Secure Role and Rights Management for Automotive Access and Feature Activation

Christian Plappert
christian.plappert@sit.fraunhofer.de
Fraunhofer SIT
Darmstadt, Germany

Lukas Jäger
lukas.jaeger@sit.fraunhofer.de
Fraunhofer SIT
Darmstadt, Germany

Andreas Fuchs
andreas.fuchs@sit.fraunhofer.de
Fraunhofer SIT
Darmstadt, Germany

ABSTRACT

The trend towards fully autonomous vehicles changes the concept of car ownership drastically. Purchasing a personal car becomes obsolete. Thus, business models related to feature activation are gaining even higher importance for car manufacturers in order to retain their customers. Various recent security incidents demonstrated however that vehicles are a valuable attack goal ranging from illegal access to car features to the theft of the whole vehicles.

In this paper, we present a secure access and feature activation system for automotive scenarios that uses a TPM 2.0 as trust anchor within the vehicle to mitigate potential security threats. Our system enables a fine-granular authorization mechanism by utilizing TPM 2.0 enhanced authorization constructs to implement usage restrictions and revocation policies as well as offline rights delegation. The TPM 2.0 acts as a communication end point to the vehicles' environment and integrates seamlessly with already deployed security features of the in-vehicle network.

We implemented our concept on a Raspberry Pi as a lightweight equivalent to hardware used in the automotive domain and evaluate our solution by performance measurements.

CCS CONCEPTS

• **Security and privacy** → *Access control*; **Trusted computing**; *Security protocols*; Domain-specific security and privacy architectures.

KEYWORDS

access control; automotive; feature activation; trusted platform module (TPM)

ACM Reference Format:

Christian Plappert, Lukas Jäger, and Andreas Fuchs. 2021. Secure Role and Rights Management for Automotive Access and Feature Activation. In *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security (ASIA CCS '21)*, June 7–11, 2021, Hong Kong, Hong Kong. ACM, New York, NY, USA, 15 pages. <https://doi.org/10.1145/3433210.3437521>

1 INTRODUCTION

With the advance of fully autonomous vehicles, the transportation landscape as known today will change drastically. The concept

of car ownership is predicted to become less important and will be replaced by a network of autonomous electric cars owned by ride-share companies [1]. For a few years, these so-called robot taxis have already been tested in the field [4, 7, 44].

For car companies, business models such as feature activation will become even more important to improve both customer loyalty and allow customers to keep obtained features across different cars. In other words, comfort features such as navigation or infotainment options are no longer bound to a specific car but rather to the user. As soon as the user successfully authenticates with any car, the corresponding personal features are activated.

The by now omnipresent smartphones with their various interfaces have already proven suitable for today's car sharing and taxi services like Uber [43]. Their mobile internet connections enable the registration with various service providers and combined with GPS allow them to search the surroundings for available car sharing offers or to transmit their current location and destination to Uber drivers. For robot taxis, the personal smartphone would be an option to interact with the driver-less car to access and enable obtained infotainment features.

At the same time, the emergence of autonomous vehicles is further accelerating the trend towards fully connected cars. However, as vehicles become more and more connected also their attack surface increases making them a worthwhile target for cyber criminals. In case of a Tesla, the weak cipher used in the vehicle key signal to unlock the car and deactivate its immobilizer could easily be cracked resulting in the ability of stealing the whole car within seconds [27]. Besides attacks on the vehicle's sensors [28], attacks on the in-vehicle network were also executed. They enable the reprogramming of a car's infotainment system, disabling of airbags, tampering with locking systems, and finally stealing the vehicle [20, 21]. Other attacks could even be executed remotely, modifying the infotainment system [42], allowing steering the car while driving [6, 15, 22] and to locate, unlock and remotely start the car [26, 35].

Consequences of these security attacks do not only harm business models and cause monetary losses in millions, especially if a whole vehicle series or even brand is affected, but also can result in physical harm and are in the worst case threatening lives.

Contribution and Outline. To mitigate threats in the automotive context, we propose a novel secure role and rights management system which considers the security requirements for cars in the field and enables a secure access and feature activation mechanism. Its functional features include both fine-grained online and offline delegation of usage rights and roles as well as an online revocation feature. The keys which enable certain features are securely stored within the Trusted Platform Module 2.0 (TPM 2.0) and their usage

This work is licensed under a Creative Commons Attribution International 4.0 License.

ASIA CCS '21, June 7–11, 2021, Hong Kong, Hong Kong.

© 2021 Copyright held by the owner/author(s).

ACM ISBN 978-1-4503-8287-8/21/06.

<https://doi.org/10.1145/3433210.3437521>

is bound to a TPM 2.0-inherent access policy mechanism called Enhanced Authorization. With this system, we achieve hardware-level security guarantees while maintaining a high level of flexibility.

In particular, we make the following contributions.

- We define general use cases that cover the majority of feature activation scenarios in the automotive domain. These include data confidentiality and IP protection as well as (mutual) authentication of both local and remote Electronic Control Units (ECUs) within the car network and with backend servers. From these use cases we derive functional and security requirements for a novel feature activation and car access system. To the best of our knowledge, currently no similar system exists combining all of these functional features while providing a similar level of security.
- We design our automotive feature activation system with a TPM 2.0 as trust anchor within the car to fulfill the identified goals. Moreover, we map our requirements to TPM 2.0-inherent commands and features. Specifically, the TPM 2.0's Enhanced Authorization mechanism is utilized to map our fine-granular access policy concept. The resulting concepts and protocols are detailed and the fulfillment of the previously defined requirements are adequately evaluated.
- We implement a proof-of-concept prototype for performance evaluation. We use a Raspberry Pi as ECU equivalent, since it has similar capabilities to current automotive head units (c.f., Section 5), and an Android smartphone as authentication token. Based on this prototype and distinct evaluation policies we report performance measurements for authentication and access control.

In our system, the TPM 2.0 acts as a trust anchor within the car which withstands software attacks and to some extent even hardware attacks. The only trusted components are the TPM 2.0 and the remote ECU inside the car as well as the user's authentication token (e.g., smartphone) and the backend server issuing access policies. These trusted components communicate via secured end-to-end channels over the internet or the internal automotive bus systems to prevent third parties from gaining unauthorized access to the vehicle or its features. Inside the car, the TPM 2.0 acts as end point for all internal and external communication. All keys used for the in-vehicle network may remain symmetric. The existing protocols and architecture remain untouched since only the keys used in the protocols are now securely stored and accessed within the TPM 2.0.

Our concept is applicable to general car sharing or fleet management scenarios where it may also be used to authenticate immobilizers of non-autonomous vehicles to start the engine.

The remainder of this paper is structured as follows: Section 2 presents background information and related work while Section 3 describes the designed system and protocols. Section 4 elaborates on the system's security and Section 5 provides information about the prototypical implementation and the evaluation results. Finally, Section 6 concludes the paper and shortly elaborates on future work.

2 BACKGROUND AND RELATED WORK

This section shortly describes related mechanisms and technologies our work is based on and sketches relevant properties of the automotive environment which is the application domain of our solution. We also present some related work dealing with access control systems and hardware-based trust anchors in the automotive domain.

2.1 Policy-based Access Control

The general purpose of access control systems is to protect valuable objects like data, devices or applications from unauthorized operations. Object owners can define access restrictions that subjects wanting to access certain objects need to fulfill. These are enforced by access control mechanisms which can be described with various access control models. Some of them are described in [25, 32, 33].

With the occurrence of Service Oriented Architecture (SOA), OASIS standards body published the eXtensible Access Control Markup Language (XACML) specification [24] which is the basis for the Attribute Based Access Control (ABAC) model. In contrast to more static models like an Access Control List (ACL) or Role-Based Access Control (RBAC) where access is granted based solely on the identified subject, the ABAC model is much more flexible and allows authorization by evaluating attributes associated with subject, object and environmental conditions against a defined policy.

