Secure Software Delivery and Installation in Embedded Systems*

André Adelsbach, Ulrich Huber, and Ahmad-Reza Sadeghi

Horst Görtz Institute for IT Security, Ruhr-University Bochum, Germany
Andre.Adelsbach@nds.rub.de
{sadeghi, huber}@crypto.rub.de

Abstract. Increasingly, software (SW) in embedded systems 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 base: with the installation, the user acquires full access rights to any functionality. Concepts for solving individual deficiencies of current procedures have been proposed, but no unified solution has been published so far.

In this paper we propose a method for secure SW delivery and installation in embedded systems. The automotive industry serves as a case example leading to complex trust relations and illustrates typically involved parties and their demands. Our solution combines several cryptographic techniques. For example, public key broadcast encryption enables secure SW distribution from any provider to all relevant embedded systems. Trusted computing allows to bind the distributed SW to a trustworthy configuration of the embedded system, which then fulfills a variety of security requirements. 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 SW in embedded systems used to be tied together as one single product and rarely changed once the system had been shipped. Nowadays, HW and SW in an electronic control unit (ECU) have become separate products. SW can be updated or upgraded after shipment and add customer value due to the ubiquitous use of flashable ECUs. Examples are the ECUs in a modern car where updates can increase the engine performance and reduce emission levels.

Current procedures for installing SW in an embedded ECU are insecure—details about the deficiencies will be given in Sect. 2. Historically, these deficiencies didn't matter because SW installation was focused on warranty-based

* A full version of this paper containing further details is available at [1].

replacement of defective SW. The system owner was informed in costly recalls and received the SW updates free of charge. Recently, a paradigm shift has taken place: value-added SW components can be distributed to interested owners and new business models allow the extraction of revenues even after shipment.

The secure delivery of SW to embedded systems 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 relies on system-specific equipment which is only available to maintenance personnel. For example, a SW update in a vehicle ECU is usually installed via a manufacturer-specific diagnostic tester that is only available to maintenance providers.[1] Second, different classes of such intermediaries exist: depending on their equipment and capabilities, maintenance providers usually have different installation rights. In the automotive example, an uncertified garage might not be granted the right to install SW for safety-relevant ECUs such as the airbag ECU. Third, a newly developed SW component is not necessarily compatible with any target ECU and the SW of all other ECUs in the embedded system. For example, an average compact-class vehicle contains 40 ECUs while high-end and luxury class vehicles can have up to 70 ECUs.[2] Secure SW installation must therefore fulfill a variety of requirements regarding security and usability. Last, new business models for embedded systems will induce new requirements. Due to the high value of the embedded system and the potential consequences of system failure, non-repudiation will be an important requirement.

We propose a procedure for secure SW delivery and installation in embedded systems. We combine a variety of different cryptographic techniques to build such a procedure. The main contribution of our proposal is the secure installation procedure itself based on Public Key Broadcast Encryption (PKBE) and Trusted Computing. Another contribution is a requirement model for all parties that participate in a typical distribution and installation setting. To the authors' knowledge, neither a suitable procedure nor a general requirement model have been previously published although individual requirements have been proposed [9, 2, 10]. The use of the PKBE scheme proposed in [11] has several advantages in this particular setting.[3] First, it enables *efficient* one-way communication from SW providers to a potentially large, but selected set of embedded systems, even though they have to be considered *stateless* receivers containing a fixed set of secret keys which can't be updated. Specifically, the length

[1] Maintenance providers such as dealers, garages and road service teams carry out the SW installation as the car owner lacks the necessary equipment and skills [2].

[2] The Volkswagen Phaeton has 61 ECUs [3]. In addition, each OEM usually has different car models with differing ECU configurations. The ECU configuration of a particular model changes during the production life cycle due to an update of HW or SW components [3, 4]. The compatibility of a SW component does not only depend on the target ECU hardware, but also on other ECUs in the vehicle [5, 6, 7].

[3] Broadcast encryption was first introduced in [12]. Several improvements were proposed, e.g., in [13]. We refer to the public key broadcast encryption scheme of [11].

of the message header does not grow with the number of intended receivers as in the case of a standard Public Key Infrastructure (PKI).[4] Second, the proposed PKBE scheme allows the revocation of an unbounded number of receivers. Even if a large number of receivers has been compromised or is to be excluded, messages can still be broadcast to the remaining receivers. Last, it gives non-discriminatory access to the broadcast channel. The public key property allows any (not necessarily trusted) party to broadcast to any chosen set of receivers. Specifically, the manufacturer of the embedded system can't exclude any SW provider from the broadcast channel or otherwise prevent competition.[5]

Trusted Computing is the enabling technology for an embedded system to become a trusted receiver of broadcast messages. Based on minimum additional hardware and cryptographic techniques such as attestation and sealing, an embedded system can be trusted to be in a particular configuration. The assessment of the compatibility of a particular SW component with the embedded system can be based on this configuration. In order to avoid discrimination of certain SW providers, we suggest the use of property-based attestation [14].

