Embedding Trust into Cars—Secure Software Delivery and Installation

André Adelsbach
Andre.Adelsbach@nds.rub.de

Ulrich Huber
huber@crypto.rub.de

Ahmad-Reza Sadeghi
sadeghi@crypto.rub.de

Christian Stüble
stueble@acm.org

Horst Görtz Institute for IT Security
Ruhr-Universität Bochum
Universitätsstraße 150
44780 Bochum, Germany

October 17, 2005

Abstract

Increasingly, software (SW) in vehicles can be updated due to the rising share of flashable electronic control units (ECUs). However, current SW installation procedures are insecure: An adversary can install SW in a given ECU without any sender authentication or compatibility assessment. In addition, SW is installed on an all-or-nothing basis: With the installation, the vehicle owner acquires full access rights to any functionality. Concepts for solving individual deficiencies of current installation procedures have been proposed, and the adoption of a complete solution considering all requirements is to be expected in the medium term.

In this article we summarize an existing protocol for secure SW delivery and installation in vehicles. This protocol respects the complex trust relations in the automotive industry and illustrates typically involved parties. However, we merely use it as an example for an arbitrary SW delivery protocol. Assuming SW delivery has already occurred, we give three exemplary design options for the vehicle's internal security architecture. Based on these options, an arbitrary installation protocol can be implemented. Specifically, we propose to use Trusted Computing technology in order to embed a trusted party into each vehicle. This party is implemented as a light-weight security ECU combining trusted hardware (HW) and SW. It assumes all security-relevant tasks related to SW delivery and thereby eases the workload of regular ECUs, which can rely on weaker security assumptions. Finally, we outline the management of flexible access rights to individual functionalities of the installed SW, thus enabling new business models.

1 Introduction

Control unit hardware (HW) and software (SW) in vehicles used to be tied together as a single product and rarely changed once the vehicle had been delivered to its owner. Nowadays, HW and SW in an ECU have become separate products. SW can be updated or upgraded during the vehicle's whole life-cycle and add customer value due to the ubiquitous use of flashable[1] ECUs. Examples are an optimization of the fuel injection parameters to increase engine performance or reduce emission levels, an upgrade of the navigation system and an update of the navigation data.

[1] A flashable ECU is a microcontroller capable of reprogramming its memory for application programs and data based on so-called flash memory technology [9].

Current procedures for installing SW in automotive ECUs are insecure; details about the deficiencies will be given in Section 2. Historically, these deficiencies didn't matter because SW installation was focused on warranty-based replacement of defective SW. The vehicle owner was informed of a recall and received the SW updates free of charge, e.g., when safety-relevant subsystems like airbags or the Electronic Stability Program (ESP) contained SW bugs. Recently, a paradigm shift has taken place: SW providers can distribute value-added SW components to interested owners and extract revenues even after delivery, e.g., annual fees for updated navigation data.

The secure delivery of SW to vehicles and the management of the corresponding digital rights differ from any existing DRM system known to the authors. First, the distribution currently necessitates a skilled intermediary between SW provider and user because the installation process requires specific skills and equipment. The SW update is carried out via a manufacturer-specific diagnostic tester that is only intended for maintenance service providers, e.g., dealers, garages and road service teams.[2] Second, different classes of such intermediaries exist: Depending on their skills and equipment, maintenance service providers usually have different installation rights. For example, an uncertified garage should not be granted the right to install SW for safety-relevant ECUs such as the airbag ECU. Third, compatibility is a major issue in the automotive industry. An SW component may be incompatible with the intended ECU or the SW components running in other ECUs of the vehicle [3, 19, 22, 17, 26]. The reasons for this incompatibility include the high number of ECUs—approximately 40 in an average compact-class vehicle—and the number of different car models, each with a continuously evolving ECU configuration. Last, new business models for automotive SW will induce new requirements. Due to the high value of the vehicle and the potential consequences of an accident, non-repudiation may become an important requirement. For example, if an honest car owner has an accident due to defective SW, his dealer and the SW provider may not be able to deny the installation of this SW.

We outline an existing protocol for secure SW delivery and installation in vehicles. This protocol is general and can be used for any embedded system. We will omit its details that can be found in [1, 2]. The protocol respects the complex trust relations in the automotive industry and illustrates typically involved parties. While it merely serves as an example for an arbitrary delivery and installation protocol, it motivates our assumption that such protocols need to fulfill security requirements and will become relevant for the automotive industry.

For the main contribution of this paper, we will assume that an SW component has already been securely delivered to the vehicle. Based on this assumption, we will give exemplary design options for a security architecture within the vehicle. We explain three such options that enable secure installation of SW components as well as a basic mechanism for enforcement of the SW providers' terms and conditions. The options visualize the inherent trade-offs between trust assumptions and flexibility of the implementation. The architectures' components aren't idealized black boxes, but can be implemented with existing technology under realistic assumptions.

We note that it is often assumed that cryptography alone can ensure security requirements such as confidentiality or authenticity. However, all cryptographic schemes rely on a strong assumption. The HW and SW in which these schemes are executed are assumed to be secure, e.g., tamper-resistant. Specifically, it is assumed that the keys used in cryptographic schemes cannot be extracted from the CPU or the memory in which they are processed. In addition, the SW component implementing a cryptographic scheme is assumed to be immune to attacks by malicious SW components running in parallel.

These assumption are unrealistic due to a variety of known HW and SW attacks; we will therefore tailor our design options to the prevention of such attacks while avoiding a cumbersome security architecture. Only a small part of the vehicle's HW and SW needs to be fully trusted, while other parts can be implemented with significantly lower trust assumptions. This has strong implications on the vehicle's costs: With lower trust assumptions, the HW components become less expensive.

[2] Although diagnostic testers are reported to have been cloned or stolen in some cases, the vast majority of SW updates is still carried out by maintenance service providers.

The design options use Trusted Computing technology in order to embed a small trusted party into each vehicle. This party is implemented as an ECU combining trusted HW and SW. It assumes all security-relevant tasks in the SW delivery phase, e.g., decryption or authentication, and thereby eases the workload of regular ECUs in the SW installation phase.

Section 2 summarizes related work and illustrates deficiencies of current SW installation protocols. Our overall system model is explained in Section 3, while a possible solution is highlighted in Section 4. In Section 5, we give our design options for the vehicle's internal security architecture and conclude in Section 6.