The reference architecture of XACML consists of one or more policy information (PIP), decision (PDP), and enforcement (PEP) points. A basic information flow consists of the following steps. The PEP intercepts a subject's resource access request and forwards it to the PDP which evaluates the authorization request against its configured policies and may retrieve additional information from the PIP to reach a decision. The result is transmitted back to PEP that depending on the result will grant access to the requested resource.

2.2 TPM 2.0 and its Enhanced Authorization concept

The Trusted Platform Module 2.0 (TPM 2.0¹) is a security co-processor specified by the Trusted Computing Group (TCG) [38]. Generally speaking, it is a Trusted Execution Environment (TEE) featuring some advanced capabilities going beyond traditional key management functionality and provides tamper-proof cryptographic key storage which secures keys against all software attacks and to some extent even hardware attacks.

Its non-volatile memory is used to persist data, e.g., monotonic counters or keys, across boot cycles. A specific non-volatile memory region is dedicated to the Platform Configuration Registers (PCRs) that are used to securely store the platform's integrity measurements, e.g., for Trusted Computing concepts like measured boot and remote attestation.

Key usage within the TPM may be secured with three different authorization mechanisms: passwords, Hash Message Authentication Codes (HMACs), or policy sessions. The latter is also called

¹For the sake of brevity, we refer to TPM 2.0 as TPM in the following.

Enhanced Authorization and allows key access by successfully processing a TPM policy. During authorization, the TPM will process presented policy commands sequentially and update an internal policy session digest with each processed policy. If the final digest matches the authentication value of the key, key usage is granted. Currently, 20 different combinable policy commands exist. A special policy command is TPM2_PolicyAuthorize which links key access to a public key and allows key access to any retroactively created policy signed with the corresponding private key.

2.3 Automotive Domain

The in-vehicle network of today's cars consists of up to a hundred small heterogeneous devices called ECUs that are interconnected via distinct bus networks [12]. In most modern vehicle architectures the ECUs are clustered in domains according to their functionality (e.g., drive, comfort, body) and the domains may be connected through one or several Gateways (GWs). Depending on the individual ECU and connected bus system they have strong restrictions on memory and computing power as well as bandwidth. For example, despite the fact that Automotive Ethernet is more and more integrated into modern vehicle architectures [46], CAN-FD with a payload size of 64 bytes per frame is still widely used in sub networks to connect more constrained ECUs.

With the exception of a few more powerful ECUs, e.g., the Telematic Control Unit (TCU) which is responsible for all external communication, the majority of ECUs have sparse capabilities.

Therefore, within the in-vehicle network almost no cryptographic schemes are applied to secure data transmission. In case data transmission is cryptographically secured, it is usually done among the more sophisticated ECUs and even then security is mostly limited to message authentication with symmetric keys, e.g., applying a Cipher-based Message Authentication Code (CMAC) to the messages in the Secure Onboard Communication (SecOC) protocol.

2.4 Related Work

Within the academic field, Dmitrienko et al. [10] present a generic access control system with offline delegation and revocation features where access credentials are securely stored on the smartphone. Busold et al. [5] adapt this generic access control system to the automotive use case and design a smartphone-based immobilizer system with underlying security framework to overcome vulnerabilities and shortcomings of current immobilizer solutions.

In [9], Dmitrienko and Plappert propose an access management system for a free-floating car sharing scenario. They focus on alternative Secure Element (SE) implementations for the smartphone using protocols compatible with smartcards and outsource the SE to smartcards, smartwatches, and secure SD cards. In this context, Symeonidis et al. [36] present a decentralized privacy-enhancing protocol for car access provision utilizing multi-party computation.

Additionally, Groza et al. [17] present another car sharing system that uses identity-based signatures for car access. The authors deploy a TPM in the backend that issues respective credentials to the user.

Next to generic access control, there is some work dealing with hardware trust anchors in the vehicle. For example, in their survey on security threats and protection mechanisms in embedded

automotive networks, [34] presents the TPM's trusted boot mechanism to mitigate against a variety of different attacks. In [16] the authors describe results of the European research project Oversee where a secure in-vehicle application and communication platform is developed to share valuable resources and communication periphery within the car for OEM and third-party applications. Here, hardware security anchors like HSMs are implemented to shield security-critical information and measure the integrity of the host system. In [45], the authors present a novel multi-staged vehicular hardware security module that they implement and evaluate on a Xilinx FPGA.

Also some work focus on the TPM Enhanced Authorization concept. For example, Fuchs et al. [13] use Enhanced Authorization to enable secure firmware upgrades which enforce Intellectual Property Rights and Privacy protection of device-resident data. The same authors also utilize enhanced authorization to secure runtime firmware product lines [14]. Thereby, identical image blobs can be distributed among unified hardware architectures and certain firmware features are only activated (decrypted) on a per-device type basis.

Nyman et al. [23] use TPM's Enhanced Authorization in the context of the European Electronic Identification (eID) system, e.g., to authenticate to government e-services. The focus of the system lies in compatibility with the current eID system.

While some of the access control solutions have a similar feature set compared to our system, the authors' focus lies on securing the credentials on smartphone and backend or on securing transmissions using proprietary protocols. Since the focus of our work lies in securing assets on the vehicle and corresponding transmissions, some of the solutions focusing on smartphone security are compatible to our system and may further enhance our security concept as sketched within this work.

3 SYSTEM DESIGN

In this section we describe our system and adversary model, requirements for our system as well as our system concept and protocols.

3.1 System Model

The system model depicted in Figure 1 consists of the following three entities: (1) The backend system $\mathcal{B}$, that manages roles and rights and issues policies. (2) The car C , where access to feature activation keys is enforced based on the presented policies. (3) The user $\mathcal{U}$ carrying an authentication token $\mathcal{T}$, e.g., a smartphone. $\mathcal{T}$ securely stores a signature key that is used to authenticate towards $\mathcal{B}$ and C . $\mathcal{U}$ may delegate access rights both online and offline to another user $\mathcal{D}$.

Each entity is equipped with a trusted component (TC) where security-critical information is stored, e.g., cryptographic keys or revocation indices, and processed, e.g., via cryptographic operations or policy enforcement. Inside C our TC is implemented as TPM. It is embedded in an ECU that we define as Function Controller (FC) which is responsible for all external communication and for enabling features across the car. The FC would be the vehicle's TCU since it is already responsible for external communication and thus has the correct interfaces. Additionally, it is typically more powerful than other ECUs in terms of processing power and available

Figure 1: System Model.

memory, and thus suitable to fulfill additional tasks coming along with our feature activation enhancements.

A typical life cycle consists of $\mathcal{U}$ registering with $\mathcal{B}$ to bind an authentication key to $\mathcal{U}$'s identity. The authentication key is used as secret credential within a two-factor authentication scheme and must be kept confidential, e.g., by storing it inside $\mathcal{T}$'s shielded location. This could be a TEE like TrustZone [2] or, e.g., another SE implementation as proposed by [9]. $\mathcal{B}$ deploys it via a secure out-of-band channel. Once the user is registered with $\mathcal{B}$, the user can obtain various features such as general car access or certain infotainment options. Each feature accessible for the user is bound to a respective key inside the car's TPM and key usage is authorized via a respective policy that is authenticated by a backend key and bound to the user's authentication key.

3.2 Considered Use Cases

For our system we describe three generic use cases that should cover the majority of feature activation scenarios in the automotive domain. These use cases are feature activation of the local ECU (UC1: Local feature activation), feature activation of a remote ECU (UC2: Remote feature activation), and feature activation of an online service (UC3: Online feature activation). Their implications will be described in the following and are illustrated in Figure 1.

UC1: Local feature activation. In this use case features of the FC are activated. The FC is the vehicle's TCU which is responsible for all external communication and is equipped with the TPM. The TCU is typically more powerful than other ECUs in terms of processing power and available memory. A specific instantiation of this use case is the user buying additional map data, e.g., maps of Europe. In our concept all map data would be stored encrypted in the memory of the FC and the user would get access to the respective feature, which in turn unlocks the decryption key inside the FC.