2 Related Work

Several types of embedded systems exist and specific literature on each type is available. However, we consider a modern vehicle to be the most challenging example, namely due to the specific qualities of SW distribution and installation as outlined in Sect. 1. In particular, the high number of ECUs and their variants leads to a complex assessment of compatibility. Therefore, we focus on automotive literature and add an example from the field of IT security. A typical procedure for installing SW in an automotive ECU is described in [15]. It is performed by a so called flashloader, a standard SW environment that allows for in system re-programming of ECUs. Current installation procedures rarely apply any cryptographic techniques [15, 16, 6].

A framework for international automotive SW installation standards is introduced in [16]. However, it doesn't consider any DRM or security aspects. A proposal for "end-to-end security" of SW installation in vehicles is made in [17]. However, the signing of the SW component by "an authorized party" is the only protective measure, which provides only a partial solution[6] to the requirements that we will introduce in Sect. 4. An extended discussion of related work is presented in the full version of this paper [1].

[4] If standard PKI was used, the message header length would be $O(|\mathcal{U}|)$ where $\mathcal{U}$ is the set of intended receivers. In the PKBE scheme from [11], this length is only $O(r)$ where r is the number of revoked or excluded receivers.

[5] Non-discrimination is also important on the receiving end: for instance, the European Commission Regulation 1400/2002 prevents discrimination of independent maintenance providers. The manufacturer must give them access to necessary material and information, e.g., spare parts, technical information and diagnostic equipment.

[6] For 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.

3 Model

3.1 Roles and Objects

The following roles (cf. Fig. 1) will be used throughout this paper:

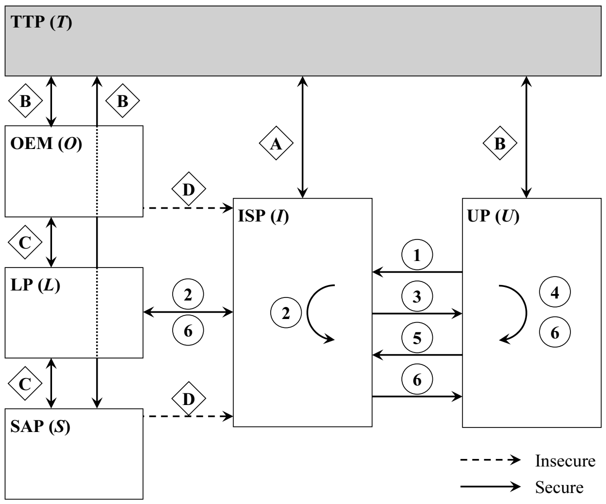

Fig. 1. Installation procedure in six steps

(O) **OEM:** The Overall Equipment Manufacturer (OEM) develops, assembles and delivers the embedded system to users. For this, O cooperates with suppliers that develop and/or manufacture components for the embedded system. The initial SW components at shipment time may be either from O or from his suppliers. Automotive examples are car manufacturers such as Daimler Chrysler, Ford, GM or Toyota.

(S) **SAP:** SW Application Programmers (SAPs) develop SW components for the embedded system. They may either be (i) suppliers that participate in developing and/or assembling the embedded system or (ii) independent application programmers that develop SW components (updates and/or upgrades) and distribute them after shipment. Automotive examples are suppliers such as Bosch, Delphi, Denso, Siemens and Visteon.
We use the term *"SW provider"* as a synonym for "OEM or any SAP".

(I) **ISP:** The Installation Service Providers (ISP) maintain the embedded system, i.e., mechanical parts, ECU HW and SW. As part of their maintenance services, they install updates and/or upgrades of SW components. They have equipment that is necessary for the installation procedure and capabilities that allow them to correctly install SW components. Automotive examples are car dealers, garages and road service teams. The installation rights of I are modeled as clearance levels. Each SW component requires a minimum clearance level $\text{Clear}_{\text{min}}$. I can have any clearance level in $\{1, 2, \ldots, m\}$. If I has clearance level i, it may install any SW with $\text{Clear}_{\text{min}} \leq i$. The highest level m permits the installation of any SW. Without clearance level, no SW may be installed.[7]

(L) **LP:** The License Provider (LP) distributes licenses for SW components that the SW providers O and S have developed. Prior to distribution of a license, L needs to establish terms and conditions with the SW providers in which the model for sharing license revenues is detailed.[8] To the authors' knowledge, automotive examples don't exist yet, but might be established as spin-offs of OEMs and SAPs.