2 Related Work

A typical procedure for installing SW in an automotive ECU is described in [9]. It is performed by a so-called flashloader, a standard SW environment that allows for in-system re-programming of ECUs. After initialization of the installation mode, the flashloader erases the programmable memory of the ECU. Then it writes the new SW into the programmable memory. Finally, the procedure ends with the deinitialization of the installation mode.

Current installation procedures rarely apply any cryptographic techniques [9, 10, 19]. The use of signatures has been proposed, but not yet implemented [16, 19, 20]. However, the only signature mentioned in the proposals is that of the manufacturer.[3] If the manufacturer must sign every SW component prior to installation, he is capable of discriminating individual SW providers by refusing to sign their SW components.

We illustrate several other deficiencies with some examples. First, the intellectual property contained in the SW is not protected. As the SW is available in unencrypted form, reverse-engineering attacks may be possible. Second, the installation rights of the maintenance service providers are not verified in the course of an installation. Hence anyone with the necessary equipment—including an adversary—can install any (potentially malicious) SW component. Third, the owner cannot prove that he has legally acquired an SW component that has been installed in his vehicle. Even if the manufacturer applies a signature, the owner can still be accused of having acquired the SW illegally, e.g., without payment of license fees. Fourth, compatibility is not checked during installation. Even if signatures are used, they only prove the source of the SW, not compatibility. SW might be erroneously accepted by an incompatible vehicle due to the manufacturer's signature. Last, no rights management is currently applied. Techniques such as expiry dates or usage counters are not yet implemented, which prevents the introduction of more flexible business models. For example, those techniques would allow to sell additional horsepower or country-specific navigation data for a limited time frame or number of usages.

A framework for international automotive SW installation standards is introduced in [10]. However, it doesn't consider any DRM or security aspects. An infrastructure for installing SW from any external interface is proposed in [16]. Compatibility is ensured by checking if the hash values of all involved SW components form an SW release. An SW release is defined as an SW configuration which has been released by the vehicle manufacturer, while an SW configuration is defined as a valid and operational set of SW components and corresponding coding parameters which can be programmed in the ECUs of a vehicle. However, [16] doesn't consider any further security aspects. Requirements such as confidentiality, integrity, non-rejection and authenticity are mentioned, but not considered in the proposed architecture and left open to the specific implementation of each vehicle manufacturer. Several other research papers introduce the concept of distributing SW to vehicles in the field [6, 28], but even if security requirements are mentioned, no specific proposal to fulfill them is mentioned.

A proposal for "end-to-end security" of SW installation in vehicles is made in [20]. However, the signing of the SW component by "an authorized party" is the only protective measure, which provides only a partial solution to the requirements that we will introduce in Section 3.3. For

[3] The proposals generally do not specify whether they refer to the manufacturer of the vehicle or that of the relevant ECU.

example, it does not prevent discrimination of independent SW providers as the vehicle manufacturer is assumed to take over the role of the authorized party. Another proposal for secure SW installation is made in [19]: It contains an authentication phase, in which the diagnostic tester is authenticated, as well as an installation routine, which verifies checksums or signatures of the SW provider. Again, only some of the requirements are fulfilled. For example, signatures on the receiving end are omitted. Therefore the proposal does not prevent repudiation of a successful installation by the vehicle owner.

3 Model

3.1 Roles

The following roles (see also Fig. 1) will be used throughout the remainder of this article: The Overall Equipment Manufacturer (**OEM**) O develops, assembles and delivers the vehicles. O cooperates with suppliers that develop and/or manufacture vehicle components. The initial SW components at production time may be either from O or from O's suppliers. Examples are car manufacturers such as Daimler Chrysler, Ford, GM or Toyota.

An SW Application Programmer (**SAP**) S develops SW components for the vehicle. S may either be (i) a supplier that participates in developing and/or assembling the vehicle or (ii) an independent application programmer that develops and distributes SW components. Automotive examples are suppliers such as Bosch, Delphi, Denso, Siemens and Visteon. Henceforth, we use the term **SW provider** as a synonym for "OEM or any SAP".

A Maintenance Service Provider (**MSP**) M maintains the vehicle, i.e., mechanical parts, ECU HW and SW. As part of the maintenance service, M installs updates and/or upgrades of SW components. M has the equipment and capabilities that are necessary for the installation procedure. Automotive examples are car dealers, garages and road service teams. Each M has individual installation rights, which may be modeled as clearance levels [1, 2].

The License Provider (**LP**) L distributes licenses for SW components that the SW providers O and S have developed. Prior to license distribution, L establishes terms and conditions with the SW provider, detailing the model for sharing license revenues. To the authors' knowledge, automotive examples don't exist yet, but might be established as spin-offs of OEMs and/or SAPs.

The **vehicle** V is purchased by the vehicle owner. The owner is interested in SW for V and willing to pay for it in case of a perceivable value-added. We define V's **configuration** as the collective information on each SW (and implicitly HW) component that is installed in V.

The Trusted Third Party (**TTP**) T has two different certification tasks: First, T creates SW certificates for O and S. These certificates confirm the properties of each newly developed SW component. By **SW properties** we mean characteristic features such as functionality, interfaces, supported protocols, memory requirements and necessary environment. Second, T creates installation right certificates which certify M's right to install specific SW components. In the automotive example, this role is currently taken over by O. This implies a trust model in which S must trust O. However, an independent T becomes necessary if O is not fully trusted and discrimination should be avoided. T might evolve out of safety standards authorities such as the NHTSA[4] in the USA or the TÜV and Euro NCAP[5] in Europe.

3.2 Assumptions

3.2.1 Trust Relations

All honest parties are assumed to keep their secrets private, e.g., their cryptographic keys. There are no specific trust assumptions for O, S and M. All SW providers trust L to adhere to their terms and conditions. Finally, all parties trusts T. For example, this includes correct certification of SW properties and clearance levels.