UC2: Remote feature activation. In this use case features of ECUs that are not the FC are activated. These ECUs are connected to the FC through the in-vehicle network such as a Controller Area Network (CAN) bus or Automotive Ethernet. A specific instantiation of this use case could be activating the engine immobilizer and ignition system inside the engine management ECU. In this case, the user authorizes usage of an HMAC key secured by the TPM.

The HMAC key is then used in a challenge-response mechanism to answer a challenge sent by the remote ECU. The remote ECU activates its feature when the challenge is answered correctly.

UC3: Online feature activation. In this use case a secure communication channel between the car and an external party such as the OEM backend is established. The TPM may store the authentication key that is used to establish the Transport Layer Security (TLS) connection. The channel to the OEM backend can be used for several purposes such as securely transmitting updates to the car or synchronizing music playlists.

3.3 Adversary Model

Our adversary model is based on the cyber-physical Dolev-Yao attacker [31] which extends the classical Dolev-Yao attacker [11]. The latter can act as Man-in-the-middle and eavesdrop, intercept, inject, replay, and modify messages transferred on the network but cannot break any cryptographic schemes without knowledge of the corresponding secret or private keys. The former adds capabilities of a physical attacker who can additionally tamper with system components directly, e.g., read or modify security-sensitive memory regions of the FC. However, he cannot access or modify our TCs. Furthermore, we exclude relay and denial of service (DOS) attacks. Relay attacks are not specific to our system but introduced by the short range wireless communication interfaces we rely upon. Once successful mitigation techniques like distance bounding become widely available on off-the-shelf hardware, they can be easily integrated in our system. Also DOS attacks are not specific to our system and can always be executed by attackers with full control over the communication channel. However, disrupting traffic to prevent token download and general feature activation does not gain any monetary benefit for an attacker and is thus unlikely to happen.

3.4 Functional and Non-Functional Requirements

Our system has the following functional (FR) and non-functional (NFR) requirements.

FR1: Rights management. Users in the system may obtain certain rights for a car, e.g., opening the car or use enhanced infotainment

features that will be evaluated during authentication (FR3). Usage rights are bound to exactly one user. Users shall have the right to delegate their access rights to other registered users through the backend system. Especially delegated access rights may have further restrictions, such as time or usage constraints.

FR2: Revocation mechanism. It shall be possible to actively revoke previously issued credentials, e.g., in case of a compromised cryptographic key or an authorization expires due to the end of a contract period.

FR3: Offline Authentication. It shall be possible to access the vehicle and unlock personal features while the system is offline or little to no network access is available, e.g., underground car parks or in remote locations.

FR4: Offline rights delegation. It shall be possible to delegate rights among different users offline without interaction with the backend. Online right delegation is already part of FR1.

NFR1: Adaptation to deployed architecture. A seamless integration of our system with existing in-vehicle architectures shall be possible to interfere as little as possible with the strict requirements in the automotive context.

3.5 Security Requirements

Our main security objective is to prevent unauthorized access to features in the car. Only registered users with a policy issued by $\mathcal{B}$ (or $\mathcal{U}$ in case of offline delegation) and in possession of the corresponding authentication key may authorize feature key usage. To achieve this objective the following security requirements (SR) need to be fulfilled.

SR1: Confidential credentials. It should be ensured that all authentication credentials, e.g., cryptographic keys, are kept confidential on all respective entities. With regard to our adversary model, the credentials shall be stored in trusted components that shield them against software and hardware attacks. At least for entities directly accessible by an adversary, e.g., $\mathcal{T}$ or $\mathcal{C}$, security level guarantees similar to smartcards should be achieved. This also achieves strong isolation between trusted and untrusted components of each entity of the system.

SR2: End-to-end security. To prevent attacks from the classical Dolev-Yao attacker comprising the communication channels either on the wire or from compromised ECUs firmware, we require that transmissions shall be secured between all communication end points. The secured channel shall be terminated only inside the respective trusted components.

SR3: Strong credentials and crypto. All credentials shall have sufficient key strength. All used cryptographic schemes must be implemented using well-established cryptographic primitives and algorithms. Minimum standards are detailed in [3].

3.6 Concept

The TPM securely stores one key per feature per car denoted as K_{Fn}^C . Depending on the use case (Section 3.2) a key may be a decryption, HMAC, or signing key. Each key is created with the policy digest of a TPM2_PolicyAuthorize command as authorization value (c.f., Figure 2). The TPM2_PolicyAuthorize command is bound to certain backend key denoted as K_{Fn}^B . This construction allows to create additional policies after the key creation that can be successfully

verified as long as their digest is signed by the corresponding secret key $K_{Fn,Sec}^B$. Any additional policies may now be modeled to fulfill the functional requirements defined in Section 3.4.

Figure 2: Key with TPM2_PolicyAuthorize as initial authorization value.

Applying the terminology of XACML specification [24], the TPM acts as PEP, PDP and PIP and thus implements ABAC. It intercepts key usage requests as PEP and validates these requests as PDP by processing the presented policy and comparing the resulting digest with the initially configured authorization policy digest (c.f., Figure 2). The presented policy may contain additional information like usage restrictions which may be validated by the TPM as PIP by comparing Non-Volatile (NV) indices or internal clock values.

Modeling additional policies is done by mapping the requirements to TPM-inherent policy constructs as described in the following. A full policy example is detailed in Appendix A.

FR1: Rights management. To ensure that key usage is only granted to the user to whom a user policy was issued to, the policy needs to be bound to the respective user's unique authentication key $K_{Auth,Sec}^U$. This can be done by including the TPM2_PolicySigned command in each user policy. During execution of the policy, TPM2_PolicySigned will ask the authorizing party to sign a challenge. Since only the intended user possesses the right signing key to answer the challenge successfully, the policy is bound to the respective user. This mechanism prevents from successfully authorizing with any illegally obtained policy. Additionally, it can be used to assign users to roles and rights. This assignment is done by the backend creating a policy that activates features of a distinct role, e.g., normal user or repair shop employee, and binds the respective user key to the policy. In case of the repair shop employee, a policy could be issued that gives access to the debugging interface of certain ECUs.

Additionally, FR1 demands the possibility to delegate access rights to other registered users through the backend. This can be accomplished by the user instructing the backend to create a new policy where TPM2_PolicySigned asks for the delegated user's authentication key $K_{Auth,Sec}^D$. The backend will issue this new policy if the user has the right to delegate (c.f., Section 3.7.3).

Lastly, FR1 requires usage constraints. Typical examples of usage constraints are timing restrictions or a predefined number of usages. Timing restrictions can be realized with TPM2_PolicyCounterTimer and usage restrictions with TPM2_PolicyNV.

With the command TPM2_PolicyCounterTimer, the policy can be bound to a timing value within the TPM. The policy is then not valid before or after the specified value. TPM2_PolicyCounterTimer comes with certain limitations regarding the TPM clock that need to be considered in order to design the system. It must be ensured that the TPM clock is securely synchronized with $\mathcal{B}$ to allow calculating an offset that matches a global UTC time validity period. A possible secure synchronization mechanism is sketched out in Section 6, but beyond the focus of this paper.

With TPM2_PolicyNV the policy may be bound to a counter within the TPM's NV memory. Similar to the timing restriction, a

comparison operation can be defined which results in a policy either valid before or after a certain counter value is reached. Since all policies are bound to the TPM of exactly one car, usage constraints are not intended to be migratable across cars.

Figure 3 shows the schemes of the described usage restricted policies for both user $\mathcal{U}$ and delegated user $\mathcal{D}$.

FR2: Revocation mechanism. Revocation can be handled similarly to the usage restriction described above with TPM2_PolicyNV. All policies need to be bound to an individual NV counter per car. To revoke a certain policy, the backend needs to increment the counter in the respective car. All policies issued for this car and bound to the original value are invalid and need to be issued again by the backend. The counter may be securely incremented over an unsecured channel using a policy which is bound to a backend authentication key or within an established TLS channel as described in UC3. Since policies of other users issued for the same car also become invalid, the backend needs to provide them with new policies, e.g., via push/pull mechanisms. To revoke all access rights of a user, the backend needs to increment the counters in all cars the user has access to.