(U) **UP:** The User Platform (UP) is manufactured by O and purchased by the user. The user is interested in SW for U and willing to pay for it if it offers a perceivable value-added. We define U's configuration as the collective information on each SW (and implicitly HW) component that is installed in U. The obvious automotive example for U is a car. We assume U to have an internal communication network over which all its components denoted by $\{u_0, u_1, \ldots, u_n\}$ are connected. In the implementation of an embedded system, they correspond to ECUs. u_0 is assumed to be the trusted computing base and provides a central installation and license service. u_0 is the only component capable of distributing new SW to the other components u_i, $1 \leq i \leq n$. Due to cost constraints, we cannot assume the u_i to be high-performance components, i.e., their computational resources are limited, especially related to cryptographic techniques. The SW distribution from u_0 to the u_i is performed over an internal communication network to which all components are connected.[9]

(T) **TTP:** The Trusted Third Party (TTP) 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. With SW properties we mean characteristic features of SW such as functionality, interfaces, supported protocols, memory and processor requirements, necessary environment, etc. Second, T creates clearance level certificates which certify I'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 each S must trust O. However, an independent T becomes necessary if O is not fully trusted and discrimination of any S should be avoided. An independent T might evolve out of safety standards authorities such as the NHTSA[10] in the USA or the Euro NCAP[11] in Europe.

[7] Other models for installation rights can easily be integrated into our proposal. For the purpose of this paper, clearance levels serve as an example.

[8] The discussion of licensing models, e.g., pay-per-installation or pay-per-usage, is out of the scope of this paper.

[9] In the automotive example, this holds for communication networks ("data busses") such as CAN, LIN and MOST.

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

[11] `http://www.euroncap.com/`

4 Security Requirements

We consider the security requirements of each party separately. The following terms will be used in this section: when the installation results in *success*, we mean the execution of a complete installation. A *complete installation* includes the installation of a legal SW component and the delivery of a legal license. A *legal SW component* is a SW component whose properties have been certified by T and committed by the SW provider. A *legal license* is a license which was legally generated by L and legally acquired by U. With *failure* we mean that no SW is installed, i.e., U's configuration does not change. A *legal* I *for a specific SW* is defined to be an I with an authentic clearance level certificate from T with a clearance level sufficient for the SW. A *legal* U does neither request illegal nor incompatible SW nor involve an illegal I.

4.1 OEM Requirements

(OCR) Correctness: The result of the installation procedure must be success if and only if all involved parties behave according to the specified protocol.

(OPE) Policy Enforcement: O requires enforcement of following policies:

- **(OPE1) Rights Enforcement:** After acceptance by L, the terms and conditions of O should not be circumvented.
- **(OPE2) Compatibility Enforcement:** An installation will result in *success* only if the SW and U are compatible. *Compatibility* of a SW and U means that the SW properties are conform to and suitable for U's configuration. For example, this implies that the SW must run correctly on U and may not have inconsistent interfaces.
- **(OPE3) ISP Clearance Enforcement:** Only a legal I may install SW.[12]

(OCF) Confidentiality: No party except O and the trusted component u_0 of U may be capable of reading SW developed by O *prior to* installation.[13] This is meant to protect the intellectual property contained in O's SW. However, we only consider conditional access to the SW.[14]

(OI) Integrity: The installed SW component must be integer.

4.2 SAP Requirements

S shares all requirements with O, but has an important additional requirement:

(SND) Non-discrimination: The identity of S may neither influence S's ability to send over the broadcast channel nor the result of the installation procedure.

[12] For example, this protects O from warranty claims of the user when the user pretends that O and I have colluded to install SW with an illegal clearance level certificate.

[13] This also excludes I from reading the cleartext. However, I will still be necessary in most installation procedures because I has the necessary skill set, installation equipment, maintenance area, spare parts, etc.

[14] Complementary measures, e.g., fingerprinting, are out of the scope of this paper.

For example, when S_1 and S_2 have each developed a legal SW with the same properties, S_1 may not be *technically* preferred in the installation procedure.[15]

4.3 ISP Requirements

(ICR) Correctness: This requirement is identical to the requirement OCR.

