# Component Identification: Enabler for Secure Networks of Complex Systems

(preliminary work)

André Weimerskirch [1], Katrin Höper [3], Christof Paar [1,2], and Marko Wolf [1,2]

[1] escrypt - Embedded Security GmbH, Bochum, Germany
[2] Communication Security Group, Ruhr-University Bochum, Germany
Email: {aweimerskirch, cpaar, mwolf}@escrypt.com
[3] Department of E&CE, University of Waterloo, Canada
Email: khoeper@engmail.uwaterloo.ca

## Abstract

The ad-hoc network technology matures and starts becoming of interest for several industries as enabler for more safety, comfortability, and new business models. However, such network systems must not only fulfill the conventional security requirements like efficiency, robustness, and privacy, but they must be resistant to manipulation by the system owners and further involved parties. For instance, if the considered network is an inter-vehicular network consisting of vehicles and the car owner is able to manipulate his car in a way that it does not act in a proper way, the car-traffic's safety might be endangered. In this work we propose protocols for component identification in the context of network nodes that are complex systems (e.g., vehicles) composed by many components (e.g., engine). The component identification provides protection against manipulation, counterfeits, and theft by implementing a mechanism to bind security relevant components to a given system. Without secure component identification the implementation of networks might be vulnerable to attacks that can easily be mounted. Therefore, the secure component identification is an enabler for secure network protocols.

## 1 Introduction

A basic enabler for networks consisting of nodes that are complex systems is a unique identifier for each node. For ease of consideration we consider in the following the system to be a car consisting of several components such as an engine, and the network to be an inter-vehicular network. The identification of a car would be the electronic counterpart to today's license plates. A core requirement is that the hardware module that implements the electronic license plate cannot be manipulated nor faked or stolen in order to install it in another car. The same requirements hold for all security related components. As a car is entirely accessible by its owner, the approach of ensuring such security goals is not obvious. The straightforward idea is to have a tamper resistant hardware module that implements all security mechanisms. For instance, this security module would implement the electronic license plate in order to broadcast a unique identification string over a built-in radio. However, an adversary could just steal the security module and build it into his car for impersonation, or he could remove the security module from his car. Thus, there needs to be a bond between components and the vehicle. Now, if an adversary manipulates or removes any security related part of the vehicle, the vehicle stops working.

Even more, if an adversary manipulates any device that is related to safety, e.g. the airbag or brake system, the vehicle will finally stop working. We propose a scheme for providing component identification in order to bind security and safety related parts of a vehicle to a central security module. Only if there is any such mechanism implemented in a car, security of ad-hoc networks between cars can be implemented in a way that suffices legal requirements. Otherwise, if a car owner is able to manipulate his car in a way that it broadcasts malicious messages, traffic might be affected or endangered. Hence, our scheme is an enabler for security of ad-hoc networks between cars. On the other hand, all components form an ad-hoc network by itself. New components can be added and replaced, and components are provided with a secure communication channel. Traditional methods use tags, e.g. holographic stickers, that are supposed to be unforgeable. However, as it can be seen in the airplane industry, such tags can easily be bought on the black market [2].

In the following we present a scheme to bind crucial components to a complex system such as a car. We use component identification to secure all components of a car against cloning (faked parts), manipulation, and theft. In particular, our scheme protects the car owner against bogus parts that endanger the operation of the car, and it minimizes the financial damage of suppliers. This damage is estimated to be as high as 250 billion Euros worldwide [3]. Our first protocol is based on a central security module which we call High Security Module (HSM). An HSM is a tamper resistant security module that is able to perform cryptographic operations. The key data is stored inside of the HSM. Trying to compromise the HSM in order to get a key will destroy the HSM and delete its memory such that the keys are lost. Since building such a module is hard to achieve and costly the second protocol we present gets rid of the HSM by distributing the task of the HSM to all components. Our solution is very general and easy to apply. There is no need to adjust components before they are installed, and all processes can be executed automatically. Furthermore, our scheme does not only provide a secure environment for the owner of a car but also for all involved parties such as the manufacturer and the mechanics. Our solution is applicable to cars but also to other transportation vehicles such as trains and airplanes, and also to consumer electronics such as printers with cartridges. Furthermore, our work is applicable to other systems which consist of components that need protection against cloning, manipulation, and theft.

### 1.1 Assumptions

For the remaining we assume the following.

1. A simple computing tag is attached to each crucial component in such a way that removing the tag destroys the component.

2. The HSM as well as the computing tags are able to perform cryptographic operations.

3. (optional) There is a temporary connection available between the vehicle and a central server.