FR3: Offline Authentication. The described concept enables offline authentication. The required policies are once downloaded online through a backend connection and then stored on the user's token $\mathcal{T}$. When the user authenticates with the car all communication is done over short range wireless communication technologies, e.g., Bluetooth (BT) or Near Field Communication (NFC). During the processing of the transmitted policies, the TPM 2.0 enforces the embedded usage and time restrictions and initiates the challenge-response authentication scheme with the authentication key $K_{Auth_Sec}^{\mathcal{U}}$ stored on $\mathcal{T}$ to verify the identity of the user that is trying to authenticate. In the final step, the TPM 2.0 will verify that the processed policy is originating from an authorized issuer by verifying the signature of the policy.

FR4: Offline rights delegation. Next to the straightforward online delegation concept described before, the system additionally requires offline delegation. If a user $\mathcal{U}$ is allowed to delegate access rights, the backend $\mathcal{B}$ can issue an offline delegation policy. Offline delegation policies contain an additional TPM2_PolicyAuthorize which is bound to $\mathcal{U}$'s authentication key $K_{Auth_Sec}^{\mathcal{U}}$. This allows $\mathcal{U}$ to create further sub-policies independently of $\mathcal{B}$. The TPM will later validate the whole policy path consisting on the offline delegation policy and the sub-policy created by the $\mathcal{U}$. This means that restrictions defined by $\mathcal{B}$ may only be further limited by $\mathcal{U}$ but not widened which is a desired feature.

NFR1: Adaptation to deployed architecture. With our concept, the deployed in-vehicle (security) architecture can remain untouched and only the FC, e.g., TCU, needs to be adapted to host a TPM. The TPM secures the respective secret keys that can be accessed via the policy system. In short, it acts as a broker which “translates” the asymmetric policy authorization concept to the symmetric key concept already deployed in the in-vehicle network. Thus, in the resource constraint in-vehicle network, the only overhead is introduced by transmitting additional nonces and HMACs during challenge-response schemes with remote ECUs in UC2 (c.f., Section 3.2).

The fulfillment of the security requirements SR1-SR3 is detailed in Section 4.

3.7 Protocol Specification

This section specifies our developed protocols. Where necessary, the textual description is supplemented by corresponding sequence diagrams.

3.7.1 System Initialization and Key Deployment. During the time-critical manufacturing, each car $\mathcal{C}$ is equipped with a unique authentication key pair $K_{\mathcal{C}}$, where the private key is stored within the TPM and the public key is known to the backend $\mathcal{B}$. This key is used to establish secure end-to-end session-based channels between $\mathcal{B}$ and TPM. These channels may be used to update keys within the TPM outside of the time-critical manufacturing process. All other steps within the manufacturing process are done as usual.

Within a timing uncritical time frame, e.g., when the cars are parked next to the plant waiting to be shipped and distributed, the secured end-to-end channel is used to add additional feature activation keys bound to the authorization policy digest (c.f., Figure 2). As described during Section 3.2, these may be decryption, HMAC, or TLS keys.

3.7.2 User Registration. User registration with $\mathcal{B}$ has to be done once to link $\mathcal{U}$ to a respective public authentication key $K_{Auth_Pub}^{\mathcal{U}}$. Linking may be done in a trusted environment at a service provider of $\mathcal{B}$ where $\mathcal{U}$ proves the possession of the corresponding secret key $K_{Auth_Sec}^{\mathcal{U}}$, e.g., with a challenge-response scheme. The corresponding key must be kept confidential within $\mathcal{T}_{\mathcal{U}}$'s TEE ($TEE_{\mathcal{T}_{\mathcal{U}}}$) and must never leave it. Cryptographic operations are done within $TEE_{\mathcal{T}_{\mathcal{U}}}$. Access to $TEE_{\mathcal{T}_{\mathcal{U}}}$ is typically secured by Personal Identification Number (PIN), password, unlock pattern or by a biometric authentication mechanism, e.g., fingerprint. This enables multi-factor authentication.

After registration is done, $\mathcal{U}$ can use $K_{Auth_Sec}^{\mathcal{U}}$ to authenticate with $\mathcal{B}$, e.g., to buy new features. $\mathcal{B}$ will then create individual policies for $\mathcal{U}$ by selecting a subset of restrictions x from the set of all possible restrictions A . The selected restrictions x from A for $\mathcal{U}$ are denoted as $A|x_{\mathcal{U}}$. The resulting policies then allow only access to the features $\mathcal{U}$ is authorized for.

3.7.3 Policy Download. The policy download protocol is depicted in Figure 5. Within this protocol, $\mathcal{U}$ requests access to a feature N from the backend and retrieves the corresponding policy to activate this feature.

First $\mathcal{U}$ initiates the protocol by selecting the desired function set N within the application of $\mathcal{T}_{\mathcal{U}}$. $\mathcal{T}_{\mathcal{U}}$ forwards the request with the unique UserID to $\mathcal{B}$. Based on the UserID $\mathcal{B}$ selects the public authentication key for $\mathcal{T}_{\mathcal{U}}$ ($K_{Auth_Pub}^{\mathcal{U}}$) to bind the policy via PolicySigned to $\mathcal{U}$ and the corresponding restrictions $A|x_{\mathcal{U}}$. If not currently done and the policy requires a timing restriction, $\mathcal{B}$ must first retrieve the current clock value of the TPM 2.0 with a secure session in order to calculate an offset related to the required validity period. All these information are then used to create a policy template that is then instantiated ($p*N$). During instantiation the policy template is enriched with additional information derived from the TPM 2.0's usage setting, e.g., parameters of the NV memory area or particular algorithms. The digest δ_{p*N} of $p*N$ is then signed with

Figure 3: Schemes of rights management policies. First: User policy with time restriction (valid until \$DATE). Second: Delegated policy with usage restriction (valid for 5 usages).

Figure 4: Scheme of an offline delegation policy for the delegated user consisting of two paths. First: Offline delegation policy. Second: User-defined sub-policy

$\mathcal{B}$'s signature key $K_{Fn_Sec}^B$ to retrieve the signature σ_{p^*N} . Finally, $\mathcal{B}$ creates the final authorized policy p_N that consists of p^*N , σ_{p^*N} and the public key of $\mathcal{B}$ (K_{Fn}^B).

Optionally, if $\mathcal{U}$ has the rights of delegation, an instantiated policy for offline delegation p_{ND} is created from a policy template with the base restrictions of $\mathcal{U}$ ($A|x_{\mathcal{U}}$) and a PolicyAuthorize linked to the authentication key of $\mathcal{U}$ ($K_{Auth_Pub}^U$) that allows $\mathcal{U}$ to later create sub-policies. The digest δ_{p^*ND} is signed to retrieve signature σ_{p^*ND} . Again, the final authorized policy p_{ND} is created consisting of p^*ND , σ_{p^*ND} and the public key of $\mathcal{B}$ (K_{Fn}^B). After the authorized policies (p_N and optionally p_{ND}) are created by $\mathcal{B}$, they are sent back to $\mathcal{T}_{\mathcal{U}}$ and stored.

In case of online delegation, the protocol sequence is similar for $\mathcal{D}$. However, $\mathcal{U}$ must first inform $\mathcal{B}$ that $\mathcal{D}$ is authorized for certain functions with restrictions $A|x_{\mathcal{D}}$. $\mathcal{B}$ will then generate a dedicated policy for $\mathcal{D}$ with $K_{Auth_Pub}^D$ to bind the policy via PolicySigned to $\mathcal{D}$ and $A|x_{\mathcal{D}}$ when $\mathcal{D}$ initiates the protocol.

A policy template with a corresponding authorized instantiated policy is depicted in Appendix B.

