicle, as well as the installation of new software on its integrated electronic control units. Such wireless software updates are beneficial for both automotive OEMs and customers, as they allow to securely enable new features on the vehicle and to fix software bugs by installing a new software version over the air. A secure and dependable wireless software update process is valuable in the entire lifetime of a modern vehicle as it can be used already during vehicle development and manufacturing process on the assembly line, as well as during vehicle maintenance in a service center. Additionally, future vehicles will allow to remotely download up-to-date software on the electronic control units.**

To support this process over the entire vehicle's lifetime, a generic framework is needed. In this paper, SecUp, a generic framework enabling secure and efficient wireless automotive software updates is proposed. SecUp utilizes IEEE 802.11s as wireless medium to interconnect vehicles and diagnostic devices in a dependable and fast way. Additionally, SecUp is enabling beneficial wireless software update features such as parallel and partial software updates to increase the efficiency, and comprises advanced security mechanisms to prevent abuse and attacks.

***Index Terms*—Automotive systems, Generic framework, IEEE 802.11s, SecUp, Security concept, Wireless software updates**

I. Introduction

MODERN vehicles include a growing number of electronic control units (ECU) in order to incorporate new services. The latter require elaborate and often distributed software (SW) implementations on the integrated ECUs of a vehicle, which increase the complexity of the SW. Furthermore, the growing connectivity and distribution of vehicular systems is potentially introducing a growing number of bugs in automotive SW implementations and associated functions.

Fixing such SW bugs as well as upgrading the ECU SW to enable new features requires new concepts allowing efficient automotive SW updates. Thereby, these concepts are supporting the development and maintenance of modern vehicles. Efficient and secure SW updates can be beneficial over the entire life-cycle of a modern vehicle and will significantly reduce the time needed for vehicle maintenance.

In this paper, SecUp, a generic framework enabling wireless SW updates and diagnostics is proposed. Contrary to existing works only focusing on wireless remote updates such as [1], [2], [3], [4], SecUp supports the entire life-cycle of a modern vehicle considering the requirements coming from different application scenarios (i.e., vehicle development, assembly, and maintenance in a service center). In all these scenarios, wireless SW updates performed locally in a dedicated area (e.g., the service center or assembly line) have to be fast, efficient, and secure. To enable efficient and fast wireless SW updates, SecUp supports – besides the basic automotive wireless SW update functionality – beneficial features such as parallel SW updates (where the same SW binary is installed on different vehicles and ECUs simultaneously) and partial SW updates (where only changed parts of the SW are installed).

Additionally, SecUp utilizes a comprehensive security concept based on strong authentication and encryption mechanisms to guarantee secure wireless SW updates. This security concept also ensures the integrity of the transferred data and of the entire vehicle. As a result, SecUp is protecting the diagnostic devices, the transferred data, the vehicles, and the OEM backbone from unauthorized access.

The key contribution of this paper is the introduction of SecUp, a generic framework allowing efficient and secure wireless SW updates applicable for different automotive application scenarios, namely: (i) SW updates in the vehicle development, (ii) in the assembly line, as well as (iii) during maintenance in service centers. SecUp allows not only basic wireless SW updates, but additionally addresses the efficiency aspect by enabling parallel and partial SW updates. Furthermore, SecUp is built upon a novel security concept.

This paper is structured as follows. After reviewing related work in Section II, SecUp and its architecture are described in Section III. Additionally, the advantages of the employed IEEE 802.11s network are listed, proving its applicability to perform efficient and reliable wireless SW updates. In Section IV the roles of involved devices and users in the considered application scenarios is highlighted. Section V provides an overview on the performed security analysis as well as the resulting security concept. In Section VI, the developed prototypes of the core nodes are presented and different automotive ECUs used to evaluate SecUp are described. At the same time the developed features allowing parallel or partial SW updates are presented. To evaluate SecUp, first, in Section VII, SecUp's security concept and its impact on the system performance is analyzed. Second, the evaluation of SecUp is described and the corresponding results are presented in Section VIII. Finally, a conclusion is given in Section IX.

M. Steger, M. Karner, J. Hillebrand, and W. Rom are working at the VIRTUAL VEHICLE research center.

C. Boano, T. Niedermayr, and K. Roemer are working at the Institute for Technical Informatics, Graz University of Technology, Austria.

II. Related work

The benefits of automotive over-the-air (OTA) updates compared to traditional (i.e., wired) SW updates are listed in [1] and a high-level architecture for these updates is presented. However, a description of the wireless medium, required security mechanisms or other technical details are not included. The authors of [5] described the SW update process for ECUs based on international standards. However, this work is not addressing a wireless approach for such updates at all.

Other authors of previous works such as [2], [3], [4], [6], [7], [8] only focus on remote SW updates for vehicles and the corresponding security issues but neither consider other scenarios where updates are performed locally (e.g., within a service center), nor allow advanced SW update mechanisms such as parallel SW updates. A system allowing OTA updates was presented by Idrees et al. [2]. This system utilizes a Hardware Security Module (HSM) for data encryption, key management as well as to ensure data integrity on both the wireless interface and all ECUs of a vehicle. Therefore, a HSM is required on every ECU, which leads to significant extra costs. Additionally, the authors do not give any insight regarding the specific type or the properties of the wireless link. The authors of [9] also propose to use security HW modules on the ECUs and the vehicle gateway to secure OTA updates. Thereby, the authors focus on how to integrate these modules in the ECUs and evaluate the resulting resource overhead. The paper also contains a high-level description of an architecture for OTA updates, however, no implementations details or evaluation results are given. Nilsson et al. [3], [4] propose a system allowing automotive OTA updates by connecting a vehicle to a portal server using an Internet link. Thereby the authors list important security aspects, especially data integrity and data confidentiality, w.r.t. OTA updates but neither describe the utilized wireless network nor address the data flow in the network. In [7] an architecture for secure wireless SW updates sending multiple copies of the SW to ensure data integrity is presented. However, this solution is only addressing point-to-point links between one vehicle and an OEM server and relies on numerous prerequisites (e.g., ensuring data integrity of OTA updates by sending multiple copies). The authors of [8] also propose to send a SW binary two times to a vehicle to secure the update process without providing a detailed description on the actual SW update framework or other technical insights.

Although previous works are proposing different systems allowing remote OTA updates and addressing the corresponding security aspects, the listed solutions are only addressing unicast links (i.e., a point-to-point connection between the OEM and a specific vehicle) and hence, novel features such as parallel or partial SW updates cannot be realized. Contrary to these works, SecUp allows to install the latest SW on several vehicles simultaneously and provides efficient SW update mechanisms (i.e., parallel or partial SW updates).

To date (2017), Tesla is the only car manufacturer providing a solution for automotive OTA updates. A wireless network (either based on 3G/4G or a Wi-Fi network connecting the vehicle to the Internet) is utilized to transfer the latest SW from the servers of the OEM to a Tesla vehicle [10]. However, this point-to-point connection between the OEM and the vehicle cannot be used to simultaneously install SW in different vehicles and on several ECUs in parallel.

The authors of [2], [3], [4], [7] were mainly focusing on different security aspects of automotive SW updates and have listed a number of relevant security threats, namely: *data confidentiality*, *data integrity*, *key exchange and management*, and *vehicle integrity and authentication*. Different from previous work, the security concept proposed in this paper and presented in Section V-B addresses all these aspects at once.

A. IEEE 802.11s mesh networking and its security features

The proposed framework for wireless SW updates proposed utilizes a IEEE 802.11s network to interconnect vehicles and diagnostic HW. Today, it is the only solution using 11s as medium for wireless SW updates, however, the protocol is utilized in other automotive applications: the authors of [11] and [12] are using IEEE 802.11s as a backbone network for vehicle-to-vehicle and vehicle-to-infrastructure communication (V2X) networks to interconnect V2X entities and road side units. In [13] different wireless communication protocols such as IEEE 802.11 (Wi-Fi), Bluetooth Low Energy (BLE), IEEE 802.15.4 (ZigBee) and IEEE 802.11s are compared with each to find the most suitable protocol for wireless SW updates in an automotive environment. Thereby, the authors focus on different aspects such as throughput, scalability as well as extendability, show that IEEE 802.11s is the only protocol able to satisfy all investigated aspects, and point out weaknesses of other protocols (e.g., BLE and ZigBee offering insufficient throughput). Although the authors reveal the applicability of IEEE 802.11s for wireless automotive SW updates, the paper does not include any descriptions of a framework enabling these kind of updates. The benefits of utilizing an IEEE 802.11s network to perform wireless automotive SW updates are listed in Section III-B.