[4] National Highway Traffic Safety Administration, `http://www.nhtsa.dot.gov/`

[5] URLS: `http://www.tuev-sued.de/` and `http://www.euroncap.com/`

3.2.2 Internal Architecture of the Vehicle

We assume each vehicle to have the internal architecture that is shown on the right-hand side of Fig. 1. There are several components $v_1, \ldots, v_n$ that represent the multitude of ECUs in a vehicle. The components are interconnected via an internal communication network, representing a vehicle's data busses such as CAN, LIN and MOST. In addition, there is one extra component v_0 that handles the SW delivery protocol between M and V. v_0 may either represent an additional ECU or an existing ECU integrating the functionality of v_0 and existing SW components.

3.2.3 Communication channels

We assume the existence of the communication channels represented by arrows in Fig. 1. All of them are assumed to be **integer**, i.e., free of bit errors. In addition, all but two channels are assumed to be **secure**, i.e., authentic and confidential. The first exception is the one-way broadcast channel between O and M, which is neither authentic nor confidential. However, it is **non-discriminatory**: (i) all SW providers can send over the channel and (ii) the channel has global reach, i.e., each M can receive. The second exception is V's internal communication network. Due to cost constraints on V's components, it is only assumed to be integer and reliable. By **reliable** we mean that each message sent reaches its recipient after a limited amount of time. Finally, the channels between L and M as well as between M and V are assumed to be reliable.

3.3 Security Requirements

The solution proposed in [1, 2] fulfills a variety of security requirements. A detailed description of these requirements is beyond the scope of this paper. However, we summarize them to motivate the need for TC technologies within the vehicle. We will group the requirements into two categories: (i) straight-forward requirements, which are state-of-the-art for current or planned installation procedures of the automotive industry and (ii) new requirements, that—to the authors' knowledge—haven't been considered yet.

The straightforward requirements include **correctness**, i.e., the guarantee of a successful installation in case of an execution of the installation protocol according to specification, **confidentiality**, i.e., protection of the SW from potential eavesdroppers, **integrity**, i.e., non-modification of the SW during delivery and installation, and **authenticity**, i.e., an unambiguous source of the non-modified SW. In addition, **rights enforcement** ensures that the SW providers' terms and conditions cannot be circumvented. **Compatibility enforcement** avoids failure of the vehicle or its components due to incompatibility of individual components. Finally, **clearance enforcement** ensures that only a qualified MSP is allowed to install SW, especially for safety-relevant subsystems such as the airbags or the ESP.
New requirements are non-repudiation, frame-proofness and non-discrimination. **Non-repudiation** and **frame-proofness** ensure that M and V can prove the result of the installation to any honest party. Neither of them can claim that the installation failed when it succeeded and vice versa. **Non-discrimination** prevents unjustified exclusion of individual SAPs and MSPs and therefore ensures competition. As long as an SAP S proves to the TTP that his SW is correct, e.g., successfully tested and compatible, S may distribute this SW and compete with other SAPs. The same holds for MSPs that have the necessary qualification.[6]

We stress that even the straightforward requirements such as confidentiality and authenticity make it practically unavoidable to use cryptographic schemes in an arbitrary SW delivery and installation protocol. Therefore, our proposed use of TC technology and the embedding of a trusted party into each vehicle is beneficial not only to the installation procedure of [1, 2], but to any installation procedure that fulfills the straightforward requirements.

[6] Non-discrimination is even embodied in law in many countries. For example, the European Commission Regulation 1400/2002 prevents discrimination of independent MSPs. The OEM must give them access to necessary material and technical information, e.g., spare parts and diagnostic equipment.

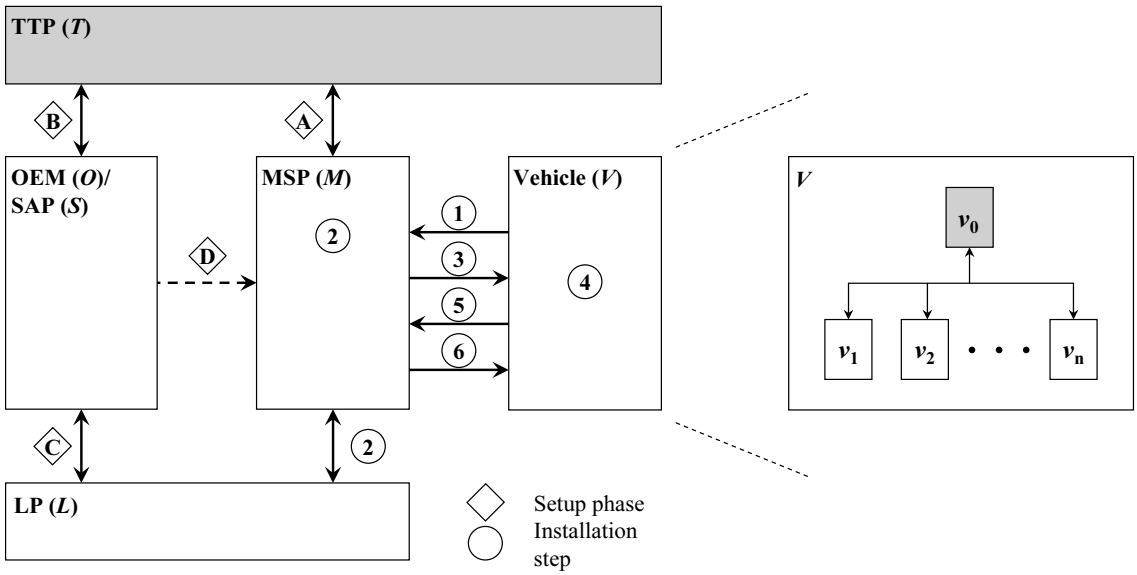

Figure 1: Installation procedure in six steps (left) and internal vehicle architecture (right)

4 Exemplary Solution

4.1 Overview

This section provides a summary of the SW delivery and installation protocol of [1, 2]. The protocol consists of a setup period (Phases A–D) and the actual installation (Steps 1–6) (see Fig. 1). In the setup period, the system parameters, e.g., security parameters of the cryptographic schemes, are chosen. Each M applies for a specific clearance level and is certified by T (Phase A). This certification is performed once and repeated only if a new M joins the system or existing certificates expire. In parallel, an SW provider who has developed a new SW component submits it to T and requests certification of the SW properties (Phase B). After certification, the SW provider establishes terms and conditions with L (Phase C). Although this needs to be done for each new component, an SW provider and L might establish more general terms and conditions covering a whole set of SW components. Finally, the SW component is distributed to each M via the broadcast channel (Phase D). The use of Public Key Broadcast Encryption (PKBE) ensures confidentiality *and* efficiency, e.g., short ciphertexts and little cryptographic key material [11].[7]

The actual installation of SW s starts with a request from V to M (Step 1). M then obtains a license l from L (Step 2). After delivery of s and l to V (Step 3), the component v_0 of V checks if s, l and M are legal (Step 4). If so, v_0 instructs the target component v_i to install s. v_0 then confirms the successful installation to M (Step 5) and awaits M's acknowledgement (Step 6). After receiving the acknowledgement, v_0 instructs v_i to use s. In addition, v_0 continuously monitors all licenses that V obtained and instructs v_i to stop using s (or otherwise adapt the parameters of s) when l expires.