3.7.4 Offline Delegation. In case of offline delegation (s. Figure 6), $\mathcal{U}$ needs to have the offline delegation policy p_{ND} . Depending on the implementation, offline delegation is started by $\mathcal{U}$ requesting the public key of $\mathcal{D}$ from $\mathcal{T}_{\mathcal{D}}$. $\mathcal{T}_{\mathcal{D}}$ prompts $\mathcal{D}$ to confirm the delegation request and sends back $K_{Auth_Pub}^D$. $\mathcal{U}$ initiates the creation of a sub-policy template p^*N_{SUB} that is bound to $K_{Auth_Pub}^D$ and has selectable restrictions $A|x_{\mathcal{D}}$. After authorizing signing key usage with an authorization value (AV) like PIN, the digest of p^*N_{SUB} is signed within $TEE_{\mathcal{T}_{\mathcal{U}}}$ and the resulting signature $\sigma_{p^*N_{SUB}}$ is sent $\mathcal{T}_{\mathcal{U}}$.

Similar to Section 3.7.3, $\mathcal{T}_{\mathcal{U}}$ creates the final authorized sub-policy $p_{N_{SUB}}$ that consists of p^*N_{SUB} , $\sigma_{p^*N_{SUB}}$ and the public key of $\mathcal{U}$ ($K_{Auth_Pub}^U$). Finally, p_{ND} and $p_{N_{SUB}}$ are sent to $\mathcal{T}_{\mathcal{D}}$ that stores both policies.

3.7.5 Authentication and Authorization. As described before, authentication and authorization is done fully offline for both $\mathcal{U}$ and $\mathcal{D}$. The protocol is depicted in Figure 7 and described in the following. Since the protocol sequence is almost identically for $\mathcal{U}$ and $\mathcal{D}$, the following example is done for $\mathcal{U}$ and the differences to $\mathcal{D}$ are explained throughout the description. For simplicity we assume that an underlying communication technology is used where the user initiates the protocol, e.g., BT. However, a passive protocol sequence, e.g., in case of NFC is also possible. In this case the protocol needs to be slightly adjusted so that a controller within $\mathcal{C}$ initiates the communication by presenting an ID that $\mathcal{T}_{\mathcal{U}}$ is able to map to the corresponding function.

In the BT use case, $\mathcal{U}$ initiates the authentication process by explicitly stating the function set to use. $\mathcal{T}_{\mathcal{U}}$ will send the corresponding policy path to FC. The policy path may consist of multiple sub-policies. For example in case of an offline delegation policy, $\mathcal{T}_{\mathcal{D}}$ must send both the offline delegation policy signed by the backend (p_{ND}) and the sub-policy signed by $\mathcal{T}_{\mathcal{U}}$ with $K_{Auth_Sec}^U$ ($p_{N_{SUB}}$) to the TPM to verify the whole policy path (c.f., Figure 4).

After receiving the policies, FC will start an policy authentication session with the TPM to sequentially process each policy command within the retrieved policy path. In Figure 7 the TPM2_PolicySigned is the first command in the policy path. During execution of this command, the TPM will generate a nonce (N_{TPM}) which is used in a challenge-response scheme to verify that $\mathcal{T}_{\mathcal{U}}$ is in possession of the right authentication key. This is done by transmitting N_{TPM} back to $\mathcal{T}_{\mathcal{U}}$. Since the required authentication key $K_{Auth_Sec}^U$ is locked within $\mathcal{T}_{\mathcal{U}}$'s $TEE_{\mathcal{T}_{\mathcal{U}}}$, $\mathcal{U}$ is prompted to unlock $K_{Auth_Sec}^U$ with authorization value (AV) defined during user registration (c.f., User Registration). Once unlocked $K_{Auth_Sec}^U$ is used to sign N_{TPM} and the resulting signature $\sigma_{N_{TPM}}$ is sent back to FC that forwards it to the TPM. The TPM will verify the signature and update its policy session digest accordingly.

Figure 5: Policy download protocol.

Figure 6: Offline delegation protocol.

The FC will then start processing the further policy commands in the policy path. However, all other policy commands do not require interaction with $\mathcal{T}_{\mathcal{U}}$ and processing is therefore much simpler. Mostly, all further policies are time or usage restriction policies that the TPM can verify right away.

The last policy command that the FC will process is the command TPM2_PolicyAuthorize. FC will load $K_{\text{Fn_Pub}}^{\mathcal{B}}$ into the TPM and will pass the policy signature σp^*_{N} extracted from p_N . The TPM will first verify the signature and second will compare the signed digest with its own policy session digest. If both values match, authorization is successful and access to the feature key is granted.

4 SECURITY ANALYSIS

Our core security concept relies on our modeled access policies and their evaluation within the TPM. In our system, the TPM acts as trust anchor within the car. It implements a trustworthy PEP, PDP, and PIP by verifying and enforcing the presented policies and

thus enables ABAC. We model our policies in such a way that (1.) only policies originating from an authorized issuer are valid, (2.) presented policies are cryptographically bound to their respective user, (3.) fine-granular authorization restrictions are enforced and (4.) access policies are fully verifiable within the TPM without relying on any probably compromised software on the FC.

The security concept is guaranteed by the fulfillment of the defined security requirements with regard to the previously defined adversary model. The fulfillment of the security requirements is evaluated in the following.

SR1: Confidential credentials. The system is designed that all policies require at least a signature by an authorized backend and the corresponding user. The required backend signature prevents any third party to modify or create arbitrary valid policies without knowledge of the backend signing key $K_{\text{Fn_Sec}}^{\mathcal{B}}$. The required user signature prevents any third party to successfully authenticate with

Figure 7: Authentication protocol for non-delegated user $\mathcal{U}$. Excerpt of policy processing for TPM2_PolicySigned.

any illicitly obtained policy without knowledge of the user signing key $K_{\text{Auth_Sec}}$.

Thus, the security of our system relies on the confidentiality of the following credentials: $K_{\text{Auth_Sec}}$ (with its access credential AV) and $K_{\text{Fn_Sec}}^B$. Additionally any revocation indices and usage counters as well as the actual feature activation keys need to be protected.

While AV is used to authorize access to the TEE of token $\mathcal{T}$, $K_{\text{Auth_Sec}}$ is used on $\mathcal{T}$ to authenticate the user to car C by signing challenges in the TEE of $\mathcal{T}$ during authentication. In case of offline delegation $K_{\text{Auth_Sec}}$ is used to sign sub-policies for the delegated user. The key $K_{\text{Fw_Sec}}^B$ is used by backend $\mathcal{B}$ to sign policies that grant distinct users access to features N for specific cars. In C , the revocation indices are used to constrain the validity of issued policies and the feature activation keys are used to unlock the features described in our use cases.

With the exception of AV which is a pure knowledge factor of user $\mathcal{U}/\mathcal{D}$ and not stored on any entity, confidentiality of all other credentials is ensured by shielding them within hardware-isolated TEEs on the respective entities. In particular, we instantiate both a TPM attached to C 's FC and ARM TrustZone [2] on $\mathcal{T}$. By design, credentials can never leave the TEEs at any point in time and they are specifically hardened against side-channel attacks thus providing physical protection similar to smart cards. In case $\mathcal{T}$ provides lower protection guarantees (e.g., no StrongBox Keymaster support [8] in case of an Android phone), the security on $\mathcal{T}$ could be further improved by incorporating SE instantiation schemes of [9] in our system.

Additionally, the system is designed that the credentials that yield the highest revenue for an attacker (e.g., $K_{\text{FN_Sec}}^B$ whose leakage would allow to create arbitrary policies for feature N) are stored at

B. Since $\mathcal{B}$ has capabilities of a full-blown server infrastructure, the most sophisticated security guarantees can be assumed including distinct TEEs for each credential isolated by structural separation (e.g., air gaps). In contrast, entities like $\mathcal{T}$ or $\mathcal{C}$ can be physically accessed by an attacker, e.g., by stealing a car. However gaining unauthorized access to the respective TEEs is highly expensive in terms of processing power and time and does not correlate to the potential gains which are limited (probably time limited) to access rights for distinct functions of a distinct car.