Several contributions mainly focusing on the security aspects of IEEE 802.11s have proposed different extensions and improvements. However, these aspects were not discussed w.r.t. automotive applications. Tan et al. [14] describe internal as well as external attacks on IEEE 802.11s networks and list relevant security requirements. The authors state that Simultaneous Authentication of Equals (SAE) [15] used in IEEE 802.11s is able to mitigate external attacks. However, a range of internal attacks is not prevented by SAE and therefore additional security features are required.

In [16] GPS positioning is used to mitigate a wide range of potential attacks on IEEE 802.11s. The proposed system called PASER is compared to other approaches for secure IEEE 802.11s networks in [17], and the authors show, beside others, the inefficiency of these approaches w.r.t. time and power. The use of GPS, as proposed in [16] and [17], however, is not applicable within the assembly line or a service center and therefore PASER cannot be utilized by SecUp.

The approaches proposed in [14], [18], [19], [16], [17] allow the creation of secure IEEE 802.11s networks, however, none of these approaches is able to fulfill both the efficiency requirements of wireless SW updates and the immunity against

possible attacks. The defined security concept presented in Section V is fulfilling both aspects by utilizing security mechanism on the network as well as the application layer. In particular, the proposed security concept is based on a structured security analysis. In [20] the latter was described in detail. This paper also showed how the analysis can be utilized to design automotive applications w.r.t. SAE J3016 [21], the new security standard in the automotive domain.

III. Framework and Architecture

This section describes the architecture of SecUp. A system overview is given in Section III-A and the utilized wireless network is addressed in Section III-B.

A. System Architecture and Core Nodes

Secure and efficient wireless SW updates will, independently from the application scenario, require a reliable and fast wireless network. Furthermore, a dependable smart gateway interface is needed for each vehicle to interconnect the latter with the wireless network in a reliable and secure way. This interface is the most critical component of SecUp and is called Wireless Vehicle Interface (WVI). A WVI interconnects the vehicular communication infrastructure (i.e., automotive bus systems such as CAN) and the ECUs of a vehicle with a wireless network and in further consequence with the Internet.

A WVI can be realized either as a fully integrated device (i.e., a smart bus gateway or a dedicated ECU) or as a plug-in solution. The latter can be temporarily connected to a vehicle using its OBD interface and is mainly utilized in service center scenarios, where the plug-in property of the WVI and the utilization of standardized in-vehicle communication protocols are ensuring backward compatibility as well as OEM-independence. In future vehicles the WVI will be part of the in-vehicle communication system, thereby enabling new services and functions which require to access data generated by or stored in the vehicle.

The so-called Diagnostic Tester (DT) can be seen as source of new SW versions within the wireless SW update framework. It can use a backbone link to the OEM to obtain the latest available SW as well as required information about the vehicle such as the vehicle configuration (e.g., types and IDs of the ECUs) or required authorization keys. Depending on the scenario and other general conditions, a DT can either be a dedicated device (e.g., tool in a service center) or more likely a SW application running on a laptop, PC, or server. Additionally, a DT typically supports various diagnostic functions. In typical OTA update scenarios, the DT application is running on a server of the car manufacturer and a secure Internet link (e.g., a VPN tunnel) between the WVI and the DT is established.

In SecUp, hand-held devices connected to the wireless SW update system are used by mechanics in a service center or by engineers during the vehicle development phase i) to run wireless diagnostics, ii) to monitor messages exchanged on the different bus systems of a vehicle, and iii) to trigger, monitor, and validate the SW update process.

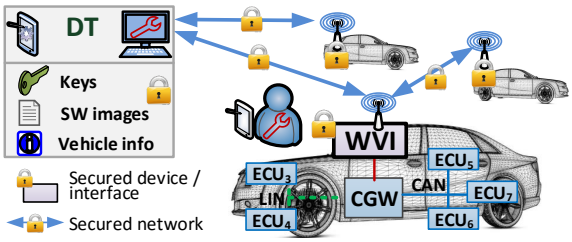

Fig. 1. An IEEE 802.11s network applied in a typical service center scenario. Mechanics use handhelds to run wireless diagnostics or wireless SW updates.

B. Wireless IEEE 802.11s network

To interconnect all involved nodes described in Section III-A, SecUp utilizes an IEEE 802.11s network. According to [13], this protocol is most suitable for wireless automotive SW updates and outperforms other wireless standards such as BLE or ZigBee in different important aspects like throughput, scalability as well as extendability. IEEE 802.11s is based on a mesh network topology, where each node will either directly communicate with other nodes in its transmission range or use other nodes in between to forward a data packet to the intended destination. The mesh characteristics of IEEE 802.11s allow a data packet to take different paths when sent through the network. This implicit redundancy increases the reliability of IEEE 802.11s networks. Similarly, the transmission range of a network can be increased by adding relay nodes (e.g., other vehicles or devices) at the edge of the network.

The multihop capability of IEEE 802.11s is a key advantage that significantly increases the reliability and the availability of such a mesh network, and this allows the use of an IEEE 802.11s network also in harsh radio environments.

A modern service center, as sketched in Figure 1 and further described in Section IV-B, is a typical example of a harsh environment, as wireless links can be affected by the shielding of vehicles or other (metal) objects, which potentially decreases the reliability of data transmission / collection. An IEEE 802.11s network, however, is able to provide a stable communication: if a direct link between the vehicle and the DT is too weak to exchange a packet, other vehicles or IEEE 802.11s relay nodes located in between will forward the data packets to the target vehicle or the DT.

The IEEE 802.11s standard includes multicast data streams in mesh networks. Such a multicast can be used to send data packets from one node to several other nodes in a network. SecUp can hence potentially use such layer two multicasts to transfer and install a new SW version on several vehicles simultaneously (i.e., to carry out parallel SW updates). Although the IEEE 802.11s standard defines multicast data streams, the current open11s implementation [24] used in the developed framework does not support multicasts on layer two yet and therefore multicast is implemented on higher layers.

IV. Supporting the Entire Vehicle Lifetime

In this section, the considered scenarios for wireless SW updates are listed first. Thereafter, an illustration of wireless SW updates in a service center is given in Section IV-B.

A. Wireless software update scenarios

SecUp can satisfy the requirements of wireless SW updates in the following scenarios: i) during vehicle development, ii) in the assembly line, and iii) in a service center. In the following, the scenario-specific requirements are highlighted, and information about involved users, available infrastructure, and related security concerns are given.

1) Vehicle development: during the vehicle development phase, engineers will have to update the SW of one or more ECUs of a test vehicle several times to analyse, compare, and evaluate newly developed features. Therefore, the development engineers require a flexible and efficient system enabling wireless SW updates as well as vehicle diagnostics. Vehicle development activities will mainly take place in restricted areas and will be performed by engineers (i.e., expert users). The wireless solution offers several advantages in this scenario, as it enables fast and flexible SW updates while allowing the usage of hand-held devices like tablets or smartphones.

2) Vehicle assembly: this step is performed in a highly automated and secure environment where many operations are performed by machines and robots. Before a vehicle can leave the assembly line, the latest SW is installed on all integrated ECUs of a vehicle. Therefore, the SW of many vehicles must be updated – ideally in parallel – to install the latest SW on all ECUs. Because of the high number of vehicles as well as the high degree of automation, scalability, reliability and efficiency of the SW update system are very important.

3) Vehicle maintenance: in a service center mechanics will diagnose, repair, and maintain several vehicles. Therefore, a mechanic will connect to a vehicle to run diagnostic functions, to check if there are any Diagnostic Trouble Codes (DTC) sent by the vehicle and its ECUs, and to perform the necessary repairs. Thereby, the mechanic will also check if new SW is available for one or several ECUs of the vehicle and install it. Simultaneous wireless SW updates would be very beneficial especially if large vehicle recalls (e.g., due to a critical software bug) are necessary: a mechanic can connect to several vehicles in parallel and install the new software simultaneously. Additionally, a wireless solution can be very beneficial in service centers as a mechanic will not have to use heavy diagnostic equipment (e.g., a PC and a battery contained in a solid metal case), but can run wireless diagnostics and SW updates utilizing a lightweight hand-held device instead.