4.2 Security of the Exemplary Solution

All of the requirements defined in Section 3.3 are fulfilled by the protocol of [1, 2]. Due to lack of space, we refer to the security analysis in the full version of the protocol [2].

[7] PKBE in [11] uses a Hierarchical Identity-Based Encryption (HIBE) scheme and cites [18, 13] as examples. However, the currently most efficient HIBE scheme is [7], which was published after [11].

5 Implementation

Our proposed method for finding an implementable vehicle architecture employs two basic ideas. The first idea is to ease the workload of the components $v_1, \ldots, v_n$ by adding a component v_0 that performs all computationally expensive operations in the SW delivery protocol. For example, if the non-repudiation requirement of Section 3.3 happens to be fulfilled using digital signatures, then these signatures are generated and verified by v_0. The aim of adding v_0 is to achieve a secure SW delivery and installation protocol with small modifications to the existing components $v_1, \ldots, v_n$ of a vehicle with respect to their tamper resistance and required processing power. We have already assumed the existence of v_0 in our model assumptions of Section 3.

The second idea is to find an implementation-specific compromise between tamper resistance assumptions on v_0 and the cost of implementing v_0. The trivial solution is to assume all components of v_0 to be tamper-resistant; however, this assumption comes at the price of expensive HW components. Another solution is to assume only some components of v_0 to be tamper-resistant, but extend the security they provide to the remaining components by using cryptographic techniques and trusted computing methodologies.

5.1 Requirements on $v_1, \ldots, v_n$

Ideally, the existing ECUs of a vehicle, corresponding to the components $v_1, \ldots, v_n$, wouldn't need to be modified at all. However, the security requirements of Section 3.3 cannot be fulfilled without any assumptions on $v_1, \ldots, v_n$. To fulfill these assumptions, the corresponding ECUs will need to be modified. The reason is straightforward: Existing ECUs can be flashed, allowing an adversary to install malicious code. For example, even if v_0 securely stores an SW component s after delivery, the adversary may obtain s through the target component v_i in which s is to be installed. He simply flashes the corresponding ECU and inserts code that outputs s in unencrypted form over the data bus.

As we need to be make modifications in any case, the next natural question is how to make them as small and cost-efficient as possible. The computational resources of existing ECUs, especially related to cryptographic techniques, are limited. For example, it would be highly unrealistic to expect RSA-based signature schemes in every ECU of a vehicle. In an attempt to make small modifications, we require the components $v_1, \ldots, v_n$ only to share a secret with v_0 that allows (i) to secure the channel between v_0 and v_i and (ii) to avoid unauthorized flashing. For example, in the protocol of [1, 2] each v_i receives a secret symmetric key k_i from its manufacturer at production time to allow symmetric encryption and authentication. v_0 may receive k_i by means of a certificate from v_i's manufacturer in which k_i is encrypted with a public key of v_0. Subsequently, v_0 sends SW to v_i only in encrypted form and authenticates each message to v_i with k_i. This can be done with symmetric encryption and message authentication schemes that are computationally less expensive than asymmetric schemes.

The assumption on v_i is simply a secret shared with v_0, but the important question is how to securely implement this idea in HW. A possible solution is the following: The manufacturer of the ECU corresponding to v_i embeds the shared secret directly into a small tamper-resistant area of the CPU. When the vehicle is assembled, the OEM ensures that v_i receives only trustworthy SW components. Subsequently, v_i protects its integrity by only accepting SW components that were encrypted and authenticated by v_0. To avoid a flashing attack, only v_0 may deliver the flashloader used to flash v_i. Note that the flashloader is not an integral part of the ECU, but erased after each flash procedure due to safety reasons. By having v_0 as the only source of the flashloader, flashing attacks are avoided as long as the trust assumptions on v_0 hold. For example, the final flashing policy may be that v_0, which is capable of asymmetric cryptography, only uses a flashloader that was certified and signed by the TTP.

Note that depending on the commercial value of the SW they contain, the ECUs corresponding to the v_i may receive different protection measures. Some of them may be protected as described in the previous paragraph. Others may receive no protection at all because their SW components are considered to have a lower value than the additional cost for protection measures. Yet others

may even receive tamper-resistant HW because the high value of the SW they contain makes it probable that an attacker would even try HW attacks. For example, the motor control unit may obtain higher protection to avoid chip tuning via HW attacks.

5.2 Requirements on v_0

Based on the protocol overview of Section 4.1 and the requirements on $v_1, \ldots, v_n$ in the previous section, we summarize the requirements on v_0. It serves as the Trusted Computing Base (TCB) of the vehicle. The requirements are generic in the sense that any other secure installation protocol with a central component v_0 will have similar requirements.

Cryptographic functionality: v_0 implements the cryptographic schemes used during SW delivery. Depending on the specific delivery protocol, this includes encryption and message authentication schemes.

Secure storage: v_0 needs to ensure confidentiality and integrity of (i) the keys of all cryptographic schemes used for SW delivery, (ii) the secrets shared between v_0 and the v_i and (iii) the SW licenses.

Verifiable boot: When v_0 is rebooted after a system shutdown, e.g., due to malicious interruption of power supply, unauthorized code may not obtain access to the securely stored data, i.e., keys, secrets and licenses.

Isolation: v_0 needs to be isolated from other components to prevent malicious code from obtaining the securely stored data via shared resources.

Functional requirements: The functional requirements include compatibility assessment, rights enforcement and, if applicable, the choice of a suitable target ECU for an SW update. **Compatibility assessment** needs to consider the current components v_i and the SW component s to be installed. Based on their properties, v_0 needs to decide whether V and s are compatible. We propose a possible implementation of this functionality in Section 5.5.1.