Assumption (3) can be omitted if there is no connection available. Protection against cloned parts is then not possible, though. We present approaches where both an HSM is used and where it can be omitted. Also we present approaches where asymmetric cryptographic methods can be replaced by symmetric methods such that low-cost RFID tags can be attached to the components. We now present our assumptions in more detail:

- Each component is equipped with a small CPU tag. This tag, for instance a smart card or RFID chip, is able to perform basic cryptographic operations. It is advantageous to bind the module to the component in such a way that removing or tampering with the module destroys the component as well as the cryptographic secrets.

- The component tags communicate over a wireless communication channel. All components are able to communicate to each other component either by a single-hop or multi-hop communication channel.

- To perform cryptographic operations and to transmit data packets the component tags need quite some energy. Hence, we assume that they are equipped with a battery having a life span of several years that outlasts the lifetime of the component. We also present a solution for passive RFID tags that do not require a battery.
- There is a trusted third authority that issues certificates, e.g., a government institution or a car manufacturer.
- The components or a central security module are able to take actions in case that a protocol step fails. For instance, the central security module might share a key with crucial car components. It then orders these components to react in case of a failure.
- If any check in the protocol fails, an alarm is issued. For instance, the engine stops working, or an alarm message is broadcasted.
- Each component holds a unique $ID$ as well as a cryptographic secret $SK$. $ID$ comprises a unique identification string as well as further information such as the manufacturing date and the component's quality class.
- The cryptographic mechanisms are well chosen and implemented. In particular, it is not possible to measure any side-channel information such as power supply or EMF radiation.

It is well known that tamper resistant hardware modules are hard to produce and costly [1]. In a complex environment, a tamper resistant device such as an HSM might be possible, however. The cost for the HSM can be high but are still low relative to the system's cost. However, later on we present a solution that does not require an HSM to overcome this drawback.

### 1.2 Adversarial Model

For the attacker we assume the following. In short, we assume that the adversary has full control over the communication channel. Hence, the adversary can

- read all messages sent from any component,
- modify messages, delay them or send them multiple times, and
- send messages generated by himself to any component.

Furthermore, an adversary has full physical access to the car and its components. Using sophisticated technologies the adversary might be able to recover secret key data of the component tags. Since the lifetime of components and tags will be a decade or so, the secret key material might be exposed to such attacks after a while. The adversary has full control over all components, and he can program faked component tags.

The backend system is not vulnerable to any attacks. We assume that there are no insider attacks, that central servers and directories are not vulnerable and cannot be read or manipulated by non-authorized entities. The adversary has several aims:

1. to a given car install a component that is not an original one
2. to any car install a component that is not an original one
3. modify or replace a component in any car by a component that is not an original one
4. steal a component of any car and install it to another one

The aims (1) and (2) can be seen as the attack of a single entity, e.g., the owner of a car who wants to install a faked component to his car. Aims (3) and (4) might be seen as the objectives of organized groups that want to distributed stolen, faked, or cloned components.

## 2 Asymmetric Component Identification

The main goal is to bind components to a vehicle. We first present a solution based on an HSM using asymmetric cryptography. The main threat for a car arises if components are stolen,

manipulated, or cloned. We define *cloning* as the act of rebuilding a component up to a perfect copy, in some cases even including secret cryptographic keys. Hence, our scheme provides piracy protection, system protection, and theft protection. *Piracy protection* provides the ability to identify original parts and the possibility to detect counterfeits and cloned components. *System protection* provides system integrity by monitoring systems for unauthorized changes, and *theft protection* prevents the use of stolen components in another system. We are considering a system in which all components hold authentic data which can be verified via a data bus whereby all claimants and at least one verifier are connected to the bus. There are three phases that we consider: (1) The installation of a component into a car, (2) the running system, and (3) the demounting of a component out of the car. We describe our security goals by an example. Only original parts can be built into the car. Every time the ignition key is turned the system integrity is checked, and only in case of an unaltered system the engine starts. A display in the dash board shows the status of all present system components. This could be useful, e.g, to prove that the mechanics of a car garage used original parts only. There might also be a button to manually start the check such that the owner can prove integrity to a third party. For instance, the owner can prove to police that a proper electronic license plate is built in.