4) Remote SW updates: mainly relevant for future vehicles with an integrated WVI, either realized as bus gateway or as dedicated ECU, allowing wireless connectivity via 3G/4G or Wi-Fi. The current version of SecUp is mainly focusing on SW updates performed in local environments, however, the developed SW updates mechanisms could also be used when connecting the vehicle to the wireless network of the user.

Scenario-specific particularities are summarized in the following: SW updates in the assembly line or during the vehicle development phase are performed by expert users in secured and restricted areas. In these use cases security is an important issue (e.g., industrial espionage) but not as critical as in OTA scenarios, where the update is performed by an untrained user at the users' home or in public, potentially utilizing a compromised device or an insecure network. Especially the service center and the assembly line scenarios require a very efficient and fast way to install SW updates. During vehicle development high flexibility by easily extending the transmission range of the wireless network is especially required due to the big variety of function tests, system evaluations, and diagnostics performed during this phase.

B. Wireless software updates in a service center

In this section the required steps to install latest SW on a vehicle's ECU are described by utilizing a typical service center scenario as shown in Figure 1. In this sketched example, mechanics maintain vehicles by using handhelds to run wireless diagnostics and to perform wireless SW updates. Similar procedures and schemes apply in the other update scenarios. The main goal of this section is to explain the basic SW update procedure without providing details on the involved security mechanisms, as this is described in Section V-B.

1) Connecting the vehicle to the SW update system: a mechanic first connects a WVI to the vehicle (i.e., if the vehicle doesn't have an integrated WVI) using its OBD interface. The WVI joins the IEEE 802.11s network and will connect to a DT once a periodically broadcasted beacon advertising the presence of the DT was received. The same principle is used by the handhelds to connect to the DT (i.e., after receiving a DT beacon) as well as to WVIs (i.e, by sending beacons). It is important to note that in case of an update, the SW is directly transferred from the DT to the WVI and is never stored on the handheld devices due to security reasons.

2) Gathering information about the vehicle: once the connections between the DT, the handheld device and the WVI are established, the DT starts to gather information about the vehicle by retrieving the Vehicle Identification Number (VIN). With the retrieved VIN, the DT can either query its local database or access an OEM server to obtain vehicle-specifics including vehicle model and variants, the integrated ECUs including ECU-IDs and the CAN-IDs, as well as available SW versions. This information is often condensed in so-called Open Diagnostic data eXchange (ODX) [25] files.

3) Performing the wireless SW update: if new SW for a vehicle is available, the SW binary will first be fully transferred from the DT to the WVI. The latter will then verify the received SW binary and start to install the binary on the ECU utilizing the Unified Diagnostic Service (UDS) protocol [26]. Besides this basic SW update mechanism, SecUp also supports *partial SW downloads*, where only the difference between the current and the new SW version is sent to the ECU. This feature can significantly reduce the duration of a SW update, however, it has to be supported by the ECU (see also Section VI-D3). To perform a SW update on several vehicles in parallel, a mechanic will first register vehicles for a certain SW update on the DT, which will then automatically start the update once all vehicles are ready to receive the SW. Parallel SW updates can again significantly shorten the duration of wireless SW updates and thereby reduce costs as several vehicles are addressed simultaneously.

V. Security and trust

Security is a critical aspect of wireless SW updates, as an attacker can compromise involved devices and protocols to

reveal sensitive data and keys. Depending on the application scenario, different attack vectors have to be considered and therefore different levels of security are required.

SecUp's security concept is built on a system-centric design employing a measurable security approach as described in [27]. The used design approach, the DEWI security metric, is based on a structured system decomposition rather then a traditional approach where first attack vectors are analyzed and second corresponding countermeasures are implemented.

Such a system-centric design encompasses the following essential steps: i) **security analysis** of the framework resulting in a secure system configuration, ii) **extraction of the security requirements** from the secure system configuration, iii) **Security concept definition** based on these requirements and the peculiarities of the application scenario, and iv) **evaluation of the defined security concept** using the STRIDE threat model [28]. [20] provides more detailed information and also highlights that this approach can be aligned with the new SAE security standard SAE-J3061 [21].

In the following the most important revealed security threats are stated and thereafter, in Section V-B, the security concept is described in more detail. This description also shows how the SecUp's security concept is able to mitigate these threats. More detailed information can be found in [29].

A. Security threats and attack vectors

A system allowing wireless automotive SW updates is a worthwhile target for a wide range of attacks. In the following we will collect and describe critical external as well as internal threats and show in a later step that the security mechanisms employed by SecUp are able to prevent these threats.

An external attacker can try to eavesdrop the wireless channel to reveal transferred authorization keys and the latest SW itself, or tamper with the transferred data to install malicious SW on the ECU of a vehicle (Threat T1).

Besides the aforementioned external attacks (e.g., committed by a hacker without access to the wireless network), there are also a range of internal threats performed by insiders such as a rogue mechanic in a service center or other users fraudulently using the wireless network. These insider threats encompass theft of equipment such as a WVI to perform unauthorized SW updates (e.g., tuning; T2), to extract secret keys stored on the device (T3), or spoofing the identity of an WVI or a DT (T4) to gather keys, the SW, or vehicle access.

To mitigate all (aforementioned) threats, SW update frameworks like SecUp must i) employ strong authentication schemes to establish trust between the involved devices, ii) use strong encryption as well as message authentication codes (MAC) to protect confidentiality as well as the integrity of the transferred data, iii) securely store sensitive data and secret keys in dedicated memory or HW security modules (HSM).

B. Security concept

The security concept presented in this section is built upon the results of the system analysis utilizing the DEWI security metrics. The resulting security concept is based on security mechanisms on the network as well as on the application layer.

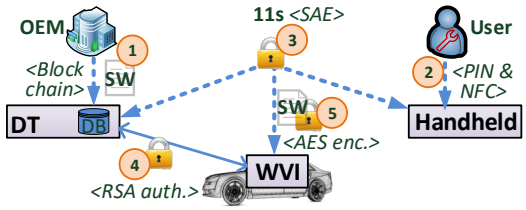

Fig. 2. Security architecture and flow. New SW is first securely distributed by the OEM [30] and stored on the DT. To secure the local update 2) a user authentication is performed, 3) a secured wireless network is established, 4) strong authentication is used between the core nodes, and 5) symmetric keys are used to securely transfer the new SW to the WVI.

On the network layer, *wpa_supplicant*, a generic security framework for different types of wireless networks, is utilized to secure the IEEE 802.11s mesh network. The *wpa_supplicant* is part of the current open11s implementation and allows to secure the network in a lightweight way by using SAE [15]. SAE was developed especially for 802.11s-based, multihop capable mesh networks, is fully integrated in the latest *wpa_supplicant* version, and is able to mitigate a wide range of external attacks [14]. However, SAE is not providing suitable countermeasures against internal attacks, where attackers use nodes already connected to the wireless network (e.g., a compromised device or a rogue user such as a mechanic in a service center; threats T2-T4) to launch an attack. Therefore, additional security features are employed on the application layer to prevent internal attacks and thereby to address the four important security aspects, namely, vehicle integrity and authentication, data integrity, data confidentiality, key management and exchange.

To ensure the integrity of a vehicle, especially when equipped with a WVI, strong authentication mechanisms must be applied to keep unauthorized users from accessing the vehicular bus system. Additionally, it is even more important to avoid that an attacker endangers a whole fleet of vehicles by breaking one vehicle and extracting a shared (i.e., symmetric) authentication key. The developed concept is based on *unique asymmetric key pairs* consisting of a private and a public key used on the WVIs, handhelds, as well as DTs to ensure an unambiguous authentication between all nodes.