For **rights enforcement**, v_0 needs to continuously monitor the rights granted to V. Following [1, 2], we propose to grant these rights using licenses that V acquires during the delivery of SW s. For details, we refer to Section 5.5.2.

For **determination of the installation target**, v_0 needs to find the most appropriate target component v_i in which to install the SW s. The choice depends on the vehicle properties, e.g., HW components and available memory, and the properties of s, e.g., necessary run-time environment and memory space. We give further details in Section 5.5.3.

5.3 Assumptions on Available HW and SW components

For the implementation of v_0, we assume the following components to be available:
1. Tamper-resistant memory, CPU and communication channels between them. Such components are obviously more expensive than insecure components. Therefore, we will try to minimize the need for them, especially for tamper-resistant memory.
2. A secure operating system (OS) providing isolation of the processes that it executes with respect to code and data. A secure OS can be based on microkernel architectures—even in embedded systems, e.g., using PERSEUS [23]. For a discussion of microkernel architectures, we refer to [24].
3. Insecure mass memory such as hard-disks or RAM chips. Due to their lower cost per byte of memory, we prefer to use such memory, especially when memory requirements are high.
4. A Trusted Module (TM) with security functionalities that we will define in the sequel.

An example for a TM is the Trusted Platform Module (TPM) of the Trusted Computing Group[8] (TCG). It provides the following four security functionalities: attestation, tamper-resistant

[8] URL: `https://www.trustedcomputinggroup.org/`.

storage, sealing and random number generation [31, 30, 29]. The basic idea of *attestation* is to assess the current system configuration by observing the boot sequence from boot loader to application SW. An alternative to the current attestation mechanism of the TCG is property-based attestation as described in [25] and [14, 15]. *Sealing* ensures that, when an adversary changes the system configuration, e.g., by booting a malicious OS, the TPM refuses to use any cryptographic secrets that were defined to belong to the original configuration. This is achieved by comparing the current boot sequence with a reference boot sequence that is securely stored in the TPM in compressed form. Using sealing, confidential data can be stored in insecure memory. The TPM encrypts the data with a sealed secret and stores the encrypted data in insecure memory without comprising confidentiality.

Remark 1 Note that perfect tamper-resistance is considered to be impossible [4]. In the constant competition between designers of secure HW systems and hackers, tamper-resistance has always been temporary. With sufficient resources, e.g., time, money and expertise, a determined hacker may eventually break any HW. However, we stress that for most practical applications, sufficiently tamper-resistant HW can be developed. This means that an ordinary user and even a realistic adversary with limited resources cannot break the tamper-resistance of this HW. □

5.4 Design Options for Implementation of v_0

When the TCB v_0 is implemented, we find a multitude of different architectures comprising HW and SW components. It is of course desirable to determine the optimal architecture. However, this is impossible without additional implementation-specific information such as the cost per byte of memory, the storage requirements and the cost of a TM. As an analytical approach fails due to the lack of this information, we qualitatively discuss three different architectures that all implement v_0 and fulfill the requirements of Section 5.2. However, they differ in their tamper resistance assumptions and the shared use of HW components.

The trivial option is to exclusively rely on the components of type 1, i.e., tamper-resistant memory, CPU and communication channels. Although this option avoids the need for secure OS and TM, it has other inconveniences such as a high cost per byte of memory. Another option uses a secure OS and a TM to incorporate an insecure mass memory. This increases flexibility, but also incurs additional costs for OS and TM. This investment only pays off if the savings in cost per byte of memory offset the cost of TM and OS. To illustrate these inherent tradeoffs, we discuss the following three options: (i) an independent tamper-resistant ECU, exclusively implementing v_0, (ii) a central tamper-resistant ECU hosting v_0 *and* other SW components, using only secure memory components and (iii) a central ECU for v_0 and other SW components, using secure *and* insecure memory components. The security of options (i) and (ii) relies on their closed-ness, while option (iii) is a fully open system.

5.4.1 Independent Tamper-resistant ECU

This is the least complex, but also the most expensive option with respect to the cost per byte of memory and the relative cost of the CPU. The functionality of v_0 is isolated in a single security ECU that is implemented with (i) tamper-resistant HW according to item 1 of Section 5.3 and (ii) SW certified by a TTP T. Specifically, each party that assumes one of the roles in our model (see Section 3.1) may derive the trust in v_0 from an independent evaluation by a TTP, e.g., according to the Common Criteria [8] or other security criteria. In short, v_0 is a typical example of a closed subsystem that resides in a virtual safe preventing all attacks (see Fig. 2). Physically, this ECU should reside in an individual casing that could also be mechanically sealed to indicate intrusion. This is similar to the trust assumptions on consumer electronics devices such as pay-TV decoders. Note that the isolation and verifiable boot requirements of Section 5.2 are trivially fulfilled by implementing v_0 as a closed (sub-)system.

As v_0 is trusted by all parties and might be maliciously modified in a flash process, only T may perform updates of SW in v_0; modification of this SW needs to be inaccessible to all parties

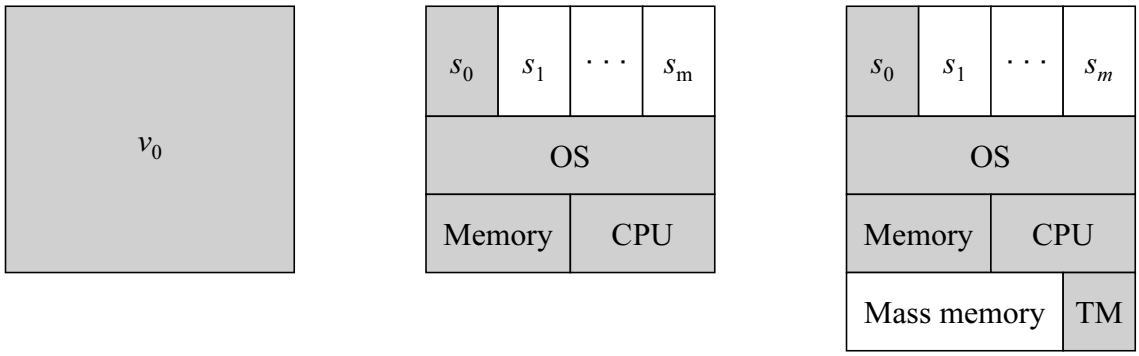

Figure 2: Security architecture for options 5.4.1 (left), 5.4.2 (middle) and 5.4.3 (right)

except T. Although possible, an update seems rather impractical as M cannot execute the update like in the usual SW update protocol. If M should nevertheless execute the update, then it may flash v_0 only partially while a dedicated self-update part of v_0 must remain in memory and ensure that the update originated from T.