(INR) Non-repudiation: After each installation procedure, I must be able to prove the origin and the result of the installation to any honest party.

(ICE) Clearance Enforcement: This requirement is identical to OPE3. For example, this justifies I's effort to obtain a clearance level certificate.

(IND) Non-discrimination: A legal I must be able to install *any* SW component which U requests and which is at or below his clearance level. For example, the SW provider may not be able to separate ISPs with identical clearance level into subgroups and exclude individual subgroups from the SW distribution.

(IFP) Frame-Proofness: If no installation has occurred, I may not be wrongly accused of treachery, e.g., of having installed SW.

4.4 License Provider Requirements

(LNR) Non-repudiation: A licensee cannot deny the receipt of a legal license.[16]

4.5 User Requirements

(UCR) Correctness: This requirement is identical to the requirement OCR.

(UNR) Non-repudiation: After the installation procedure, U must be able to prove the result, i.e., either success or failure, to any honest party.

(UIO) Installation Origin: No SW installation may be performed without request by U.

(UA) Authenticity: The installed SW component and the license must be authentic, i.e., as requested by U and sent by the SW provider and L respectively.

5 Proposed Solution

This section provides a summary of the proposed installation procedure (cf. Fig. 1) that consists of a setup period (Phases A–D) and the actual installation (Steps 1–6). The protocols for these two parts will be detailed in Sect. 5.2.

We first give an overview of installation procedure: in the setup period, the system parameters, e.g., security parameters of the cryptographic schemes, are chosen. Each I applies for a specific clearance level and is certified by T. This

[15] However, non-technical influence of O on the user cannot be prevented, e.g., when O advertises for S_1's products.

[16] For example, U cannot receive a legal license and later refuse payment, pretending he never received the license.

certification is performed once and repeated only if a new I joins the system or existing certificates expire. In parallel, a SW provider who has developed a new SW component submits it to T and requests certification of the SW properties. After certification, the SW provider establishes terms and conditions with L. Both steps need to be done for each new component.[17] Finally, the SW component is distributed to each I via the broadcast channel. The actual installation starts with an installation request by U. I checks if it has the necessary clearance level and, if so, obtains a license from L. After delivery of SW and license to U, u_0 checks if the SW, the license and I are legal (for definitions see Sect. 4). If so, u_0 instructs the target component u_i to install the SW. u_0 then confirms the successful installation to I and awaits I's acknowledgment. After receiving the acknowledgment, u_0 instructs u_i to use the SW.

5.1 Conventions, Building Blocks and Message Formats

- $\mathsf{ID}()$ is a function that maps a principal or an object to a unique identifier.
- $\mathsf{Hash}()$ is a cryptographic hash function.
- ($\mathsf{GenKey_A}()$, $\mathsf{Sign}()$, $\mathsf{Verify}()$) denotes the key generation, signing and verifying of a digital signature scheme. $\sigma_X \leftarrow \mathsf{Sign}(k_X^{\text{sign}}; M)$ means the signing of the message M with X's signing key k_X^{sign}, resulting in the signature $\sigma_X = (M, \mathrm{Sig}(M))$. $ind \leftarrow \mathsf{Verify}(k_X^{\text{test}}; \sigma_X)$ means the verification of σ_X with the key k_X^{test}. The result of the verification is the Boolean value ind.
- ($\mathsf{GenKey_P}()$, $\mathsf{Reg}()$, $\mathsf{Enc_P}()$, $\mathsf{Dec_P}()$) is a tuple that denotes the key generation, user registration, encryption and decryption of a PKBE scheme. $\mathsf{GenKey_P}()$ is used by T to set up all the parameters of the scheme, e.g., the set of all public keys $\mathcal{K}^{\text{enc}}$ which is available to any party. $\mathsf{Reg}()$ is used by T to compute the set of secret keys $\mathcal{K}_U^{\text{dec}}$ to be delivered to a user U. $\mathsf{Enc_P}(\mathcal{K}^{\text{enc}}, \mathcal{U}; M)$ is used by a (not necessarily trusted) sender to encapsulate a message M with the set of public keys $\mathcal{K}^{\text{enc}}$ in such a way that only the unrevoked users $\mathcal{U}$ can recover it. $\mathsf{Dec_P}(\mathcal{K}_U^{\text{dec}}; C)$ is used by a user U to decipher C with his private key set $\mathcal{K}_U^{\text{dec}}$ and returns M if and only if the user is unrevoked, i.e., $U \in \mathcal{U}$.
- ($\mathsf{GenKey_S}()$, $\mathsf{Enc_S}()$, $\mathsf{Dec_S}()$) is a tuple that denotes a symmetric encryption scheme for key generation, encryption and decryption. The shared key of X and Y is denoted $k_{X,Y}$ (for details on sharing the key, cf. [1]).
- $\mathsf{MAC}(k_{X,Y}; M)$ is a function that calculates the Message Authentication Code (MAC) of message M under the shared key $k_{X,Y}$ of X and Y.
- $\mathrm{Clear}(I)$ denotes the clearance level of the ISP I. $\mathrm{Clear}_{\min}(s)$ denotes the minimum clearance level that is required for an ISP to install the SW s.
- $\mathsf{Comp}(U; P_1^s, P_2^s, \ldots)$ denotes a compatibility check function that returns `true` iff the requested SW s and U are compatible (cf. Sect. 4.1). The compatibility check $\mathsf{Comp}()$ is computed by u_0 based on the properties P_i^s of s (see Sect. 3.1 on p. 259). For this purpose, u_0 interprets those properties and