We assume the same vehicle environment as above. Furthermore, we assume the following:

- There is an HSM inside of each car that is considered to be tamper resistant.
- The component tag is able to perform asymmetric operations.
- The HSM performs cryptographic operations and stores secret key data. The HSM might be a component itself. For instance, the role of the HSM might be executed by the board computer. The HSM is able to take actions in case of a malicious behavior of a component. For instance, the HSM might share a special secret key with the engine in order to send the engine a command to turn itself off.
- Each component holds a certificate $\langle PK, ID \rangle$ as well as the corresponding secret key $SK$. $ID$ comprises a unique identification string as well as further information such as the manufacturing date and the component's quality class. Certificates are only issued for trustworthy components, i.e. components of trustworthy manufacturers.
- The HSM of a vehicle holds a list $\mathcal{UL}$ of all components built into the vehicle. This list is regularly synchronized with a global list $\mathcal{GL}$ of all components. The synchronization is performed in a secure manner to avoid any manipulation. Each component can be uniquely identified.
- There might be a global certificate revocation list ($\mathcal{CRL}$) with all components that were revoked. A component might be revoked if it was stolen, or if it is known that it was cloned.

All identifications are performed in a challenge-response manner. After $A$ presents a certificate, $B$ needs to check whether $A$ has knowledge of the corresponding secret key $SK$. Algorithm 1 provides this check.

**Algorithm 1** Challenge-response identification

1: $B \rightarrow A : r_B$
2: $A \rightarrow B : r_A, S := SIG(r_A || r_B || B, SK)$
3: $B$ checks whether $VER(S, PK) \stackrel{?}{=} 'valid'$

The check of a symmetric key $K$ is performed in a similar way by using a MAC. Algorithm 2 provides this check in a mutual way. Here, $A$ checks whether $B$ knows the secret key $K$ and also $B$ checks whether $A$ knows the secret key.

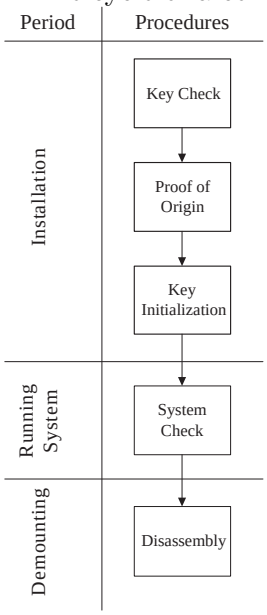
Note that in general all messages of components are encrypted and authenticated by a commonly shared key $K$.

As already mentioned there are three phases to consider when talking about component identification which we will discuss in detail. The main idea is to have an initial component, e.g., the HSM that is imprinted with a secret key $K$.

**Algorithm 2** Mutual challenge-response check of a symmetric key

1: $B \rightarrow A : r_B$
2: $A \rightarrow B : r_A, M_A := MAC(r_A||r_B||B, K)$
3: $B$ checks whether $MAC(r_A||r_B||B, K) \stackrel{?}{=} M_A$
4: $B \rightarrow A : M_B := MAC(r_B||r_A||A, K)$
5: $A$ checks whether $MAC(r_B||r_A||A, K) \stackrel{?}{=} M_B$

Figure 1: Life cycle of a component




Each component holds a certificate. If a component is added to the vehicle, the certificate is checked. Then the component obtains the secret vehicle key $K$ which is checked while the car is running in order to prevent manipulations after the installation. Finally, our solution allows a controlled demounting of a component in order to distinguish stolen components from properly demounted ones. The life cycle of a component is depicted in Figure 1.

## 2.1 Initialization

At initialization time, the HSM is installed into the car as first component. Further components might be installed into the car at initialization time. If the HSM requires means of disabling the car or issuing an alarm the HSM exchanges secret keys with crucial components. For instance, the HSM might agree on a key with the car's engine and the car's dashboard such that it is able to disable the car or to display warning messages. Note that for this purpose a separate key is shared between the HSM and the component that is only used for this purpose.

The HSM now randomly chooses a key $K$ that becomes the vehicle key. Assuming that the car is assembled in a trustworthy environment such as a manufacturer plant the vehicle key $K$ is now distributed to all installed components. If there is no trustworthy environment available then for each installed component the installation procedure can be executed as described below.

If the component tags are equipped with computationally strong CPUs a traditional key management for the formation of ad-hoc networks can be used. For instance, a group key-agreement scheme based on the components' certificates could be used for a higher security level at higher computational costs [4]. For the devices we envision here we believe such a scheme to be too resource demanding, though.

## 2.2 Installation of a Component

A new component is installed by adding it to an already existing set of components that form the car. Components are added stepwise. Once a new component is added to the car the installation phase is executed. It consists of the following steps:

- *key check:* check whether the component is already part of the vehicle, e.g., after the component was disassembled for repairing it.
- *proof of origin:* check whether the component has a valid certificate.
- *key initialization:* providing a valid component with the vehicle key $K$.