SR2: End-to-end security. Our system is designed to achieve a small trusting computing base where only TPM, remote ECU, the smartphone’s TEE and backend are considered trust anchors. All our secured channels within the car $\mathcal{C}$ and towards backend $\mathcal{B}$ or token $\mathcal{T}$ terminate within our proposed trust anchors (e.g., TEEs). The secure channels are established using standard security protocols with well-understood security guarantees like TLS (UC3) or challenge-response schemes (UC2) to mitigate attacks from the classical Dolev-Yao attacker. This includes the introduction of appropriate freshness values into the protocols to avoid an attacker simply replaying messages and the usage of Message Authentication Codes (MACs) to secure integrity and authenticity of all messages in transit.

The TPM is the trust anchor within the car and acts as both PEP, PDP and PIP by enforcing access restrictions for features in the car. All security protocols are completely handled within TPM and any unauthorized modification or replay of data in transit would result in an unsuccessful verification of the policy and thus denied access to the key. Since even a compromised host FC cannot gain access to stored credentials inside the TPM, he cannot authorize the respective policy without the authorization key which is securely

Figure 8: Schemes of the basic policies used for performance evaluation. First is a user policy and second is an offline delegation policy consisting of two paths: the backend signed path and the user signed path.

stored on $\mathcal{T}$. Trusted boot or PCR-based policy schemes do not gain a higher security level and can be neglected for our specific use cases. However, future use cases may benefit from binding key usage to the platform state.

SR3: Strong credentials and crypto. The proposed system uses well-established cryptographic primitives and algorithms supported by the TPM; AES-128, RSA-2048, ECDSA, and SHA-256 which have been evaluated and represent the current state-of-the art [3]. The TPM specification is designed to be cryptographically agile so that the deployed algorithms and standards can be easily replaced with more secure versions once the current ones become vulnerable [37, 39]. $\mathcal{U}$'s signature key and the feature activation keys are either stored inside the TEE of $\mathcal{T}$ or inside the TPM and will never be used outside of these trusted environments. The feature activation keys are protected by the defined policies and $\mathcal{U}$'s signature key is additionally protected by a PIN or biometrics.

Table 1: Execution time of the basic user and a basic delegation policy.

User Type	BT Discovery [ms]	Policy Processing [ms]	Overall [ms]
$\mathcal{U}$	142.79 (± 50.28)	1207.64 (± 40.11)	1350.40 (± 63.68)
$\mathcal{D}$	116.12 (± 23.74)	1486.42 (± 35.03)	1602.54 (± 42.84)

5 PROTOTYPE AND EVALUATION

In the following section, we describe the hardware, software and cryptographic primitives we used to implement our prototype and evaluate performance, feasibility and limitations of our system. A picture of our prototype is shown in Appendix C.

5.1 Prototype Hardware and Software

Hardware. For prototyping the concept we use a Raspberry Pi [30] (1.4 GHz, 1 GB RAM) as FC since it has similar capabilities as current automotive head units. For example, the Modular Infotainment Platform (MIB) of Volkswagen contains a Nvidia Tegra 2 Applications Processor clocked at 1.2 GHz with 1 GB DRAM [18]. An Iridium add-on board [29] hosts the TPM and is attached to the Raspberry Pi's Serial Peripheral Interface (SPI) bus pins. The Raspberry Pi runs Raspbian 10 (Buster) with Kernel 5.4.

Two Android smartphones are used as the user tokens. They run Android Version 10. The authentication key is stored in the Android KeyStore [8] that utilizes TrustZone as isolation layer to

the host system. The key is further protected by a PIN. The token communicates to the FC via Bluetooth.

Software. The software on the FC is implemented using C as programming language and the TSS 2.0 Feature API (FAP) [40] is used for communication with the TPM. The Android app on the smartphone is implemented using Java and the signature keys are stored in the TEE of the smartphone utilizing the Android keystore. The policies are encoded using the TSS 2.0 JSON Data Types and Policy Language [41]. Exemplary JSON policy templates are depicted in Appendix B. We enhanced the specification with the two additional data fields *uid* and *features*. The field *uid* is used to identify related policy paths and corresponding keys and *features* allows a user-friendly decoding of features enabled by the corresponding policy.

Primitives and parameter sizes. For signatures used within the two commands TPM2_PolicyAuthorize and TPM2_PolicySigned, P-256 ECDSA is used with SHA-256 as hash algorithm. Depending on the use case (c.f., Section 3.2), usage keys within the TPM are either 128-Bit AES decryption (UC1), 256-Bit HMAC (UC2) or 2048-Bit RSA signature keys (UC3).

5.2 Performance Evaluation

For performance analysis we analyzed the transmitted message sizes and execution times for the basic policies depicted in Figure 8 during authentication.

The first policy is a user policy with a time restriction. It requires a signature by $\mathcal{U}$ and is bound to an NV counter for revocation. The second policy is an offline delegation policy consisting of two policy paths. The first is the offline delegation policy authorizing $\mathcal{U}$ to create own policies. It is bound to an NV counter for revocation. The second policy is the sub-policy for $\mathcal{D}$ created by $\mathcal{U}$ which requires a signature by $\mathcal{D}$ and has a time restriction.

5.2.1 Transmitted Message Sizes. During authentication, the JSON-encoded authorized and instantiated policies (including the policy signatures, digests and public authentication keys) and the (signed) nonces need to be transmitted between $\mathcal{T}$ and FC (c.f., Figure 7).

Table 2 shows the total bytes needed to be transmitted. The table shows the message sizes according to our prototype. They depend on the used cryptographic primitives and parameters as well as the distinct encoding of value names used in the JSON policies. Since we are using ECDSA as signature algorithm we get a varying signature size with a maximum value of 72 byte. For the JSON-encoded policies we removed all unnecessary white spaces.

At minimum, in case of a basic user policy, one policy (including its signature and the PEM-encoded public key) and a nonce and its

Table 2: Transmission sizes for authentication protocol

User	Data (Size [Byte])	Total Size [Byte]
$\mathcal{U}$	p_N (2194), N_{TPM} (36), $\sigma_{N_{\text{TPM}}}$ (72)	2302
$\mathcal{D}$	p_{N_D} (1537), $p_{N_{\text{SUB}}}$ (2194), N_{TPM} (36), $\sigma_{N_{\text{TPM}}}$ (72)	3839

signature needs to be transmitted. In our prototype, this results in a total of 2302 byte in the minimum case.

Additionally, in the offline delegation use case, two policies (each including its signature and the PEM-encoded public key) as well as a nonce and its signature need to be transmitted resulting in a total of 3839 byte.

5.2.2 Execution Time. For performance analysis we measured time required to complete an offline user authentication both of a basic user and a basic delegation policy depicted in Figure 8. Measurements were taken 1000 times and average values as well as standard deviation are reported in Table 1.

For the overall authentication time we separate Bluetooth (BT) discovery from the policy processing. The time difference between user and offline delegation policy can be explained with the second TPM2_PolicyAuthorize that requires a additional signature verification

5.3 Feasibility Evaluation in the Automotive Context and System Limitations

We designed our system with respect to the sparse capabilities of in-vehicle networks and ECUs (c.f., Section 2.3) by defining a dedicated requirement NFR1. We fulfilled this requirement by designing the system that the TPM not only is the trust anchor within the car which evaluates and enforces all presented policies but also acts as “cryptographic broker” for the in-vehicle network. It translates the resource-intensive operations of asymmetric cryptography used between backend $\mathcal{B}$, user token $\mathcal{T}$, and the function controller FC of car C to the more lightweight symmetric cryptography that is appropriate to be used within the in-vehicle network. The broker functionality is relevant in UC2 (c.f., Section 3.2) which covers communication with a resource-constraint remote ECU through the in-vehicle network. This is done by using the policies to unlock symmetric feature activation keys (256-Bit HMAC keys) that are used in challenge-response authentication schemes with remote ECUs within the vehicle network to unlock features. Even within limited CAN-FD bus networks, the challenge and response scheme with 36 byte for the challenge and 72 byte for the response fit into three single 64 byte payload frames. These additional frames per feature generate little additional overhead in the network. Especially, since they need to be exchanged only once during vehicle start. The use of asymmetric cryptography on FC in use cases UC1 and UC3 seems feasible since FC would be part of the vehicle’s TCU which typically is one of the more powerful ECUs in terms of processing power and available memory.