[17] However, a SW provider and L might establish more general terms and conditions which cover a whole set of SW components.

derives requirements for U such as interfaces, protocols, minimum memory and processing power, etc. If U fulfills all of the requirements, $\mathsf{Comp}()$ returns `true` which confirms compatibility of U and s. If any requirement is unfulfilled, $\mathsf{Comp}()$ returns `false`.[18]

- $\mathsf{Target}(U; P_1^s, P_2^s, \ldots)$ denotes a function which returns the target component u_i, $u_i \in \{u_1, \ldots, u_n\}$ on which the SW s is to be installed.
- $\mathcal{R}_U = \{r_1^U, r_2^U, \ldots\}$ denotes the set of rights that U asks for when it sends an installation request. An example for r_i^U is a one-year validity period.
- $\mathsf{right}(\mathcal{R}_U(\sigma), i)$ denotes a separator which returns the right r_i^U of $\mathcal{R}_U = \{r_1^U, r_2^U, \ldots\}$ where σ is a signature on $\mathcal{R}_U$ or on a string containing $\mathcal{R}_U$.
- $instr_{u_i} \leftarrow \mathsf{install}(\mathsf{ID}(u_i), \mathsf{ID}(s), s_{\text{enc}}^{u_i})$ is an order from u_0 to u_i to install $s_{\text{enc}}^{u_i}$.
- $\widetilde{instr}_{u_i} \leftarrow \mathsf{use}(\mathsf{ID}(u_i), \mathsf{ID}(s); p_1, p_2, \ldots)$ denotes an order from u_0 to u_i to use the SW s with the input parameters $(p_1, p_2, \ldots)$. For example, if $p_i \in \{0, 1\}$ represents a functionality of s, then this functionality is activated for $p_i = 1$ and deactivated for $p_i = 0$.[19] u_0 derives the parameters from the rights $\mathcal{R}_U$ granted in the license.

5.2 Protocols

Setup. The setup period consists of four phases A–D (cf. Fig. 1):

Phase A: Each ISP applies for certification of a particular clearance level.[20] The result of the certification process is the clearance level certificate ζ_I, more precisely $\zeta_I \leftarrow \mathsf{Sign}(k_T^{\text{sign}}; \mathsf{ID}(I), k_I^{\text{test}}, \text{Clear}(I))$.

Phase B: T sets up the PKBE scheme, publishes the public keys and provides each U with its private keys. In addition, every party distributes its test key, e.g., using T to certify and distribute the public keys. Each SW provider (either O or S) encrypts any newly developed SW component s for later distribution via the broadcast channel (1). He computes a hash $h \leftarrow \mathsf{Hash}(s)$, generates the SW signature $\sigma_{O|S}^{\text{SW}} \leftarrow \mathsf{Sign}(k_{O|S}^{\text{sign}}; h)$ and sends the property certification request $(s, \sigma_{O|S}^{\text{SW}})$ to T.[21] With $O|S$ we mean any of the two roles O and S

$$s_{\text{enc}} \leftarrow \mathsf{Enc}_{\mathsf{P}}(\mathcal{K}^{\text{enc}}, \mathcal{U}; s). \tag{1}$$

T verifies the SW signature $\texttt{true} \stackrel{?}{=} \mathsf{Verify}(k_{O|S}^{\text{test}}; \sigma_{O|S}^{\text{SW}})$ and the hash value $h \stackrel{?}{=} \mathsf{Hash}(s)$. If both are valid, T generates the SW property certificate ζ_s for the

[18] For example, `false` might be the result when U and s have inconsistent interfaces.