In a classic Public Key Infrastructure (PKI) the public keys are exchanged (e.g., over the Internet) and then used to encrypt data or to verify digital signatures. To allow a user to check if a public key really belongs to a certain entity, classic PKI systems employ a third party, the Certificate Authority (CA). As a consequence, each node requires an Internet connection to communicate with a CA.

In SecUp, a different approach has been chosen: to keep the system local (i.e., only the DT is connected to the Internet and/or the OEM backbone), a security concept without a CA was designed. However, such a concept requires that public keys are initially exchanged and then securely stored. This pairing step is performed in a controlled environment (e.g., close proximity to the DT) using a dedicated media or mechanism (e.g., SecUp supports NFC and the use of one-time passwords) by authorized users (e.g., head of a service center). The pairing only has to be done once: first, a *master key pair* is created on each device and securely stored in dedicated

memory or in a Trusted Platform Module (TPM). Second, the public keys are exchanged and then securely stored. After the initial pairing step, the master keys can be used to handle the authentication between the involved nodes and, additionally, to sign as well as to encrypt unicast packets.

As a WVI securely stores the public key of the DT (and vice versa), digital RSA-based signatures (RSA signature with 2048bit key length and SHA-256 hash) can be used to authenticate the involved parties. In the authentication step (e.g., performed on a daily basis), the entities agree on a symmetric AES-GCM-based session key which is used to ensure confidentiality and integrity of the data transfer. Please note that in SecUp, i) data packets are protected using session keys and not by utilizing the master keys as the use of symmetric keys is more efficient, and ii) timestamps and nonce are employed to protect exchanged messages and thus to mitigate common attacks such as replay attacks, similar to TLS and other common secure message transmission protocols.

Secure multicast data streams required to allow parallel SW updates, however, need symmetric keys to verify and encrypt data. Data sent from a DT to several vehicles must be signed and encrypted using a shared key. This shared *session key* is created by the DT and then distributed individually to all WVIs using a unicast packet signed with the private master key of the DT. In further consequence, multicast data packets are encrypted and signed using the *session key* to ensure the confidentiality and the integrity of the exchanged data.

By utilizing the *master key pairs* and the (shared) symmetric *session key*, the security aspects authentication, confidentiality, and integrity can be solved w.r.t. the vehicle itself, as well as on data exchange level (unicast as well as multicast). To prevent an attacker from extracting the utilized keys (e.g., by stealing a WVI), these keys must be stored securely on the devices and kept secret all the time. Therefore, keys used in SecUp are stored in dedicated secure memory or TPMs.

C. Application of the security concept to a use case

The service center scenario sketched in Section IV-B will again be used to illustrate how to secure wireless SW updates by applying the defined security concept (see also Figure 2):

1) User authentication: mechanics authenticate with the system using a NFC smartcard and a PIN code. Different user profiles are used to authorize different modes: normal mechanics can use SecUp to run wireless diagnostics only. A privileged user can additionally perform wireless SW updates.

2) Interconnecting the WVI: after connecting the WVI to a vehicle, the WVI powers up and connects to the IEEE 802.11s network using a shared network key.

3) Authentication between WVI and DT: the master keys of the WVI and the DT are used to authenticate with each other.

4) Parallel SW updates: first, the DT creates a session key and distributes it to all concerned WVIs (unicast). Second, the DT sends the fragmented SW binary to the WVIs by signing and encrypting every packet using the session key (multicast).

5) Data verification: to ensure data integrity, the DT first computes the hash value of the SW binary. Then the DT signs and encrypts the hash value using the master keys and sends it to the concerned WVIs (unicast). Hence, these WVIs use the transferred hash to verify the received SW binary before installing it on the concerned ECUs. Please note that the genuineness of new SW (i.e., that SW is really coming from the OEM) is verified by the DT before a new SW binary is stored in the local database of the DT. However, this secure SW distribution process (see [30]) is out of the scope of this paper. In the current concept, the WVI is not verifying the genuineness again, as the DT already verified it.

D. Formal security concept evaluation

The defined security concept was first formally evaluated using the Microsoft STRIDE threat model [28]. STRIDE is an attack-centric approach that can be used to analyze the security of a system by identifying a number of potential security threats and grouping them into six categories: **S**poofing, **T**ampering, **R**epudiation, **I**nformation disclosure, **D**enial of service, and **E**levation of privilege. To apply STRIDE, each part of the system is analyzed and all potential threats for every component or process are determined. Based on these threats, suitable countermeasures can be defined in a subsequent step. SecUp's security concept evaluation was performed in a similar way: first a list of potential STRIDE threats was created, and a prioritization of these threats was performed w.r.t. likeliness and severity. Second, these threats were (theoretically) applied to the security concept and its security features to prove that suitable countermeasures exist to avoid all of these threats. In [29] the evaluation process is described in more detail and the results of the formal analysis are presented.

Due to space constraints a complete threat analysis and a system evaluation w.r.t these identified threats is not possible. However, in the following the threats T1-T4 described in Section V-A are used to show the security features of SecUp. Threat T1 is addressing confidentiality and integrity of exchanged data. In SecUp, exchanged data is secured by i) SAE on network level, used to protect the IEEE 802.11s network and mitigating external attacks, and ii) security features on the application layer encompassing per-packet encryption and integrity checks as well as a final verification of the transferred SW binary by sending the hash value of the binary (encrypted and signed by the DT) to the WVI.

Threats T2 and T3 are addressing theft of equipment (e.g., a plug-in WVI or a handheld in a service center). SecUp allows wireless SW updates within dedicated areas like a service center: after an initial pairing step between the DT and a new WVI, the used equipment is forming a trusted network and SW updates will only be allowed when a WVI is connected to a trusted DT. A stolen WVI can therefore not be used to perform unauthorized SW updates (T2), as the WVI will not allow an attacker to perform an update without being connected to the trusted DT. The attacker can in a second step try to spoof the identity of the trusted DT (T4), however, due to the employed authentication step using digital signatures, also this attempt will fail. An attacker can also use stolen equipment to read or replace keys stored on the device, however, will fail as SecUp utilizes TPMs to securely store keys and other sensitive data.

VI. Implemented prototypes and demonstrators

In this section the implemented HW and SW prototypes of the core nodes of the wireless SW update system are described and the utilized demonstrator ECUs are presented.

A. The developed WVI prototype

The developed WVI prototype consists of a BeagleBone Black (BBB) board and a developed Printed Circuit Board (PCB). This PCB can be mounted on the BBB using the designated header pins and encompasses the required HW interfaces (i.e., CAN and OBD) as well as the corresponding transceivers to interconnect a WVI and a vehicle. The PCB is also responsible for the battery management of a WVI: the battery is used as power supply of the WVI when the ignition of the vehicle is off and no power is provided via the OBD interface of the vehicle. The SW implementation of the WVI is mainly done in Java and the Java Native Interface (JNI) is used to interact with the HW-related parts of the WVI prototype.

A key aspect of the developed security concept is to use strong authentication mechanisms between the involved devices (i.e., WVIs, handhelds, and DTs). The corresponding authentication keys must be kept secret to guarantee a trustworthy system. Especially the plug-in WVI is critical, as an attacker can steal such a device and try to extract the secret keys. To avoid that, a TPM can be used on the WVI to securely store secret keys and other sensitive material. Although, the TPM is currently only used to store RSA keys and to handle the authentication between the WVI and the DT, it would also allow the use of certificates in future versions of SecUp.

1) Trusted platform module integration: The used TPM from NXP is connected to the WVI via Inter-Integrated Circuit (I^2C) bus. The I^2C protocol required to exchange data between the TPM and the WVI was not fully supported by the I^2C library available on the BBB. Because of that, an additional I^2C bus using normal I/O pins was implemented on the BBB to fulfill the requirements of the TPM w.r.t. the I^2C communication protocol. The disadvantage of this approach, however, is that the communication between the TPM and the BBB is rather slow: about 7 kBaud.

In Section VII-A the impact of the slow bus connection between the WVI and the TPM is evaluated by first comparing the performance of the integrated TPM with the software-based cryptography library Java Bouncy Castle (JBC) and second using an oscilloscope to analyze the timing behavior of the TPM when performing different cryptographic operations.