The advantage of option 5.4.1 is that it isolates the SW delivery and installation functionality such that the corresponding ECU may be jointly developed and certified by several OEMs, leading to an industry standard and significant economies of scale that reduce the production costs of v_0. The inconvenience is that v_0 is a single-purpose ECU that prevents the parallel execution of any other SW component in the same ECU. The cost of v_0's HW is only attributable to v_0 and cannot be "shared" with other SW components using the same CPU.

5.4.2 Central Tamper-resistant ECU Using Secure Storage

This option is more flexible and complex, but possibly restricted in the available memory space. The idea is to have s_0 share its ECU with other SW components $s_1, \ldots, s_m$, where s_0 is the SW component of v_0. For example, v_0 may be implemented in the car radio or the dashboard ECU. In addition to tamper-resistant memory and processor (item 1 of Section 5.3), this option necessitates a secure OS (item 2 of Section 5.3) that isolates s_0 from the s_i. The trusted components of this architecture are shaded in grey. As the memory and the CPU are assumed to be tamper-resistant, the interaction between s_0 and these components is secure. For example, s_0 may use them to store the keys of cryptographic schemes and execute their algorithms. Again, the trust in s_0, OS, memory and CPU is derived from an independent evaluation by a TTP. In summary, the grey-shaded components of the architecture in Fig. 2 are still considered to be a closed subsystem, i.e., inaccessible to the attacker. Note that depending on the security requirements on $s_1, \ldots, s_m$, these SW components may also profit from an OS providing isolation.

It is instructive to compare how the requirements on the TCB v_0 are fulfilled in this option. The cryptographic schemes are implemented in SW certified by a TTP. Secure storage is provided by the tamper-resistant memory which is protected from the other SW components through isolation. Isolation originates from the secure OS. To achieve a verifiable boot, an adversary may not succeed in changing the ECU's memory and restarting the ECU with a different OS. This is achieved with the tamper resistance of CPU and memory.

An update of s_0 is possible, but has the same restrictions as the option of Section 5.4.1. However, there is an additional alternative: The OS can contain a secure and dedicated installation procedure for s_0, which is separate from the regular installation procedure for SW components. As the OS is trusted and doesn't change during an update of s_0, it can ensure that the update was certified by the TTP.

The advantage of this option is that it allows to execute several SW components in parallel on a single ECU. The cost of the ECU is shared among these SW components. The HW and the OS of v_0 might be jointly developed and certified by several OEMs, leading to an industry standard

and economies of scale. However, compared to Section 5.4.1 this seems less likely and feasible as the standard would have to cover the requirements of all participating OEMs regarding the s_i—a compromise that is more difficult to achieve than for a single-purpose component v_0. The inconvenience of this option is that it exclusively relies on tamper-resistant memory. This may be acceptable if the OS and the SW components are small. Otherwise, the next option is more appropriate.

5.4.3 Central Tamper-resistant ECU Using Partially Insecure Storage

This option is the most flexible and even allows execution of components v_i with large memory requirements. For example, navigation data, music or video files can be securely stored using inexpensive mass memory. The basic idea is to add insecure mass memory, but maintain security through the use of a TM that at least enables attestation and sealing. Note that we do not require all functionalities of the TPM. As shown in Fig. 2, mass memory and TM are the only additional components compared to the previous option.

In this option, the requirements on the TCB v_0 are fulfilled as follows: The cryptographic schemes are again implemented in SW certified by a TTP; if a scheme happens to be available in the TM, it may of course also be executed there. Secure storage is achieved by either using the small tamper-resistant memory part or by storing in encrypted form on the insecure mass memory. The TM helps to verify that the overall ECU is in a trustworthy configuration before decryption keys are released. Specifically, a verifiable boot is achieved using the sealing mechanism of the TM. Isolation is provided by the secure OS as in the previous option.

As the TM can assess the ECU configuration during every boot sequence using the attestation mechanism, we can use this functionality to make the ECU an *open* system. All SW components and even the OS can be updated as long as the update is accompanied by an SW certificate of a TTP, confirming its security. When an adversary changes the mass memory, e.g., to insert an insecure OS, and causes the ECU to reboot, e.g., by temporarily interrupting the power supply, the stored secrets remain locked as the TM detects the changed configuration.

The advantage of this option is that it not only allows to execute several SW components in parallel, but also uses cost-efficient mass memory in a secure way and implements an open system. Again, the cost of the ECU is shared among the SW components. Even if it is unlikely that the whole ECU will become an industry standard, individual components such as the TM and the secure OS may be jointly developed and certified, allowing a flexible use of all other related HW components. The inconvenience is that the investment in the TM will only pay off if the memory requirements are high. When the memory requirements increase, there is a threshold where the reduced cost per byte of memory compensates the additional cost for the TM.

5.5 Details on Functional Requirements

To show the practical relevance of our proposed design guidelines and the exemplary security architectures, we show how they can fulfill the functional requirements of Section 5.2.

5.5.1 Details on Compatibility Assessment

Current SW installation procedures in the automotive industry rely on a description of allowed SW configurations to determine compatibility [16]. The description of an allowed configuration is a logical expression of the following form: "SW s_1 in component v_1 `and` SW s_2 in component v_2 `and` ... `and` SW s_n in component v_n". To give an artificial, but illustrative example, an SW configuration may be version 1.4 of the Adaptive Cruise Control (ACC) SW, version 1.7 of the engine control SW and version 2.3 of the ABS SW. However, upward/downward compatibility is not necessarily given. An SW may work fine with ACC version 1.4, but not with 1.2. When versions change frequently, the list of allowed SW configurations can become very long—there can easily be 40 ECUs in a vehicle. The list needs to be delivered as fresh information at installation

time; whenever a new SW component is developed, the compatibility information needs to be updated.