[19] In the automotive example, the functionality might be additional horsepower.

[20] Many certification models are possible, but we omit their discussion here. One example is a joint definition of clearance level requirements by T, O and S, possibly including spokespersons of I and official authorities.

[21] Care has to be taken in order to avoid security vulnerabilities when signing an encryption [18].

SW s in (2). For example, the SW provider may submit the claimed properties of his SW which T then verifies:[22]

$$\zeta_s \leftarrow \mathsf{Sign}(k_T^{\text{sign}}; \mathsf{ID}(s), P_1^s, P_2^s, \ldots), \quad P_1^s := \text{Clear}_{\min}(s),\ P_2^s := \mathsf{Hash}(s). \quad (2)$$

We use $\text{Clear}_{\min}(s)$ and $\mathsf{Hash}(s)$ as the first two properties because this simplifies the notation: both properties need to be certified by T.

Phase C: During this step, terms and conditions between the SW providers and L are negotiated and committed. Afterwards L can independently create licenses for any U of the form:

$$\gamma_L \leftarrow \mathsf{Sign}(k_L^{\text{sign}}; \texttt{license}, \mathsf{ID}(U), \mathsf{ID}(s), \mathcal{R}_U). \quad (3)$$

Phase D: The SW provider signs the property certificate in order to commit to the properties of s. Finally, he broadcasts the encrypted SW component together with his signature $\sigma_{O|S}^{\text{comm}} \leftarrow \mathsf{Sign}(k_{O|S}^{\text{sign}}; \zeta_s)$.

Installation of a SW Component. After this setup phase, the installation procedure for a specific SW component can start:

1. In the first step, U sends a signed installation request σ_U^{req} to I. The request contains the identifier of the requested SW s and the desired rights $\mathcal{R}_U$

$$\sigma_U^{\text{req}} \leftarrow \mathsf{Sign}(k_U^{\text{sign}}; \rho_U), \quad \rho_U = (\mathsf{ID}(U), \mathsf{ID}(s), \mathcal{R}_U) \text{ and } \mathcal{R}_U = \{r_1^U, \ldots\}. \quad (4)$$

2. I verifies U's signature $\texttt{true} \overset{?}{=} \mathsf{Verify}(k_U^{\text{test}}; \sigma_U^{\text{req}})$ and its own clearance level $\mathsf{Clear}(I) \overset{?}{\geq} \mathsf{Clear}_{\mathsf{min}}(s)$. If both are valid, I obtains a license for U from L of the form (3) and signs the installation package $(\gamma_L, s_{\text{enc}})$ for L and I (5):

$$\sigma_I^{\text{inst}} \leftarrow \mathsf{Sign}(k_I^{\text{sign}}; \gamma_L, s_{\text{enc}}). \quad (5)$$

   If at least one condition remains unfulfilled, I sends a signed denial to U.
3. In case of success, I sends the tuple $(\sigma_I^{\text{inst}}, \zeta_I, \zeta_s, \sigma_{O|S}^{\text{comm}})$ to U where ζ_I represents his clearance level certificate, ζ_s the SW properties certificate and $\sigma_{O|S}^{\text{comm}}$ the SW provider's commitment to ζ_s.
4. The trusted component u_0 verifies that the SW s was indeed requested (6), I possesses an authentic clearance level certificate ζ_I (7), I has the necessary clearance level (8), the SW property certificate ζ_s is authentic (9), the delivered SW component is identical to the SW component referred to in the property certificate (10), s and U are compatible (11), the SW provider has made a commitment to ζ_s (12), I has added his signature σ_I^{inst} (13), I has delivered a legal license (14), which grants the requested rights $\mathcal{R}_U$ (15):

$$\texttt{true} \overset{?}{=} \exists\ \rho_U \text{ for } \mathsf{ID}(s) \quad (6)$$

[22] In a different trust model, O might be the party that certifies SW properties. This would significantly reduce the workload on T. However, it would require all S to trust O or result in dispute if O denied fair evaluation.