B. Prototypes and implementation of the DT and handhelds

The DT implementation was developed in Java and tested on both a dual-core laptop running Win7 as well as on a BBB running Debian Linux (the system evaluation was performed by an BBB running as DT). The latest DT implementation supports different modes such as ECU programming, monitoring of the CAN bus, and OBD diagnostics.

A Nexus 7 Android tablet is used as hand-held device. It can be connected to a WVI and a DT simultaneously, thereby empowering the hand-held device to monitor the bus systems of the vehicle, to run OBD requests, and to trigger SW updates.

C. Volvo FlexECU used for basic framework evaluation

The Volvo FlexECU is a prototyping ECU mainly used to test new applications. The ECU offers several connectors (i.e., CAN interface, power supply, and enable pin) and comes inside a solid metal case. The bootloader of the ECU contains an UDS stack and thus supports a UDS-compliant SW update process. A reset must be triggered to access the bootloader when the ECU is already running its application SW.

This ECU was used to perform basic evaluations of the SW update framework (Section VIII). Two slightly different SW versions, one periodically sending CAN frames with ID 1001 and the other with ID 2001, were used to test the developed framework and to run some basic performance analysis. Due to the different CAN-IDs it is easy to validate that a new SW version was correctly installed on the ECU.

D. Infineon AURIX as advanced ECU demonstrator

The Volvo ECU is not suitable to test advanced features such as delta or partial wireless SW updates as neither the FlexECU HW nor the SW running on the ECU can be adapted or extended. Because of that, a second demonstrator ECU based on an Infineon AURIX was developed. Utilizing AURIX brings several advantages and offers more freedom to develop advanced SW update features, as the default bootloader can be modified or even a new AURIX bootloader can be developed.

1) AURIX platform description: the developed demonstrator ECU consists of an Infineon AURIX multi-core ECU integrated in the AURIX application kit TC277 TFT. It is a high performance ECU compliant to support safety requirements up to ASIL-D, the highest automotive safety level [31]. The AURIX ECU is based on a 32 bit scalar TriCore CPU running at up to 300 MHz in the full automotive temperature range $\{-40, +170\}$°C. It is equipped with up to 4MB flash and 472KB RAM memory and comes with high speed CAN transceivers, a safety processor and watchdog, as well as dedicated closely-coupled memory areas per core.

2) Flash-over-CAN mechanism for AURIX: this mechanism is the basis for wireless SW updates as it encompasses the transfer of new SW from the WVI to the AURIX ECU utilizing the CAN bus as well as the installation of this SW on the ECU. AURIX does not provide such a mechanism by default. Therefore an extended AURIX bootloader enabling a reliable flash-over-CAN mechanism was developed. The latter basically consists of three main components: i) a *CAN driver* handling the data transfer over CAN, ii) a *flash driver* required to program and erase the flash memory, and a iii) *controller* coordinating the programming sequence.

The CAN driver itself is already implemented in the AURIX basic SW and must only be configured to support the right bitrate (e.g., 1Mbit/s) of the bus. The flash driver is responsible for programming and erasing the data as well as the program flash memory of the ECU and basically consists of two main functions: erasing of sections and programming of pages. The flash driver is able to erase one or more consecutive sections and can write single pages or use the burst programming mode to write multiple consecutive pages of the flash memory. The controller component was developed compliant to the UDS

standard and is started each time the AURIX bootloader is called (e.g., after a HW or SW reset). It initializes the CAN driver and checks for programming requests. If no such request was received, it starts the ECU's application SW or reboots if no application SW is present. Otherwise, if a programming request was found, the controller handles the data transfer as well as the installation of the new application SW.

New application SW can be installed on AURIX by first sending a programming request containing the start address of the SW to the ECU. Second, an authorization step based on a typical Seed & Key mechanism is performed and then an UDS-compliant programming session is started on the ECU. In the third step, the SW is transferred to the ECU block-by-block using the corresponding UDS commands. The ECU receives a block, stores it in a temporary buffer and finally utilizes the flash driver to write it to the flash memory. After transferring all blocks to the ECU, the new SW version is validated by computing the CRC over the new SW blocks. Finally, the ECU reboots and starts the new application.

3) Partial SW updates: transferring the SW binary to the ECU via CAN bus is, according to Section VII-C and the results shown in Table I, by far the most time consuming step when performing a wireless SW update. Speeding up this step will significantly decrease the duration of a SW update.

One way to achieve this is to utilize partial SW updates. The basic idea behind partial SW updates is to only update the parts of the software that have changed compared to the currently installed SW version. The possible benefit can be up to nearly 100% in case of a simple parameter value change: a normal SW update will require to download the entire new binary to the ECU regardless of whether the old and the new binary are very similar (i.e., just one or a few parameters have changed) or if the new version has a lot of new features and is therefore much bigger in size. Partial SW updates will, in contrast, only download the changed parts of the binary (i.e., one or several memory blocks). In flash architectures, the available memory is divided into different sections, sometimes even with different sizes. To manipulate a section, a flash driver first has to erase the entire flash section before it can be filled with new content. Therefore, it is not possible to just update the value of one parameter stored in a specific section, but the entire memory section has to be rewritten. For partial SW updates this fact leads to two different approaches:

Delta download: only relevant (i.e., the changed) parts of a SW section are sent to the ECU. On the ECU the affected section is copied into RAM memory, the flash section is erased by the flash driver, the content of the section (in RAM) is modified using the received delta bytes, and finally the resulting section content is written to the flash memory again. This approach on the one hand minimizes the amount of data to be transferred to the ECU, but on the other hand significantly increases the complexity at ECU level.

Partial SW update: instead of only transferring delta bytes, the entire section is transferred to the ECU. Although, this limits the benefit of a partial SW update as more data must be sent to the ECU, it is very simple to implement.

To decide whether to use delta downloads or to utilize partial SW updates, different factors such as section size, bitrate of the bus, as well as the bus load must be taken into account.

Fig. 3. Flash memory is divided into sections with different block sizes.

For AURIX ECUs the partial SW update approach was chosen as its flash memory is divided into sections with different sizes (see Figure 3) and as the SW developer can utilize a linker file to influence where specific parts of the SW binary are stored in the flash memory. When developing new AURIX SW, the binary shall be divided into different sections and mapped to the ECU flash memory accordingly as illustrated in Figure 3: areas that are most likely unchanged in the vehicle's lifetime (e.g., basic ECU functionality) and areas that are subject to change (e.g., blocks containing parameters or optional features). The latter will be located in one or several small sections (mostly a few KBs) of the flash memory.

VII. Evaluation of the Security Concept

In this section the impact of the employed security mechanisms on the system performance is evaluated.

A. Evaluating the performance of the integrated TPM

In SecUp a TPM is integrated in the WVI to securely store secret keys and sensitive data. The used TPM is capable to store keys and to perform cryptographic operations such as RSA encryption/decryption or RSA/AES key creation. The TPM can also be used for symmetric data encryption and signing. Therefore all cryptographic tasks required by the defined security concept are supported by the TPM. However, they can also be performed by the used Java SW-library JBC.

In a first evaluation step the performance of the TPM and the JBC is compared. Therefore, the authentication step between a WVI and a DT is performed 20 times first using the TPM and second utilizing JBC and thereby the duration of these operations was measured. The authentication step including different RSA operations (i.e., data encryption/decryption and signing) can be performed by the TPM, the JBC library, or as hybrid solution utilizing both in a combined way.

The gathered results show that the JBC outperforms the TPM: min/max/avg duration when using the TPM (6328/6588/6415.5ms) and JBC (575/701/635.8ms), respectively. However, the most important advantage of the TPM is its ability to securely store sensitive data and keys. Because of that, the final concept uses a hybrid solution (3525/3690/3577.9ms) where the fast JBC library is combined with the secure storage of the TPM. As the performance difference between JBC and the TPM, a HW chip dedicated to perform cryptographic operations, is not really obvious in the first place, an additional evaluation was performed to get to the bottom of this issue.