We considered the uncertain environments in which a vehicle may be operated, e.g., areas with little to no network coverage, by designing the crucial authentication and optional delegation

protocol steps to work fully offline. Thus, users may still access the vehicle and corresponding features regardless of their surroundings, e.g., in underground car parks or remote locations.

Regarding the performance of our system, we see that average authentication times of 1,4 to 1,6 seconds seem feasible for a single authentication. However, in a worst case approach, the user would have a policy for every obtained feature which would not scale well. While certain less critical features, e.g., pure infotainment features like play list synchronization, could be delayed until after the vehicle has been started to prioritize more critical features, e.g., automotive access and motor ignition, we decided for a hybrid key store approach where multiple software key stores hold different groups of feature activation keys. The software key stores are encrypted by different keys residing inside the TPM. Thus, one policy unlocks a set of features and the mapping is done in the backend. The organization of the hybrid key store depends on the provider’s business model.

From a hardware point of view we would only add a TPM, e.g., to the TCU of the vehicle, and leave the rest of the vehicle untouched. For our performance evaluation, we used a TPM that is close to its automotive-grade counterpart in terms of features and power consumption (standby of 110 μA and peak at 25 mA) [19]. It only lacks to operate in extended temperature ranges and compliance with some automotive safety standards which is at this point irrelevant for our general feasibility evaluation. In a real-world setting the currently used TPM can easily be substituted with its automotive-grade counterpart without losing any relevance from our feasibility and performance evaluation.

Since we have little interference with the already deployed vehicle architecture (hardware requirement for a TPM per car and three additional messages per feature in the in-vehicle network), the system is easily adaptable to a variety of different vehicle architectures. However, because of the multitude of different architectures across different (and also sometimes within the same) car models, we only describe exemplary features to be activated. However, we defined generic use cases to cover a majority of feature activation scenarios (from sole authentication to encryption) and describe an efficient way to activate a multitude of features without significantly increasing the activation time. In our current design, we do not activate any features during driving which may need to be accompanied by an additional safety assessment. However, this is beyond the focus of this paper.

A limitation comes with the revocation procedure that relies on a counter in the TPM. Since all policies are bound to the same counter value in the respective car, increasing the value would also invalidate the policies of all other users. As described in Section 3.6, this would be no major problem and the backend just needs to provide them with new policies, e.g., via push/pull mechanisms. For future work, more fine granular approaches for revocation are also conceivable either using multiple counters directly in the TPM’s NV memory or a mechanism similar to the proposed hybrid key store that would allow the most flexibility, e.g., with counters for each authenticated user.

6 CONCLUSION AND FUTURE WORK

In this paper we propose and implement a fine-grained feature activation system for automotive scenarios which utilizes TPM 2.0-inherent policy constructs to authorize key usage within the car. The system allows users to delegate their access rights both on- and offline with self-defined restrictions and to actively revoke authorizations via online connection. In this system, the TPM 2.0 is used as the endpoint of a secured communication channel between the backend and the user's authentication token, e.g., smartphone. The authentication of the user combined with the policies that currently apply grants access to symmetric keys of the TPM 2.0. These keys allow for the integration with existing in-vehicle networks and security solutions with minimal integration effort.

Future work include overcoming limitations of the current system like making the revocation system more fine-granular or further hardening the time-restricted policies. For the latter, the TPM 2.0's clock needs to be securely synchronized which could be enforced by binding policies to a counter which allows key usage only when the clock is set. Also the performance could be improved by storing most of the verification keys on the function controller or by sealing multiple function keys to a single policy authorization which changes the performance security ratio. Moreover, NV-Policies can be utilized to enable further applications, such as geo-fence scenarios. An offline revocation system could be implemented by binding counter increments via TPM2_PolicyCpHash to user policies such that revocations are being enforced during regular authorizations. Finally with TPM2_PolicyPCR, policies may additionally be bound to the TPM 2.0's PCR values, granting key usage only when the platform is in an unmodified state and thus protecting from rogue controller firmware.

ACKNOWLEDGMENTS

This research work has been partly funded by the German Federal Ministry of Education and Research (BMBF) and the Hessen State Ministry for Higher Education, Research and the Arts within their joint support of the National Research Center for Applied Cybersecurity ATHENE and by the BMBF project VITAF (ID 16KIS0835). Additionally, the project leading to this application has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 883135.

REFERENCES

- [1] James Arbib and Tony Seba. 2017. Rethinking Transportation 2020-2030. <https://www.rethinkx.com/transportation>. A *RethinkX Sector Disruption Report* (2017). Accessed: 2020-05-08.
- [2] Arm Limited. 2019. Introducing Arm TrustZone. <https://developer.arm.com/technologies/trustzone>.
- [3] BSI. 2018. Technical Guideline – Cryptographic Mechanisms: Recommendations and Key Lengths. <https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TG02102/BSI-TR-02102-1.pdf>.
- [4] Sarah Buhr and Megan Rose Dickey. 2017. Self-driving Ubers are now picking people up in Arizona. <https://techcrunch.com/2017/02/21/self-driving-ubers-are-now-picking-people-up-in-arizona/>. Accessed: 2020-05-08.
- [5] Christoph Busold, Ahmed Taha, Christian Wachsmann, Alexandra Dmitrienko, Hervé Seudié, Majid Sobhani, and Ahmad-Reza Sadeghi. 2013. Smart Keys for Cyber-cars: Secure Smartphone-based NFC-enabled Car Immobilizer. In *Proceedings of the Third ACM Conference on Data and Application Security and Privacy* (San Antonio, Texas, USA) (CODASPY '13). ACM, New York, NY, USA, 233–242. <https://doi.org/10.1145/2435349.2435382>
- [6] Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, Stefan Savage, Karl Koscher, Alexei Czeskis, Franziska Roesner, Tadayoshi Kohno, et al. 2011. Comprehensive experimental analyses of automotive attack surfaces. In *Proceedings of the 20th USENIX Conference on Security* (San Francisco, CA) (SEC'11, Vol. 4). San Francisco, USENIX Association, Berkeley, CA, USA, 447–462. <http://dl.acm.org/citation.cfm?id=2028067.2028073>
- [7] Alex Davies. 2017. GM's Robocar Service Drives Employees Around SF for Free. <https://www.wired.com/story/gm-cruise-anywhere-self-driving-san-francisco/>. Accessed: 2020-05-08.
- [8] Google Developers. 2019. Android keystore system. <https://developer.android.com/training/articles/keystore.html>.
- [9] Alexandra Dmitrienko and Christian Plappert. 2017. Secure Free-Floating Car Sharing for Offline Cars. In *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy* (Scottsdale, Arizona, USA) (CODASPY '17). ACM, New York, NY, USA, 349–360. <https://doi.org/10.1145/3029806.3029807>
- [10] Alexandra Dmitrienko, Ahmad-Reza Sadeghi, Sandeep Tamrakar, and Christian Wachsmann. 2012. SmartTokens: Delegable Access Control with NFC-Enabled Smartphones. In *Proceedings of the 5th International Conference on Trust and Trustworthy Computing* (Vienna, Austria) (TRUST'12). Springer-Verlag, Berlin, Heidelberg, 219–238. https://doi.org/10.1007/978-3-642-30921-2_13
- [11] D. Dolev and A. C. Yao. 1981. On the Security of Public Key Protocols. In *Proceedings of the 22nd Annual Symposium on Foundations of Computer Science (SFCS '81)*. IEEE Computer Society, Washington, DC, USA, 350–357. <https://doi.org/10.1109/SFCS.1981.32>
- [12] Christof Ebert and Capers Jones. 2009. Embedded software: Facts, figures, and future. *Computer* 42, 4 (05 2009), 42–52. <https://doi.org/10.1109/MC.2009.118>
- [13] Andreas Fuchs, Christoph Krauß, and Jürgen Repp. 2016. Advanced Remote Firmware Upgrades Using TPM 2.0. In *ICT Systems Security and Privacy Protection*, Jaap-Henk Hoepman and Stefan Katzenbeisser (Eds.). Springer International Publishing, Cham, 276–289. https://doi.org/10.1007/978-3-319-33630-5_19
- [14] Andreas Fuchs, Christoph Krauß, and Jürgen Repp. 2017. Runtime Firmware Product Lines Using TPM2.0. In *ICT Systems Security and Privacy Protection*, Sabrina De Capitani di Vimercati and Fabio Martinelli (Eds.). Springer International Publishing, Cham, 248–261. https://doi.org/10.1007/978-3-319-58469-0_17
- [15] Andy Greenberg. 2015. Hackers Remotely Kill a Jeep on the Highway - With Me in It. <https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/>.
- [16] André Groll, Jan Holle, Marko Wolf, and Thomas Wollinger. 2010. Next Generation of Automotive Security: Secure Hardware and Secure Open Platforms.
- [17] Bogdan Groza, Lucian Popa, and Pal-Stefan Murvay. 2019. CarINA - Car Sharing with Identity Based Access Control Re-enforced by TPM. In *Computer Safety, Reliability, and Security*, Alexander Romanovsky, Elena Troubitsyna, Ilir Gashi, Erwin Schoitsch, and Friedemann Bitsch (Eds.). Springer International Publishing, Cham, 210–222.
- [18] IHS Electronics & Media. 2006. Audi MIB High (8V0035021 - MIB Processing Unit) Infotainment Teardown. <https://electronics360.globalspec.com/article/3337/audi-mib-high-8v0035021-mib-processing-unit-infotainment-teardown>.
- [19] Infineon Technologies AG. 2018. OPTIGA TPM SLB 9670 TPM2.0. https://www.infineon.com/dgdl/Infineon-SLB%209670VQ2.0-DataSheet-v01_00-EN.pdf?fileId=5546d4626fc1ce0b016fc78270350cd6.
- [20] Karl Koscher, Alexei Czeskis, Franziska Roesner, Shwetak Patel, Tadayoshi Kohno, Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, et al. 2010. Experimental security analysis of a modern automobile. In *2010 IEEE Symposium on Security and Privacy*. IEEE, 447–462. <https://doi.org/10.1109/SP.2010.34>
- [21] Federico Maggi. 2017. The Crisis of Connected Cars: When Vulnerabilities Affect the CAN Standard. <https://blog.trendmicro.com/trendlabs-security-intelligence/connected-car-hack/>.
- [22] Charlie Miller and Chris Valasek. 2014. A survey of remote automotive attack surfaces. *Black Hat USA 2014* (2014), 94.