The key check runs as presented in Figure 2. First, the newly installed component $C$ provides its unique $ID$. The HSM checks whether $ID \in \mathcal{UL}$. This holds if $C$ was part of this car before. If so, the HSM checks if the new component

Figure 2: Key check

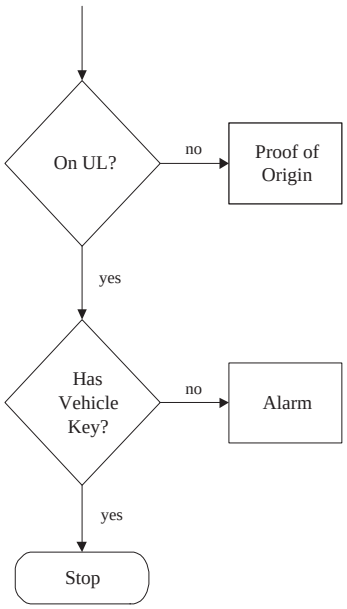
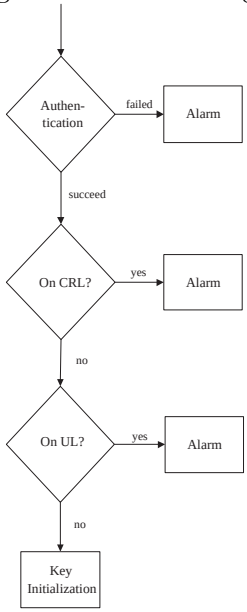



Figure 3: Proof of origin




knows $K$ by a challenge-response authentication. Otherwise, if the component is new, the proof of origin check is performed.

During the proof of origin, the HSM checks whether the new component provides a valid certificate. The HSM performs the steps as presented in Algorithm 3 and depicted in Figure 3 for a new component $C$.

**Algorithm 3** Proof of origin

1: The HSM verifies the certificate $\langle PK, ID\rangle$ and checks whether $C$ knows the secret key $SK$ by a challenge-response method.
2: It checks the $\mathcal{CRL}$ for $\langle PK, ID\rangle$.
3: It puts $C$ on $\mathcal{UL}$ with a preliminary flag.
4: After $\mathcal{UL}$ was synchronized with $\mathcal{GL}$ the preliminary flag of $C$ is deleted. If $C$ was already on $\mathcal{GL}$, an alarm is raised.

In the last two steps, $C$ is put on $\mathcal{UL}$ in a preliminary manner. If $C$ is on the global built-in list $\mathcal{GL}$ then it is already built into another vehicle which indicates that this component was cloned or stolen.

Finally, after all checks were performed successfully, $C$ is provided with the vehicle key $K$ during the key initialization. For that reason, the HSM sends $E(K, PK)$, which is the encryption of $K$ by the key $PK$, to $C$.

## 2.3 The Running System

We ensured that all components that are installed into the car were verified. As trustworthy components they were initialized with the vehicle key $K$. However, after they are installed they might be manipulated, exchanged, or removed. Thus, we execute the system check in order to verify whether all components know the vehicle key $K$. The system check can be executed several times:

- every time the car is started
- periodically, e.g., every 30 minutes
- initiated manually, e.g., to prove system integrity to a policeman

There are basically two methods to execute the system check. In the first version, which is depicted in Figure 4, the HSM challenges all components one after the other by a challenge-response method to check the components' knowledge of $K$. Clearly, only components that are listed in $\mathcal{UL}$ are involved.

In the second method, which is depicted in Figure 5, each component of the vehicle checks

Figure 4: System check with one verifier

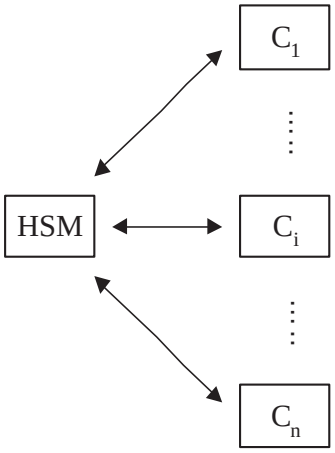
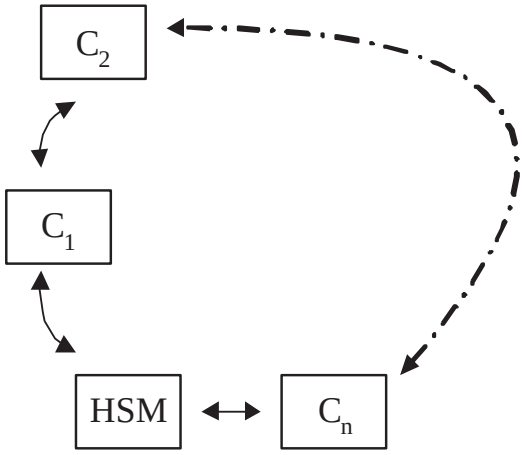