As mentioned in Section VI-A1, the I^2C bus connecting the TPM with the WVI was realized using the I/O pins of the BBB and therefore the speed of the bus is rather slow (7

kBaud). To evaluate the impact of the slow bus connection on the overall performance of the TPM, several RSA operations (RSA encryption/decryption with 2048 bit key, 208 byte data) were performed and meanwhile the timing behavior of the TPM was analyzed using an oscilloscope. The measurements clearly show that the I^2C bus has an significant impact on the overall performance as the time required for the data exchange between the TPM and the WVI is about 45% for RSA decryption and up to 87% for RSA encryption of the overall duration of the operation. The measurements reveal that either a faster I^2C bus or a different bus such as Serial Peripheral Interface (SPI) would help to significantly decrease the overall duration. In the current setup, the TPM shall only be used to store keys and to perform the authentication, but not for symmetric encryption using AES (i.e., hybrid solution).

B. Impact of security mechanism on the network layer

In this section, the impact of SAE on the system performance is analyzed by evaluating the Round Trip Time (RTT) and the code size of the IEEE 802.11s kernel module.

The first evaluation step is about evaluating the RTT – the time needed to send a request packet from node A to node B plus the time needed to transfer the response packet from node B back to node A. For this experiment five IEEE 802.11s nodes were used. Static paths through the network were defined to force multi-hop routes with different lengths (from direct connections to multi-hop routes with up to four hops). For each measurement with different path lengths involved and either SAE on or off, 10000 messages were sent from node A to node B. To measure the RTT, up to five BBBs (BBB0 to BBB4) were used to send UDP packets from BBB0 to BBB4 and back, using zero to three boards (BBB 1, 2, and 3) in between to forward the data. On BBB0 and BBB4, a UDP server-client application was used to i) send the request (BBB0), ii) receive this request and send the response (BBB4), and iii) to receive this response (BBB0). An adapted version of IEEE 802.11s was used to detect the exchanged UDP test packets. On BBB0 and BBB4 the received and sent timestamps were added to the test packets. These packets were collected on BBB0 during a test, and were used in a subsequent evaluation to compute the network layer RTT (by removing the time spent on application layer; only relevant for BBB0 and BBB4, as packet forwarding on BBBs 1-3 is only done on network layer).

In Figure 4 the results of these measurements are shown. The green, continuous line shows the RTT for different numbers of hops with SAE on and the blue, dashed line for the measured RTT with SAE disabled. Figure 4a shows that each hop significantly increases the RTT median (similar results were obtained when analyzing the average RTT). Additionally the delta of the median values is presented in Figure 4b: each hop increases the RTT, as a packet has to be encrypted and decrypted at each hop in between node A and node B.

In a second evaluation step, the impact of SAE on the code size of the MAC80211 kernel module was analyzed using the source code of this module (by default with SAE) of a standard 3.19 Linux kernel: first, the module with SAE included was compiled, and second, all the SAE-related source code was removed and the resulting module was compiled (and tested) once again. The results of the evaluation show that the impact of SAE on the code size of the IEEE 802.11s kernel module is about 5% w.r.t lines of code (LOC) (58545 LOC without SAE, 61354 LOC with SAE) and 3% w.r.t. the size in memory (774 KB without and 801 KB with SAE).

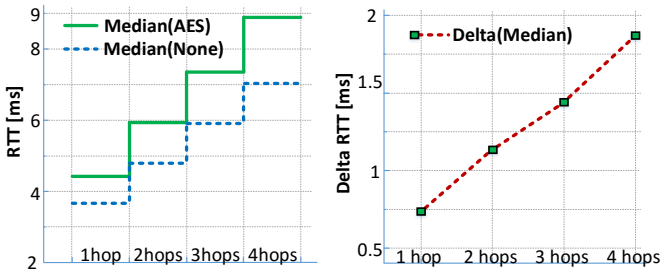

Fig. 4. Impact of SAE on network layer: SAE enabled vs. SAE disabled (i.e., None). Median (a), and delta(median)=median(SAE)-median(None) (b) of the RTT measurements using 10000 UDP packets are shown.

C. Impact of security mechanisms on the SW update duration

To secure and protect SecUp, different security mechanisms on network and on application layer are used. These mechanisms have an impact on the overall performance of the developed system. Latency measurements in the scope of real wireless SW updates using the Volvo FlexECU were performed to assess this impact. In the following, all steps required to successfully perform a wireless SW update using the developed framework are listed: i) WVI discovery and connection establishment, ii) authentication process between WVI and DT, iii) SW update initialization (mainly w.r.t. the ECU), iv) authorization on the ECU using a Seed&Key procedure, v) wireless data transfer from the DT to the WVI, vi) data download to the ECU via CAN, and vii) validation of the downloaded SW on the ECU.

Each step adds latency to the overall duration of a wireless SW update and the per-step-latency differs depending on the used security mechanisms. For this experiment the FlexECU and the corresponding binaries consisting of the application SW (378KB) plus the secondary bootloader (67KB) were used. On the application layer, security mechanisms implemented in SW utilizing the Java Bouncy Castle (JBC) were employed.

The gathered evaluation results presented in Table I reveal that i) the data transfer via CAN comprises most of the SW update duration (75% if all security mechanisms are enabled and up to 87% if disabled) and ii) that the employed security features increase the overall time required to carry out the wireless SW update by 20%. Although this duration increase is quite significant, it must be accepted as all security mechanisms are required to guarantee a secure system execution.

VIII. Framework Evaluation

In this section, the evaluation of SecUp and its wireless SW update mechanisms is presented. The performed evaluation is the first available analysis of a wireless SW update framework providing advanced SW update mechanisms as well as allowing secure and efficient SW updates in different

(local) scenarios such as an assembly line or a service center. Therefore, no suitable benchmark is available to compare the gathered results. A basic system evaluation was already presented in [32]: a developed Vehicle and ECU Model (VEM) was used to perform fundamental experiments, system evaluations, and communication tests. Furthermore, the VEM was used to evaluate the behavior of the developed system in case of errors (e.g., sending unexpected frames) and communication problems (e.g., lost, delayed or duplicated CAN frames). This evaluation also included an analysis of the wireless network, where the DT is connected to a real vehicle and run wireless diagnostics using the developed WVI prototype. Thereby, the evaluation results were used to prove the applicability of IEEE 802.11s as media for wireless SW updates.

A. Wireless SW update analysis using Volvo FlexECU

The SW update process was first analyzed using the FlexECU, provided by Volvo Trucks and described in Section VI-C, connected to a WVI. This ECU is programmed in two steps: first, the secondary bootloader (SBL) is transferred to the ECU and then launched on the ECU using a specific UDS command. Second, the application binary is sent to the ECU and installed on it. SecUp supports both the use of an SBL and the application binary as well as an approach where the application binary is directly installed without using an SBL.

In Table II the duration of the wireless data transfer, where the SBL and the application SW is transferred from the DT to the WVI, is compared with time required to install the SBL and the application SW on the ECU via CAN. The measured SW update duration using the Volvo ECU is also used as a benchmark for the evaluation of AURIX and its implemented SW update features. The results reveal that the wireless data transfer (including data transfer, integrity check, etc.) is 12 times faster than forwarding the binary using CAN and thereby corresponds to the results presented in Table I. Thus, it makes sense that the WVI autonomously controls the data transfer to the ECU once the binary was received from the DT (i.e., no permanent wireless connection to the DT needed). The WVI then informs the DT when the SW is installed on the ECU successfully or when any problems occur.

B. Wireless SW update analysis using AURIX ECU

The AURIX ECU described in Section VI-D was used to further evaluate SecUp and its features. Therefore, the SW implementation of SecUp's core nodes, the DT and the WVI, were running on BBBs. These BBBs are connected to a measurement PC via USB to control the measurements and to collect the results. Per measurement campaign (i.e., one per mode) 20 wireless SW updates were performed and the duration of each single step, as already described in Section VII-C, was measured. These steps are then grouped in i) *Connection-related*, including the DT discovery and the authentication between DT and WVI, ii) *ECU-related*, encompassing the initialization of the SW update and the corresponding authorization step between WVI and ECU, iii) *Upload*, i.e., the wireless data transfer, and iv) *Download*, the data transfer via CAN as well as the installation of the SW on the ECU. This setup allows to compare the overall duration of a SW update w.r.t. the used update mechanism and additionally to analyze the latency added by each step.