If these inconveniences are acceptable, the compatibility assessment may be based on SW configurations and implemented as follows: Together with each SW component, v_0 receives a list of allowed SW configurations. For example, an SW provider may experimentally establish this list in a testing series and ask the TTP to certify it. v_0 stores the current version of each SW component within V, with an initial list generated at production time. The compatibility assessment function of v_0 simply evaluates compatibility by comparing the version of the SW component s to be installed and the current versions of the other SW components with the allowed configurations in the list. If a match is found, v_0 outputs "compatible", otherwise "incompatible". v_0 refreshes the version list by updating the version of s.

There is a more efficient alternative to this approach. Suppose that all involved parties agree on SW properties in a standardized way. For example, the engine control SW provides a property "controllability of engine speed" and the brake control SW a property "controllability of deceleration rate". It is unimportant which SW version provides these properties as long as they are present. If properties are standardized, then it is sufficient for compatibility assessment to check whether the SW to be installed requires only properties that are already present in the vehicle. If the SW is an update that replaces an existing SW, it must at least provide the same properties as the SW that it replaces. To continue the ACC example, when installing this SW the availability of the two cited properties is a prerequisite for compatibility.[9]

If this alternative approach is chosen, v_0 simply receives an initial list of V's properties at production time. Each entry in the list indicates the SW component that provides this property. At each installation, the SW s is delivered together with a list of necessary properties of V and a list of properties that s adds to V after installation. Prior to installation, v_0 verifies that (i) V has the properties demanded by s and (ii) s provides at least the properties of the SW it replaces. After the installation, v_0 refreshes the property list by incorporating the properties added by s. For example, when installing an arbitrary ACC SW, v_0 verifies that all necessary properties are present, e.g., the interfaces to engine control and brake control as described before. Afterwards, v_0 stores ACC as a new property that is now available for future updates.

Note that both approaches are similar in the sense that an ordered list containing information on the current SW configuration is stored and updated at each installation. However, the second approach reduces this list to the truly relevant information in an explicit form. In the first approach, relevant properties are derived from a version number. In the second approach, they become explicit and easily interpretable by all involved parties.

5.5.2 Details on Rights Enforcement

In [1, 2], the authors describe a light-weight approach for enforcing the terms and conditions related to an SW component s. This approach can be securely implemented in the TCB v_0. Based on the terms and conditions, the license provider L generates licenses containing the usage rights of vehicle V related to s. As these licenses need to be authentic, L signs them with the private key of a digital signature scheme. The TCB v_0 verifies these signatures and translates the rights granted in the license into a set of simple parameters, e.g., the value of a counter or an expiry date. Instead of the whole license, v_0 sends these simple parameters to the installation target v_i, authenticating them with the secret shared with v_i. Thereby the workload of v_i is considerably reduced to updating a parameter instead of interpreting a license.

5.5.3 Details on Determination of Installation Target

With current installation procedures, each SW s is intended for a specific type of ECU, often from a specific HW supplier. Therefore, s is delivered together with specific instructions on the target ECU. The SW provider and/or the OEM determines the target ECU. This variant of determining

[9] The ACC example is of course explanatory and not complete. ACC obviously needs other properties such as "measurability of distance to obstacles ahead".

the installation target can be implemented by delivering the target ECU as part of the properties of s. As the properties are authentic and certified by the TTP, the information on the target ECU is trustworthy.

AUTOSAR[10] is an initiative of the automotive industry aiming to establish standards for SW and electronics. The AUTOSAR Run-Time Environment (RTE) [5] provides abstraction from specific HW versions and standardization of interfaces between the RTE and SW components. Therefore, the target ECU may be determined at installation time and depend on the vehicle's current state. There may be several candidate ECUs for installing s. Possible criteria for selecting one of them are the available flash memory, optimization of the communication flow between ECUs and their physical location. If several SW components communicate with each other, they should preferably reside in the same ECU and use the internal communication channels of the ECU instead of the slower and less reliable channels between ECUs. Safety-relevant ECUs might need to be physically close to the mechanical parts that they operate, e.g., it might be preferable to have an airbag ECU close to the airbag for reasons of compartmentalization.

To implement this variant of determining the installation target, there are several options. To give an example, v_0 stores the available memory of each ECU in a list. In addition, it stores the physical location of each ECU in a separate list. When a new SW s is to be installed, v_0 can calculate a weighted average of the aforementioned criteria, i.e., available flash memory, communication flow and physical location, for all candidate components v_i and base its decision on the resulting averages. The SW provider might even deliver s with individually selected coefficients for the weighted average.

6 Conclusion

In this article we describe a protocol for secure SW delivery and installation in vehicles and other embedded systems. The protocol serves as an example for an arbitrary secure protocol and shows one of many possible ways to securely deliver SW to a vehicle. Assuming that such delivery has already taken place, we give three design options for the architecture of a trusted computing base within the vehicle. The three design options illustrate the inherent trade-offs between trust assumptions and flexibility of the implementation. Although the architectures remain simple, requiring only a small part of the vehicle's HW and SW to be fully trusted, they allow secure SW installation and basic management of usage rights related to SW components. By embedding a single trusted ECU based on trusted computing technology into each vehicle, all other ECUs rely on significantly lower trust assumptions. The trusted ECU assumes all security-relevant tasks related to SW delivery and thereby eases the workload of the other ECUs.

References

[1] André Adelsbach, Ulrich Huber, and Ahmad-Reza Sadeghi, *Secure software delivery and installation in embedded systems*, ISPEC 2005 (Robert H. Deng, ed.), Lecture Notes in Computer Science, vol. 3439, Springer, 2005, pp. 255–267.

[2] ———, *Secure software delivery and installation in embedded systems*, Technical Report, full version of [1], Horst Görtz Institute for IT Security, `http://www.prosec.rub.de/publications`, 2005.

[3] H. Alminger and O. Josefsson, *Software handling during the vehicle lifecycle*, in VDI Society for Automotive and Traffic Systems Technology [32], pp. 1047–1055.

[4] Ross J. Anderson, *Security engineering: A guide to building dependable distributed systems*, first ed., John Wiley & Sons, New York, USA, 2001.