Figure 5: System check with cyclic verifier




another component. The HSM starts by challenging another component, which in turn challenges another component, and so on, until the security module is challenged by a component such that the circle is closed. Here, each component holds the list $\mathcal{UL}$ of all installed components, and all components have to agree on an order of being challenged. If any check fails, the challenging component raises an alarm by sending a message to the HSM which then takes appropriate actions. Note that if the communication bus supports parallel communication, the system check can be performed in parallel. Obviously, it is easy to prevent an attacker from jamming the communication channel in order to delete the alarm message by letting each component either send a positive or negative alarm message. By using a challenge-response method, replay attacks are prevented.

Clearly the system checks are vulnerable to compromised components. A component might always fake an authentication since all components use the same shared key $K$. Furthermore, in the cyclic version a component might "shortcut" an authentication process by skipping components in the cycle. However, we assume here that all components that have knowledge of the system key $K$ play fair. Only components a certificate was issued for are able to obtain the vehicle key $K$. Hence, the certification process is used in order to avoid malicious components in the system. Clearly, if $K$ is known the system can be considered to be broken. The check can also be performed based on the component's certificates such as described in Algorithm 1. Such a check is not vulnerable to an insider attack where the insider has knowledge of the system key $K$. However, such a check based on asymmetric cryptography is in many cases too demanding for the considered device class.

### 2.4 Demounting of a Component

The demounting procedure has to be performed once a component is removed from a vehicle in order to install it into another vehicle. If a component is removed and then installed again into the same car, nothing needs to be done. Hence, the demounting procedure is performed to be able to distinguish between legal demounting and theft of a component. The demounting of a component $C$ works as presented in Algorithm 4.

**Algorithm 4** Demounting

1: The HSM checks whether $C$ knows the secret key $K$ by a challenge-response method.
2: Then the HSM deletes $C$ of the $\mathcal{UL}$.
3: At the next synchronization of $\mathcal{UL}$, $C$ is also deleted of the global list $\mathcal{GL}$.

To remove $C$ in a controlled way, it first has to prove that it knows the secret vehicle key $K$. Then $C$ is deleted of the system and of the built-in lists $\mathcal{UL}$ and $\mathcal{GL}$ such that it appears like a new

component. Components which are bound to a specific car and thus must not be demounted can be marked as such in their certificate, the $\mathcal{UL}$, and the $\mathcal{GL}$ by a flag. The HSM will then not allow these components to be demounted of the vehicle.

In cases where there is no $\mathcal{GL}$, a component stores a flag in its memory which indicates whether the component is installed in a car. When the component is demounted, the flag is deleted. Components can only be installed into a car if the flag is not set. Such a solution is also preferable in cases where the synchronization of $\mathcal{UL}$ and $\mathcal{GL}$ is only performed rarely. For the case that a component is mechanically or electronically broken, this component can be removed from the system by using a system PIN such that a dedicated component performs the demounting process.

# 3 Distributed Component Identification

In the previous approach we assumed that there is an HSM in each vehicle. As the HSM is a central point of failure, it probably will be the first point of attack. More realistic and cheaper is a tamper evident module. Here, a manipulation of the device cannot be avoided but detected. However, manipulation cannot be detected in an automatic way but only by individual inspection. We now extend above scheme such that the role of the HSM is distributed to all components, meaning to all component tags. As before, a tag is embedded to each component. The tags must be able to verify certificates and perform public-key operations.

We now introduce supporting procedures for our system in order to distribute the HSM's task.

**Distribution of $\mathcal{UL}$:** As said before, the HSM holds a list $\mathcal{UL}$ of built-in components. This list is now distributed to all components. In the easiest case, each component holds a copy of the $\mathcal{UL}$. Once a component changes its $\mathcal{UL}$, it broadcasts the new version to all other components. To ensure integrity of the broadcast and in order to prevent replay attacks, we perform Algorithm 5. Here, $i$ is an incremented counter.

**Algorithm 5** List distribution

1: A component $C$ increments $i$ and computes $U := [E := E(\mathcal{UL}||i, K), M := MAC(E, K)]$.
2: $C$ broadcasts $U$.
3: A receiver $R$ only accepts $U$ if $M$ verifies correctly and if $i$ is larger than its stored counter $i'$
4: $R$ stores $\mathcal{UL}$ and updates $i' \leftarrow i$.

**Selection of a Random Component:** Furthermore, we need a mechanism to choose a component randomly out of all vehicle components. As said before, we assume that all components that know the vehicle key $K$ are trustworthy. First, all components have to choose a random number $x$. This works as presented in Algorithm 6.