To compare the SW update duration of both the Volvo FlexECU and the AURIX ECU, a SW binary for AURIX was created using Hightec Studio, the recommended SW development tool for AURIX. The resulting .hex file (i.e., the binary) was developed to have the same size as the Volvo FlexECU secondary bootloader (SBL) plus the size of the application SW (i.e., 67 KB + 378 KB = 445 KB). Although the .hex file contains all the information needed for the wireless SW update, the file format is not suitable to directly analyze the binary w.r.t. to partial SW updates and therefore a parser was developed to transform the .hex file in a so-called .phex (i.e., parsed hex) file format. In a .phex file, each line represents a block of the SW binary and it can therefore be used to check if a partial SW update is possible by comparing the lines of the SW versions. A further advantage of this file format is that the .phex file is smaller than the corresponding .hex file: the binary used for the evaluations is originally stored in a .hex file with 445 KB and can be transformed to a .phex file of about 316 KB.

1) Volvo FlexECU and AURIX ECU comparison: in a first evaluation step the SW update duration using both the FlexECU and AURIX are compared. The results presented in Table III show that AURIX can be updated more than twice as fast. This is due to: i) the initialization process on ECU level (i.e., start a programming session and handle authorization) is ten times faster on AURIX compared to the FlexECU due to the higher CPU power available, ii) the wireless data transfer is about 30% faster as the .phex is 29% smaller than the sum of SBL plus application SW, and iii) the data transfer on CAN (both ECUs are using a CAN bus with 500 Kbits/s) and the storage of the binary is close to three times faster on AURIX as the Volvo ECU first has to store and start the SBL and then load, store, and start the application SW.

2) Partial SW update evaluation: this experiment shows the benefits of a partial SW update compared to a traditional SW update. A SW update is often required due to a necessary bug fix or to update/adapt a parameter field of the ECU. In these cases, most parts of the SW remain unchanged and only some bytes have to be changed. To illustrate this, two different SW applications for AURIX were developed. The size of the .hex files of both versions is still 445 KB and they are utilizing a parameter field of 1024 byte in total. This parameter field is stored in a dedicated memory section of the AURIX ECU (see also Figure 3) and is the only difference between the SW versions. When utilizing partial SW updates, only the section containing the parameter field (1 KB) must be transferred to the ECU instead of transferring the entire binary.

In Table III the impact on the SW update duration is shown: a partial SW update is about six times faster compared to the normal update using an AURIX ECU (i.e., a duration decrease of 83%). This is mainly due to the fact that less data has to be transferred wirelessly from the DT to the WVI (a speed-up of 98%) as well as from the WVI to the ECU via CAN (about twenty times faster). The benefit of a partial SW update, however, strongly depends on the memory architecture of an

TABLE I
DURATION OF ALL REQUIRED STEPS IN MS OF A WIRELESS SW UPDATE W.R.T. SECURITY MECHANISMS ON NETWORK AND APPLICATION LAYER

Nw/Appl	Total	Discovery	Authentication	Init SW update	Seed&Key	Upload	Download	Validation
on/on	49193,3	3.2 (<0.1%)	2366,4 (4.8%)	1900,8 (3.9%)	509,8 (1.0%)	6585,4 (13.4%)	37315,3 (75.9%)	502,4 (1.0%)
off/on	47167,1	3.1 (<0.1%)	2360,4 (5.0%)	1992,0 (4.2%)	504,2 (1.1%)	5414,2 (11.5%)	36380,2 (77.1%)	505,1 (1.1%)
on/off	43745,6	3,2 (<0.1%)	1,88 (<0.1%)	1761,9 (4.0%)	510,7 (1.2%)	3756,0 (8.6%)	37200,2 (85.0%)	503,1 (1.2%)
off/off	41528,6	3.6 (<0.1%)	0,66 (<0.1%)	1764,67 (4.3%)	509,0 (1.2%)	2445,0 (5.9%)	36294,8 (87.4%)	502,7 (1.2%)

TABLE II
SW UPDATE DURATION: SW IS TRANSFERRED FROM A DT TO A WVI UTILIZING IEEE 802.11S AND THEN FORWARDED TO A ECU VIA CAN

Binary type	Binary size	On 11s	On CAN	Update duration
SBL	67KB	0.503s	6.268s	6.771s
Application	375 KB	2.527s	30.664s	33.191s

ECU as well as on the difference of two SW versions.

3) Parallel SW update evaluation: in the last evaluation step parallel SW updates are analyzed. These updates can be very beneficial in situations (see scenarios described in Section IV) where the same SW version shall be installed on ECUs integrated in several vehicles. This is of particular importance for the assembly line as well as for service centers when big vehicle recalls (e.g., due to a SW bug) are necessary, because the update can be performed on all vehicles simultaneously.

In the performed experiment two WVIs prototypes, each connected to an AURIX ECU, were wirelessly connected to the same DT. Instead of performing the SW update sequentially, the SW binary was installed on both ECUs in parallel.

The evaluation results presented in Table III show that the parallel update duration is increased by about 25% compared to a normal wireless SW update. This is due to the fact, that the required steps cannot by parallelized completely. However, parallel SW updates are still way faster (in this case 75% faster) compared to performing normal SW updates sequentially (i.e., performing a SW update two times in a row).

IX. CONCLUSION

In this paper SecUp, a generic framework enabling secure and efficient wireless SW updates is proposed. SecUp is designed and implemented to fulfill the requirements of several application scenarios: vehicle development, vehicle assembly line, vehicle maintenance, and OTA updates. SecUp encompasses efficient wireless SW update features such as parallel SW updates, where the SW of several vehicles is updated simultaneously, and partial SW updates, where only the changed parts of a SW binary are transferred and installed on an ECU. This paper also includes a description of the properties of the utilized IEEE 802.11s network and illustrates the designed cross-layer security concept used by SecUp.

A comparison of the developed SW update mechanisms show the benefits of utilizing advanced update features.

ACKNOWLEDGMENT

This work was partially funded by the SCOTT project, which received funding from the ECSEL Joint Undertaking under grant agreement No 737422. This joint undertaking is supported by EUs Horizon 2020 research and innovation program and Austria. SCOTT is also funded by the Austrian Federal Ministry BMVIT under the program "ICT of the Future" (05-2017 - 04-2020). The authors also acknowledge financial support by the COMET K2 Program of the BMVIT, BMWFW, FFG, the Province of Styria, and SFG.