$$\texttt{true} \overset{?}{=} \mathsf{Verify}(k_T^{\text{test}}; \zeta_I) \tag{7}$$

$$\mathsf{Clear}(I) \overset{?}{\geq} \mathsf{Clear}_{\mathsf{min}}(s) \tag{8}$$

$$\texttt{true} \overset{?}{=} \mathsf{Verify}(k_T^{\text{test}}; \zeta_s) \tag{9}$$

$$P_2^s \overset{?}{=} \mathsf{Hash}(s) \tag{10}$$

$$\texttt{true} \overset{?}{=} \mathsf{Comp}(U; P_1^s, P_2^s, \ldots) \tag{11}$$

$$\texttt{true} \overset{?}{=} \mathsf{Verify}(k_{O|S}^{\text{test}}; \sigma_{O|S}^{\text{comm}}) \tag{12}$$

$$\texttt{true} \overset{?}{=} \mathsf{Verify}(k_I^{\text{test}}; \sigma_I^{\text{inst}}) \tag{13}$$

$$\texttt{true} \overset{?}{=} \mathsf{Verify}(k_L^{\text{test}}; \gamma_L) \tag{14}$$

$$\mathsf{right}(\mathcal{R}_U(\gamma_L), i) \overset{?}{=} \mathsf{right}(\mathcal{R}_U(\rho_U), i) \quad \forall\, i. \tag{15}$$

If all conditions are fulfilled, u_0 finds the appropriate subset of the PKBE scheme and decrypts s_{enc} with the corresp. private key: $s \leftarrow \mathsf{Dec}_{\mathsf{P}}(\mathcal{K}_U^{\text{dec}}; s_{\text{enc}})$. Then u_0 determines the target ECU u_i in (16) and re-encrypts s for u_i with a symmetric key k_{u_0,u_i} shared only with u_i.[23] Subsequently, u_0 invokes u_i to install the SW component by sending the tuple $(instr_{u_i}, mac_{u_i})$ in (17). The message $instr_{u_i}$ provides u_i with the encrypted SW via U's internal communication network. The MAC mac_{u_i} confirms the authenticity of $instr_{u_i}$ while mac_{u_0} is u_i's confirmation to u_0 that s was successfully installed:

$$u_i \leftarrow \mathsf{Target}(U; P_1^s, P_2^s, \ldots) \quad \text{with} \quad u_i \in \{u_1, \ldots, u_n\} \tag{16}$$

$$s_{\text{enc}}^{u_i} \leftarrow \mathsf{Enc}_{\mathsf{S}}(k_{u_0,u_i}; s) \tag{17}$$

$$instr_{u_i} \leftarrow \mathsf{install}(\mathsf{ID}(u_i), \mathsf{ID}(s), s_{\text{enc}}^{u_i}) \tag{18}$$

$$mac_{u_i} \leftarrow \mathsf{MAC}(k_{u_0,u_i}; instr_{u_i}) \tag{19}$$

$$mac_{u_0} \leftarrow \mathsf{MAC}(k_{u_0,u_i}; \mathsf{ID}(s)). \tag{20}$$

5. After the installation, U confirms the result of the installation request ρ_U. For this purpose, u_0 uses the indicator $ind \in \{\texttt{true}, \texttt{false}\}$ where `true` represents success and `false` represents failure. u_0 adds the signature σ_U^{conf} and sends $(\rho_U, \gamma_L, ind, \sigma_U^{\text{conf}})$ to I, where $\sigma_U^{\text{conf}} \leftarrow \mathsf{Sign}(k_U^{\text{sign}}; \rho_U, \gamma_L, ind)$
6. I verifies the confirmation in (21) and forwards U's confirmation to L. I also sends an acknowledgment back to U (22). Within U, u_0 checks the acknowledgment (23) and, if it is authentic, u_0 invokes u_i to use the SW component with parameters $p_1, p_2, \ldots$ (24):

$$\texttt{true} \overset{?}{=} \mathsf{Verify}(k_U^{\text{test}}; \sigma_U^{\text{conf}}) \tag{21}$$

$$\sigma_I^{\text{ack}} \leftarrow \mathsf{Sign}(k_I^{\text{sign}}; \sigma_U^{\text{conf}}) \tag{22}$$

$$\texttt{true} \overset{?}{=} \mathsf{Verify}(k_I^{\text{test}}; \sigma_I^{\text{ack}}) \tag{23}$$

[23] The generation of k_{u_0,u_i} will be detailed in [1]. Meanwhile, we assume it exists.

$$\widetilde{instr}_{u_i} \leftarrow \mathsf{use}(\mathsf{ID}(u_i), \mathsf{ID}(s); p_1, p_2, \ldots) \tag{24}$$

$$\widetilde{mac}_{u_i} \leftarrow \mathsf{MAC}(k_{u_0,u_i}; \widetilde{instr}_{u_i}). \tag{25}$$

After receiving and verifying the instruction $(\widetilde{instr}_{u_i}, \widetilde{mac}_{u_i})$, u_i uses the new SW component s with parameters $(p_1, p_2, \ldots)$. u_0 stores all licenses and periodically checks if any of them has expired. When a license expires, u_0 tells u_i to execute the SW with different parameters. For example, the new parameters might instruct u_i to stop using the SW or switch off some functionality, e.g., the additional horsepower in the automotive example. In addition, u_0 indicates the need for a new license to the user.
If the installation failed, U uses the old platform configuration.