**Algorithm 6** Selection of a random number

1: Each component $C_i$ chooses a random number $r_i$ and broadcasts $E := E(r_i||ID_i, K)$ as well as $M := MAC(E, K)$.
2: All components check whether they got input from all components and verify the correctness.
3: Each component computes $x := h(\sum_i r_i)$.

Clearly, this scheme is vulnerable to a malicious component that knows the vehicle key $K$ because such a component could wait for its output until it received input of all other components. It is not vulnerable to an external attack though. An unconditionally secure version of choosing a random number is based on commitment schemes which works as presented in Algorithm 7. Assume that $m$ is an RSA modulus, and $e$ and $d$ are the RSA public and private key, respectively. Then a simple commitment function that commits to a value $x$ under a key $d$ is $O := commit(x, d) = x^e \bmod m$. Intuitively, the committed value $x$ cannot be changed since $O$ is given, but it can easily checked once $d$ is published.

Now, all components can choose a random component as presented in Algorithm 8. All components know which component acts as verifier, and only accept a new $\mathcal{UL}$ of the selected

**Algorithm 7** Selection of a random number by commitments

1: Each component $C_i$ chooses a random number $r_i$ and a random key $K_i$, commits to it as $O := commit(r_i||ID_i, K_i)$ by key $K_i$, and broadcasts $O$.
2: After all components broadcasted their commitment, each component opens its commitment by broadcasting $r_i$ and $K_i$.
3: All components verify the commitments and compute $x := h(\sum_i r_i)$.

component.

**Algorithm 8** Selection of a random component

1: All components together select a random number $x$.
2: Each component $C_i$ computes $v_i := h(ID_i||x)$ where $h$ is a hash function.
3: The component $C_i$ with the smallest value $v_i$ acts as verifier. As the IDs are publicly known, all components can verify the selection.

**Raising an Alarm:** Finally, we need an efficient scheme for raising an alarm in a distributed manner. This can be done as follows. Each component that performs a check has to broadcast its result, i.e., a positive or a negative result message. If the result is negative, each component takes action. In the easiest case, a component stops its operation. In order to prevent that an adversary jams the communication channel to prevent broadcasts of negative results, we introduce a time-out of result messages. If a time-out occurs, a component assumes a negative result and takes action. To prevent failures, the time-out value can be chosen very large, or a component can be asked to send its result again.

After designing these auxiliary protocols, we now consider the life cycle phases of a component. These follow in a straightforward manner by combining the phases installation (Section 2.2), running system (Section 2.3), and demounting (Section 2.4) with the auxiliary protocols above.

**Installation**: For the installation of a component $C$, there has to be a component acting as verifier $S$ instead of the HSM. This component is chosen randomly out of all vehicle components using Algorithm 8. Now, $S$ performs the installation procedure, updates $\mathcal{UL}$, and broadcasts it by using Algorithm 5.

**Running System**: For the system check, a random component is chosen by Algorithm 8 to initialize the check. If any check fails, a distributed alarm is raised as described above. If the cyclic system check is performed, all components perform the check in parallel.

**Demounting**: For the demounting of a component $C$, a random component $S$ is chosen by Algorithm 8 to act as verifier. Now, $S$ performs the demounting operation, updates $\mathcal{UL}$, and broadcasts it by using Algorithm 5.

# 4 Symmetric Component Identification

After presenting an approach based on asymmetric cryptography we now present an approach based on symmetric cryptography only. Such an approach is especially useful if we assume that the security tags are resource limited. In particular, the energy consumption as well as computational power might be limited. In the optimal case, RFID (Radio Frequency Identification) tags are used to implement the security tags. These do not require a battery but are powered by the radio waves such that the security tag's lifespan is not limited by its power resources. However, such low-power devices only have extremely limited computational resources just enough to perform basic symmetric cryptographic methods.

For this approach we modify our assumptions as follows:

1. There is an HSM in each car. The HSM acts as a gateway between component tags and a central server $S$.

2. The HSM is temporarily connected to a central server $S$.

3. The HSM and $S$ share a symmetric key $K_{HSM}$.

4. A security tag, e.g. an RFID, is embedded into each component.

5. The security tags as well as the HSM are able to perform basic symmetric cryptography such as computing a MAC but no asymmetric cryptography.

6. Each component $C_i$ is equipped with a symmetric key $K_i$. The central server $S$ holds a list $\mathcal{KL}$ of all symmetric keys $K_i$.