REFERENCES

[1] Redbend Software, "Updating Car ECUs Over-The-Air (FOTA)," White Paper, pp. 1–14, 2011.
[2] M. S. Idrees, H. Schweppe, Y. Roudier, M. Wolf, D. Scheuermann, and O. Henniger, "Secure automotive on-board protocols: A case of over-the-air firmware updates," Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 6596 LNCS, pp. 224–238, 2011.
[3] D. K. Nilsson and U. E. Larson, "Secure firmware updates over the air in intelligent vehicles," IEEE International Conference on Communications, pp. 380–384, 2008.
[4] D. Nilsson, P. Phung, and U. E. Larson, "Vehicle ECU classification based on safety-security characteristics," Road Transport Information and Control - RTIC, pp. 1–7, 2008.
[5] K. H. Peter Subke and V. Marquart, "ODX-based flash solution," CAN Newsletter, no. 3, pp. 42–46, 2015.
[6] M. Khurram, H. Kumar, A. Chandak, V. Sarwade, N. Arora, and T. Quach, "Enhancing connected car adoption: Security and over the air update framework," pp. 194–198, Dec 2016.
[7] S. M. Mahmud, S. Shanker, and I. Hossain, "Secure software upload in an intelligent vehicle via wireless communication links," in Intelligent Vehicles Symposium. Proceedings. IEEE, June 2005, pp. 588–593.
[8] I. Hossain and S. M. Mahmud, "Analysis of a secure software upload technique in advanced vehicles using wireless links," in Intelligent Transportation Systems Conference, 2007. ITSC 2007. IEEE. IEEE, 2007, pp. 1010–1015.
[9] R. Petri, M. Springer, D. Zelle, I. McDonald, A. Fuchs, and C. Krauss, "Evaluation of lightweight tpms for automotive sw updates over the air," in Embedded Security in Cars Conference. escar, 2016, p. 15.
[10] N. Gabe, "Over-the-air updates on varied paths," Automotive News, 2016-01-25.
[11] D. T. Tuan, S. Sakata, and N. Komuro, "Priority and admission control for assuring quality of I2V emergency services in VANETs integrated with Wireless LAN Mesh Networks," ICCE 2012, pp. 91–96, 2012.
[12] S. Chakraborty and S. Nandi, "IEEE 802.11s mesh backbone for vehicular communication: Fairness and throughput," IEEE Transactions on Vehicular Technology, vol. 62, no. 5, pp. 2193–2203, 2013.
[13] M. Steger, M. Karner, J. Hillebrand, W. Rom, E. Armengaud, M. Hansson, C. A. Boano, and K. Roemer, "Applicability of IEEE 802.11s for automotive wireless software updates," in 13th International Conference on Telecommunications (ConTEL), July 2015, pp. 1–8.
[14] W. K. Tan, S.-G. Lee, J. H. Lam, and S.-M. Yoo, "A security analysis of the 802.11s wireless mesh network routing protocol and its secure routing protocols," Sensors, vol. 13, no. 9, p. 11553, 2013.
[15] D. Harkins, "Simultaneous authentication of equals: A secure, password-based key exchange for mesh networks," in Second International Conference on Sensor Technologies and Applications. SENSORCOMM '08., Aug 2008, pp. 839–844.
[16] M. Sbeiti, A. Wolff, C. Wietfeld, and I. Technology, "PASER: Position Aware Secure and Efficient Route Discovery Protocol for Wireless Mesh Networks," in International Conference on Emerging Security Information, Systems and Technologies - SECURWARE, 2011.
[17] M. Sbeiti and C. Wietfeld, "One stone two birds: On the security and routing in wireless mesh networks," in Wireless Communications and Networking Conference (WCNC), 2014 IEEE, April 2014.
[18] M. S. Islam, M. A. Hamid, and C. S. Hong, "SHWMP: A Secure Hybrid Wireless Mesh Protocol for IEEE 802.11s Wireless Mesh Networks," in Transactions on Computational Science VI. Springer Berlin Heidelberg, 2009, pp. 95–114.

TABLE III
DURATION OF ALL REQUIRED STEPS IN MS OF A WIRELESS SW UPDATE. COMPARISON BETWEEN THE VOLVO ECU AND THE AURIX MODES.

ECU/mode	Total	Connection-related	ECU-related	Upload	Download
Volvo/normal	48681.0	2369.7 (4.9%)	2410.6 (5.0%)	6585.4 (13.5%)	37315.3 (76.7%)
AURIX/normal	20768.8	2340.2 (11.3%)	205.4 (1.0%)	4597.0 (22.1%)	13626.3 (65.6%)
AURIX/partial	3570.7	2351.7 (65.9%)	216.7 (6.1%)	347.1 (9.7%)	655.2 (18.3%)
AURIX/parallel	25881.8	3868.3 (14.9%)	262.9 (1.0%)	7378.5 (28.5%)	14372.2 (55.5%)

[19] J. Ben-Othman and Y. I. Saavedra Benitez, "IBC-HWMP: a novel secure identity-based cryptography-based scheme for Hybrid Wireless Mesh Protocol for IEEE 802.11s," Concurrency and Computation: Practice and Experience, vol. 25, no. 5, pp. 686–700, 2013.

[20] M. Steger, M. Karner, J. Hillebrand, W. Rom, and K. Roemer, "A Security Metric for Structured Security Analysis of Cyber-Physical Systems Supporting SAE J3061," pp. 1–8, 2016.

[21] SAE, "SAE J3061: Surface Vehicle Recommended Practive - Cybersecurity Guidebook for Cyber-Physical Vehicle Systems," SAE International, Tech. Rep., 2016.

[22] S. Brown and C. J. Sreenan, "Software updating in wireless sensor networks: A survey and lacunae," Journal of Sensor and Actuator Networks, vol. 2, no. 4, pp. 717–760, 2013.

[23] M. L. Chiang and T. L. Lu, "Two-stage diff: An efficient dynamic software update mechanism for wireless sensor networks," in 2011 IFIP 9th International Conference on Embedded and Ubiquitous Computing, 2011, pp. 294–299.

[24] "open80211s – An open-source implementation of the recently ratified IEEE 802.11s wireless mesh standard," http://open80211s.org/open80211s/.

[25] ISO, "ISO 22901-1: Road vehicles – Open diagnostic data exchange (ODX) – Part 1: Data model specification," ISO, Tech. Rep., 2008.

[26] ——, "ISO 14229:2006: Road vehicles – Unified diagnostic services (UDS) – Specification and requirements," ISO, Tech. Rep., 2006.

[27] I. Garitano, S. Fayyad, and J. Noll, "Multi-metrics approach for security, privacy and dependability in embedded systems," Wirel. Pers. Commun., vol. 81, no. 4, pp. 1359–1376, Apr. 2015.

[28] B. Potter, "Microsoft SDL threat modelling tool," Network Security, pp. 15–18, 2009.

[29] M. Steger, C. Boano, M. Karner, J. Hillebrand, W. Rom, and K. Roemer, "SecUp: Secure and Efficient Wireless Software Updates for Vehicles," pp. 628–636, 2016.

[30] M. Steger, A. Dorri, S. S. Kanhere, K. Roemer, R. Jurdak, and M. Karner, Secure Wireless Automotive Software Updates Using Blockchains: A Proof of Concept. Cham: Springer International Publishing, 2018, pp. 137–149.

[31] ISO, "ISO 26262: Road vehicles – Functional safety – Part 1: Vocabulary," ISO, Tech. Rep., 2011.

[32] M. Steger, M. Karner, J. Hillebrand, W. Rom, C. Boano, and K. Roemer, "Generic framework enabling secure and efficient automotive wireless SW updates," pp. 1–8, Sept 2016.

Marco Steger is senior researcher at the VIRTUAL VEHICLE research center in Graz, Austria. He received a masters degree from Graz University of Technology in 2013 with a thesis titled "Development and Evaluation of C2X applications" done in cooperation with BMW, Munich, Germany. His research interests encompass wireless automotive communication networks, security in wireless/mobile/automotive networks, wireless sensor networks, and car-to-x applications.

Carlo Alberto Boano is an assistant professor at the Institute for Technical Informatics of Graz University of Technology, Austria. He received a doctoral degree sub-auspiciis praesidentis from Graz University of Technology in 2016 with a thesis on dependable WSNs, and holds a double Master degree from Politecnico di Torino, Italy, and KTH Stockholm, Sweden. His research interests encompass the design of dependable networked embedded systems and the robustness of IoT communication protocols against environmental influences.

Thomas Niedermayr received a masters degree from Graz University of Technology in 2017 with a thesis titled "Enabling Wireless Automotive SW Updates for the Infineon AURIX ECU". His research interests are focused on the development of Embedded and Automotive systems.

Kay Roemer is professor at and director of the Institute for Technical Informatics at Graz University of Technology. Before, he held positions of Professor at the University of Luebeck in Germany, and senior researcher at ETH Zuerich in Switzerland. His research interests encompass wireless networking, fundamental services, operating systems, programming models, dependability, and deployment methodology of networked embedded systems, in particular IoT, Cyber-Physical Systems, and sensor networks.

Karner Michael did his PhD in Electrical Engineering at Graz University of Technology. He is with VIRTUAL VEHICLE since 2011. As Lead Researcher and Project Manager, he is working in several large international industrial research projects. His research interests include wireless technologies, cyber-physical systems, functional and active safety - especially with focus on simulation/co-simulation based analysis and verification.

Joachim Hillebrand leads the Embedded Systems Group at VIRTUAL VEHICLE. Before joining VIRTUAL VEHICLE in 2009, he held research positions at NTT DoCoMo Eurolabs and BMW Research and Technology in Munich, Germany. His research interests include automotive E/E architectures as well as wireless and wired vehicle data communication.

Werner Rom is head of Integrated Vehicle Development at VIRTUAL VEHICLE and coordinator of large European research projects. As a PhD physicist he has nearly 20 years of experience in international project management in industry, research and the public sector, in Traffic Telematics, Electric mobility and in Automotive / Aerospace / Rail Technology.