[10] URL: `http://www.autosar.org/`

[5] AUTOSAR, *AUTomotive Open System ARchitecture development partnership information pack*, Informative material, URL `http://www.autosar.org/download/AUTOSAR_Standard_InfoPack_V3_6_f.pdf` - mailto: `request@autosar.org` - file size: 282 kB, January 21, 2003.

[6] BMW Car IT, *Das Potenzial von Software im Fahrzeug*, Press report, BMW Group, URL `http://www.bmw-carit.de/pdf/plakate.pdf` - mailto: `info@bmw-carit.de` - file size: 2050 kB, July 22, 2002.

[7] Dan Boneh, Xavier Boyen, and Eu-Jin Goh, *Hierarchical identity based encryption with constant size ciphertext*, EUROCRYPT 2005 (Ronald Cramer, ed.), Lecture Notes in Computer Science, vol. 3494, Springer, 2005, pp. 440–456.

[8] Common Criteria Project Sponsoring Organisations, *Common criteria for information technology security evaluation*, Norm Version 2.1, CCIMB-99-031 – 33, August 1999, URL `http://csrc.nist.gov/cc/CC-v2.1.html`.

[9] Daimler Chrysler AG, *Functional specification of a flash driver version 1.3*, Specification, Herstellerinitiative Software, URL `http://www.automotive-his.de/download/HIS%20flash%20driver%20v130.pdf` - mailto: his@mbtech-services.net - file size: 224 kB, June 06, 2002.

[10] Christoph Dallmayr and Oliver Schlüter, *ECU software development with diagnostics and flash down-loading according to international standards (SAE Technical Paper Series 2004-01-0273)*, in Society of Automotive Engineers (SAE) [27], URL `http://www.sae.org/`.

[11] Yevgeniy Dodis and Nelly Fazio, *Public key broadcast encryption for stateless receivers*, Digital Rights Management Workshop (Joan Feigenbaum, ed.), Lecture Notes in Computer Science, vol. 2696, Springer, 2003, pp. 61–80.

[12] *Jahrestagung Elektronik-Systeme im Automobil, Fachtag Design – Test – Diagnose elektronischer Systeme, Munich, Germany, February 12, 2004*, Euroforum, 2004.

[13] Craig Gentry and Alice Silverberg, *Hierarchical ID-based cryptography*, ASIACRYPT 2002 (Yuliang Zheng, ed.), Lecture Notes in Computer Science, vol. 2501, Springer, 2002, pp. 548–566.

[14] Vivek Haldar, Deepak Chandra, and Michael Franz, *Semantic remote attestation—a virtual machine directed approach to trusted computing*, Proceedings of the 3rd Virtual Machine Research and Technology Symposium (May 6–7, 2004, San Jose, CA, USA) (2004), 29–41, URL `http://www.usenix.org/events/vm04/tech/haldar/haldar.pdf`.

[15] Vivek Haldar and Michael Franz, *Symmetric behavior-based trust: A new paradigm for Internet computing*, in NSPW 2004 [21], 79–84.

[16] Cornelia Heinisch and Martin Simons, *Loading flashware from external interfaces such as CD-ROM or W-LAN and programming ECUs by an on-board SW-component (SAE Technical Paper Series 2004-01-0678)*, in Society of Automotive Engineers (SAE) [27], URL `http://www.sae.org/`.

[17] A. Heinrich, K. Müller, J. Fehrling, A. Paggel, and I. Schneider, *Version management for transparency and process reliability in the ECU development*, in VDI Society for Automotive and Traffic Systems Technology [32], pp. 219–230.

[18] Jeremy Horwitz and Ben Lynn, *Toward hierarchical identity-based encryption*, EUROCRYPT 2002 (Lars R. Knudsen, ed.), Lecture Notes in Computer Science, vol. 2332, Springer, 2002, pp. 466–481.

[19] M. Huber, T. Weber, and T. Miehling, *Standard software for in-vehicle flash reprogramming*, in VDI Society for Automotive and Traffic Systems Technology [32], URL `http://www.automotive-his.de/download/presentation-baden-baden-2003-german.zip`, pp. 1011–1020.

[20] Markus Müller, *IT-Security in Fahrzeugnetzen*, Elektronik Automotive (2004), no. 4, 54–59, ISSN: 1614-0125.

[21] *Proceedings of the 2004 New Security Paradigms Workshop, Nova Scotia, Canada, September 20–23, 2004*, ACM Press, 2005.

[22] Uwe Oeftiger, *Diagnose und Reparatur elektronisch unterstützter Fahrzeuge*, in Euroforum 2004 [12].

[23] B. Pfitzmann, J. Riordan, C. Stüble, M. Waidner, and A. Weber, *The PERSEUS system architecture*, IBM Technical Report RZ 3335 (#93381), IBM Research Division, Zurich Laboratory, 2001.

[24] Ahmad-Reza Sadeghi and Christian Stüble, *Taming "Trusted Computing" by operating system design*, Proceedings of the 4th International Workshop on Information Security Applications (WISA'03) (Cheju Island, Korea), 2003.

[25] _____, *Property-based attestation for computing platforms: Caring about properties, not mechanisms*, in NSPW 2004 [21], pp. 67–77.

[26] Martin Schmitt, *Software-update, configuration and programming of individual vehicles on the aftermarket with an intelligent data-configurator*, in VDI Society for Automotive and Traffic Systems Technology [32], pp. 1021–1046.

[27] Society of Automotive Engineers (SAE) (ed.), *SAE World Congress*, Detroit, Michigan, March 8–11, 2004, URL `http://www.sae.org/`.

[28] S. Stölzl, *Software products for vehicles*, in VDI Society for Automotive and Traffic Systems Technology [32], pp. 1073–1088.

[29] Trusted Computing Group (TCG), *TCG Software Stack Specification, Version 1.1*, Technical specification, URL `https://trustedcomputinggroup.org/`, August 2003.

[30] _____, *TPM Main Specification, Version 1.2*, Technical specification, URL `https://trustedcomputinggroup.org/`, November 2003.

[31] Trusted Computing Platform Alliance (TCPA), *Main Specification, Version 1.1b*, Technical specification, February 2002.

[32] VDI Society for Automotive and Traffic Systems Technology (ed.), *Electronic Systems for Vehicles, VDI Berichte 1789, Congress*, Baden-Baden, Germany, VDI Verlag GmbH Düsseldorf, September 25–26, 2003.