7. Each car holds a system key $K$.

Hence, all challenge-response authentication processes are performed by a symmetric scheme as described in Algorithm 2. We base the symmetric approach on an HSM inside of each automobile. The HSM is temporarily connected to a central server such that data can be exchanged whenever an on-line connection is available. The HSM only needs to perform symmetric cryptography but no asymmetric one. Note that the HSM mainly acts as gateway between the car's components and the central server such that the HSM can be a dedicated component tag. Hence, this approach is based on low-cost RFID tags only and does not require any expensive devices. We now consider the life cycle of a component.

### 4.1 Installation of a New Component

As before, a new component is installed to an already existing set of components that form the car. Again, we run through the following phases:

- *key check*
- *proof of origin*
- *key initialization*

The key check to check whether the new component already was part of the car in the past is executed as presented in Section 2.2. The proof of origin and key initialization need to be adapted though. During the proof of origin, the HSM first sends a challenge $r$ to the new component $C_i$ to be signed by the new component. $C_i$ computes the MAC over $r$ by its secret key $K_i$ and sends back $M := MAC(r, K_i)$ to the HSM. The HSM accepts $C_i$ in a preliminary manner and waits for the next available on-line connection to the server. The HSM then forwards $M$ to the central server together with $r$ and the system key $K$. The server has a list of all components' secret keys $K_i$. If it can successfully verify the new component's response, it sends the new component the encrypted system key $K$. The server $S$ also checks whether $C_i$ was already build into another car and thus is a clone. If any check fails in the algorithm, an alarm is issued and countermeasures are taken as described above. These steps are presented in Algorithm 9.

**Algorithm 9** Proof of origin for the symmetric case

1: The HSM sends a challenge $r$ to $C_i$.
2: $C_i$ computes $M := MAC(r, K_i)$ and sends $M$ to HSM.
3: The HSM puts $C_i$ on $\mathcal{UL}$ with a preliminary flag.
4: After a connection between the HSM and $S$ is established, the HSM sends $Enc(M||r||K, K_{HSM})$ to the server $S$.
5: $S$ checks whether $C_i$ is on $\mathcal{GL}$.
6: If the check is negative, $S$ obtains $K_i$ for $C_i$ from $\mathcal{KL}$ and verifies whether $MAC(r, K_i) \stackrel{?}{=} M$.
7: If the verification is successful, $S$ computes $E := Enc(K, K_i)$, puts $C_i$ on $\mathcal{GL}$, and sends $E$ to HSM.
8: The HSM forwards $E$ to $C_i$ and deletes the preliminary flag.
9: $C$ and the HSM now perform a challenge-response authentication to prove knowledge of $K$.

### 4.2 Running System and Demounting of a Component

The system integrity of the running system is performed based on a symmetric cryptographic scheme as presented in Section 2.3. Furthermore, the demounting of a component is performed in the same manner as presented in Section 2.4. Hence, no adaption of above algorithms is necessary for the running system and the demounting phase.

## 5 Features and Enhancements

Our security scheme provides piracy protection, system protection, as well as theft protection. We now consider features we can implement additionally.

**Policies and Priorities:** Each component certificate can include a string about its properties and limitations. For instance, this information might include a flag whether the component is allowed to be removed of a given car, or if a replacement must be installed. Furthermore, it might include what action has to be taken if the component does not act in an appropriate way or if it is missing. For instance, the other components might stop operating if the airbag is missing, or they might display a warning message if the car radio is missing. This additional information is stored in the $\mathcal{UL}$ shared by all components.

**Controller Units:** If the component is already equipped with a micro-controller, the security tag can be combined with the component. For instance, consider a mileage counter. These are often manipulated in order to increase the value of used cars. The manipulation can be prevented by restricting access to the mileage counter to those components that know the vehicle key $K$. Here, it is possible to lower the requirement for a tamper resistant device to a trusted computing platform at the same security level. Such trusted computing solutions are able to protect the microprocessor against manipulation and thus provide a sufficient level of security.

**Component History:** This feature provides the opportunity to distinguish between new and used components. We present two solutions. First, a component holds an initial key which can only be used once. Each new component holds a certificate $\mathcal{Z} := \langle PK, ID \rangle$ as well as another certificate $\langle PK_{new}, \mathcal{Z} \rangle$ and the secret key $SK_{new}$. When a component is installed for the first time, it proves knowledge of $SK_{new}$. The security tag then deletes $SK_{new}$ from its memory once the component was successfully installed. If the same component is installed into another car, it cannot prove knowledge of $SK_{new}$ anymore. The second variant is based on a central directory of all components. This directory records all information about the component including installation and demounting date. It is straightforward to implement this directory based on $\mathcal{UL}$ and $\mathcal{GL}$. Clearly, this solution only works if there is a communication channel available to synchronize $\mathcal{UL}$ and $\mathcal{GL}$.