6 Assumptions, Security Analysis and Implementation

Due to space constraints, we present these sections in the full paper [1].

7 Conclusion

In this paper we have proposed a procedure for secure SW delivery and installation in embedded systems. It integrates installation service providers as intermediaries between SW provider and embedded system and categorizes them into separate clearance levels. Compatibility of the SW component and the target system is checked prior to installation. The fulfillment of a variety of requirements and the introduction of an elementary license system allows any SW provider to establish new business models that are currently not supported. The SW provider's intellectual property is protected and a variety of digital rights is supported. From the embedded system owner's point of view, the procedure prevents installation of illegal SW and supports warranty claims against the SW provider in case of defective SW with unambiguous evidence. Public Key Broadcast Encryption enables efficient communication with embedded system on an insecure one-way channel. The use of Trusted Computing concepts induces the necessary trust in the embedded system.

References

1. Adelsbach, A., Huber, U., Sadeghi, A.R.: Secure software delivery and installation in embedded systems. Full version, (`http://www.prosec.rub.de/publications`)
2. Heinisch, C., Simons, M.: 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). [20] URL `http://www.sae.org/`.
3. Heinrich, A., Müller, K., Fehrling, J., Paggel, A., Schneider, I.: Version management for transparency and process reliability in the ECU development. [19] 219–230

4. Schmitt, M.: Software-update, configuration and programming of individual vehicles on the aftermarket with an intelligent data-configurator. [19] 1021–1046
5. Alminger, H., Josefsson, O.: Software handling during the vehicle lifecycle. [19] 1047–1055
6. Huber, M., Weber, T., Miehling, T.: Standard software for in-vehicle flash reprogramming. [19] 1011–1020
7. Oeftiger, U.: Diagnose und Reparatur elektronisch unterstützter Fahrzeuge. [8]
8. Euroforum, ed.: Jahrestagung Elektronik-Systeme im Automobil, Fachtag Design – Test – Diagnose elektronischer Systeme, Munich (2004)
9. BMW Car IT: Das Potenzial von Software im Fahrzeug. Press report, BMW Group, URL `http://www.bmw-carit.de/pdf/plakate.pdf` (2002)
10. Stölzl, S.: Software products for vehicles. [19] 1073–1088
11. Dodis, Y., Fazio, N.: Public key broadcast encryption for stateless receivers. In Feigenbaum, J., ed.: Digital Rights Management Workshop. Volume 2696 of Lecture Notes in Computer Science., Springer Verlag (2003) 61–80
12. Fiat, A., Naor, M.: Broadcast encryption. Advances in Cryptology—CRYPTO '93 Proceedings, Lecture Notes in Computer Science **773** (1994) 480–491
13. Naor, D., Naor, M., Lotspiech, J.: Revocation and tracing schemes for stateless receivers. Advances in Cryptology—CRYPTO '01 Proceedings, Lecture Notes in Computer Science **2139** (2001) 41–62
14. Sadeghi, A.R., Stüble, C.: Property-based attestation for computing platforms: Caring about properties, not mechanisms. (2004)
15. 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` (2002)
16. Dallmayr, C., Schlüter, O.: ECU software development with diagnostics and flash down-loading according to international standards (SAE Technical Paper Series 2004-01-0273). [20] URL `http://www.sae.org/`.
17. Müller, M.: IT-Security in Fahrzeugnetzen. Elektronik Automotive (2004) 54–59 ISSN: 1614-0125.
18. An, J.H., Dodis, Y., Rabin, T.: On the security of joint signature and encryption. In: EUROCRYPT '02, Springer-Verlag (2002) 83–107
19. VDI Society for Automotive and Traffic Systems Technology, ed.: Electronic Systems for Vehicles. In VDI Society for Automotive and Traffic Systems Technology, ed.: Electronic Systems for Vehicles, VDI Berichte 1789, Congress, Baden-Baden, Germany, VDI Verlag GmbH Düsseldorf (2003)
20. Society of Automotive Engineers (SAE), ed.: SAE World Congress. In Society of Automotive Engineers (SAE), ed.: 2004 SAE World Congress, Detroit, Michigan, March 8–11, 2004, Detroit, Michigan (2004) URL `http://www.sae.org/`.