**Key Hierarchies:** In some applications there will be several groups of different interest and authorizations. There is the owner, the mechanics of independent and authorized garages, the suppliers of components, and the car manufacturers. The different levels of authorization can be implemented by the use of different keys to enable different levels of rights. For instance, each owner is provided with a smart card that allows him to replace parts of the car that are not relevant for safety. Whenever the owner wants to install or dismount a component he has to attach the smart card to the car. Whenever the owner gives the car to a garage for fixing it, the mechanics use another smart card for replacing components including safety relevant ones. Clearly, there must be no master-card with omnipotent authorization. Even when using the smart card with the highest authorization level, the car performs all security checks and raises an alarm if necessary.

**Subgroups:** We could divide all components into subgroups providing different levels of security. Each subgroup holds its own system key. The higher the security level and safety relevance of a group, the better the protection of the assigned key, for example implemented by tamper resistant features. If a key is compromised, only the key of one subgroup is compromised. Each subgroup performs the system check independent of the other subgroups. Communication among subgroups requires additional keys that are provided at initialization time.

## 6 Security

We based the security of our scheme on the tamper resistance of the HSM as well as the security tags. If these assumptions hold, our system is obviously secure since all involved parties play fair. However, what happens if a key is compro-

mised?

If an adversary Mallory compromises the vehicle key $K$, he could equip counterfeits with $K$ and install them into the car. However, this attack will only work if the counterfeits are listed on the list of built-in components $\mathcal{UL}$ that is stored by the HSM or distributed to all components. Hence, Mallory has to compromise the HSM which is assumed to be especially protected. In a distributed solution, Mallory could compromise a component and force a broadcast of the modified $\mathcal{UL}$ list. However, the broadcast will only be accepted by the remaining components after an installation or demounting procedure, and the new $\mathcal{UL}$ must origin from the component that was randomly selected as verifier for the installation or demounting procedure. If the random selection is based on commitments, Mallory can only add a new component to $\mathcal{UL}$ if his compromised component is chosen as verifier.

If Mallory is able to gain knowledge of a certificate $\langle PK, ID \rangle$ as well as the secret key $SK$, he equips counterfeits with this certificate. He then installs his counterfeits in a car to identify successfully. However, Mallory could not use the same certificate twice for the same car since the HSM would detect a collision. Otherwise, this attack is basically a cloning attack. It can only be avoided when there is a global directory of all built-in lists $\mathcal{GL}$ available as described before.

Mallory might replace the public key of the certificate issuer $PK_{CA}$ with his own public key $PK_M$. Then he is able to issue certificates for his counterfeits. These perform the proof of origin successfully. However, Mallory had to replace the public key of the HSM. In a distributed solution, he had to replace the public key of components. As described before, Mallory cannot influence which component acts as verifier though.

One might think that by time special hacker components would be available that register to the system and then output the secret key $K$. However, as already mentioned the process of certifying components can be used to avoid such malicious components. Hence, only trustworthy components that were evaluated obtain a valid certificate.

Obviously, if $SK_{CA}$ is compromised, the security of the entire system is endangered. However, there are several techniques known how to securely store the secret key of the certificate authority.

# 7 Conclusions

We presented a scheme for implementing component identification in complex systems assembled of several components based on cryptographic authentication schemes. Our solution is mainly based on the tamper resistance of special security hardware modules. In cases where the component has a mechanic main function, e.g. an engine, it is to be researched how secure hardware modules can be embedded into the component. If the component has mainly an electronic function such as an embedded device, the security tag can take over the role of the component. In such cases, trusted computing mechanisms are able to provide tamper resistance. Since more and more devices are embedded with electronics, using such a solution will provide the required means for our component identification scheme.

# References


[1] R. Anderson and M. Kuhn. Tamper Resistance - a Cautionary Note. In *Second Usenix Workshop on Electronic Commerce*, pages 1–11, November 1996.

[2] Business Week, W. Stern. Warning! Bogus parts have turned up commercial jets. Where's the FAA?, June 1996.

[3] IHK Stuttgart. Wirtschaftsrecht: IHK setzt sich aktiv fr die Bekämpfung von Produkt- und Markenpiraterie ein - Jährlich 250 Milliarden Euro Schaden durch Produktfälschung weltweit. World Wide Web, 2004. http://www.stuttgart.ihk24.de.

[4] M. Steiner, G. Tsudik, and M. Waidner. CLIQUES: A New Approach to Group Key Agreement. In *Proceedings of the 18th International Conference on Distributed Computing Systems (ICDCS'98)*, pages 380–387, Amsterdam, 1998. IEEE Computer Society Press.