- [23] Thomas Nyman, Jan-Erik Ekberg, and N. Asokan. 2014. Citizen Electronic Identities Using TPM 2.0. In *Proceedings of the 4th International Workshop on Trustworthy Embedded Devices* (Scottsdale, Arizona, USA) (TrustED '14). ACM, New York, NY, USA, 37–48. <https://doi.org/10.1145/2666141.2666146>
- [24] OASIS. 2019. OASIS eXtensible Access Control Markup Language (XACML) TC. https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml
- [25] Sylvia Osborn, Ravi Sandhu, and Qamar Munawer. 2000. Configuring Role-based Access Control to Enforce Mandatory and Discretionary Access Control Policies. *ACM Trans. Inf. Syst. Secur.* 3, 2 (May 2000), 85–106. <https://doi.org/10.1145/354876.354878>
- [26] Charlie Osborne. 2015. OwnStar: Unlock and track any GM OnStar connected car for \$100. <https://www.zdnet.com/article/how-to-steal-a-tesla-model-s-in-seconds/>.
- [27] Charlie Osborne. 2018. How to steal a Tesla Model S in seconds. <https://www.zdnet.com/article/how-to-steal-a-tesla-model-s-in-seconds/>.
- [28] Jonathan Petit, Bas Stottelaar, Michael Feiri, and Frank Kargl. 2015. Remote attacks on automated vehicles sensors: Experiments on camera and lidar. *Black Hat Europe* 11 (2015), 2015.
- [29] Raspberry Pi Foundation. 2019. Iridium9670 TPM2.0 Linux. <https://www.infineon.com/cms/de/product/evaluation-boards/iridium9670-tpm2.0-linux/>.
- [30] Raspberry Pi Foundation. 2019. Raspberry Pi 3 Model B+. <https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/>.
- [31] Marco Rocchetto and Nils Ole Tippenhauer. 2016. CPDY: extending the Dolev-Yao attacker with physical-layer interactions. In *International Conference on Formal Engineering Methods*. Springer, 175–192. https://doi.org/10.1007/978-3-319-47846-3_12
- [32] Ravi S Sandhu. 1998. Role-based access control. In *Advances in computers*, Marvin V. Zelkowitz (Ed.), Vol. 46. Elsevier, 237–286. [https://doi.org/10.1016/S0065-2458\(08\)60206-5](https://doi.org/10.1016/S0065-2458(08)60206-5)
- [33] R. S. Sandhu and P. Samarati. 1994. Access control: principle and practice. *IEEE Communications Magazine* 32, 9 (Sep. 1994), 40–48. <https://doi.org/10.1109/35.312842>
- [34] I. Studnia, V. Nicomette, E. Alata, Y. Deswarte, M. KaÄcniche, and Y. Laarouchi. 2013. Survey on security threats and protection mechanisms in embedded automotive networks. In *2013 43rd Annual IEEE/IFIP Conference on Dependable Systems and Networks Workshop (DSN-W)*. 1–12. <https://doi.org/10.1109/DSNW.2013.6615528>
- [35] Vangelis Stykas and George Lavdanis. 2018. Incorrect privilege assignment could lead to full user and vehicle compromise. <https://seclists.org/fulldisclosure/2018/May/37>.
- [36] Iraklis Symeonidis, Abdelrahman Aly, Mustafa Asan Mustafa, Bart Mennink, Siemen Dhooghe, and Bart Preneel. 2017. SePCAR: A Secure and Privacy-Enhancing Protocol for Car Access Provision. In *Computer Security – ESORICS 2017*, Simon N. Foley, Dieter Gollmann, and Einar Snekenes (Eds.). Springer International Publishing, Cham, 475–493. https://doi.org/10.1007/978-3-319-66399-9_26
- [37] Trusted Computing Group. 2014. Trusted Computing Group Releases TPM 2.0 Specification for Improved Platform and Device Security. <https://trustedcomputinggroup.org/trusted-computing-group-releases-tpm-2-0-specification-improved-platform-device-security/>.
- [38] Trusted Computing Group. 2016. TPM 2.0 Library Specification. <https://trustedcomputinggroup.org/resource/tpm-library-specification/>. Accessed: 2020-05-08.
- [39] Trusted Computing Group. 2018. Algorithm Registry. <https://trustedcomputinggroup.org/resource/tcg-algorithm-registry/>.
- [40] Trusted Computing Group. 2019. TCG Feature API (FAPI) Specification. https://trustedcomputinggroup.org/wp-content/uploads/TSS_FAPI_v0.94_r04_pubrev.pdf.
- [41] Trusted Computing Group. 2019. TCG TSS 2.0 JSON Data Types and Policy Language Specification. https://trustedcomputinggroup.org/wp-content/uploads/TSS_JSON_Policy_v0.7_r04_pubrev.pdf.
- [42] Liam Tung. 2018. VW-Audi security: Multiple infotainment flaws could give attackers remote access. <https://www.zdnet.com/article/vw-audi-security-multiple-infotainment-flaws-could-give-attackers-remote-access/>.
- [43] Uber Technologies Inc. 2020. A guide for how to use Uber. <https://www.uber.com/us/en/ride/how-it-works/>. Accessed: 2020-05-08.
- [44] James Vincent. 2016. World's first self-driving taxi trial begins in Singapore. <https://www.theverge.com/2016/8/25/12637822/self-driving-taxi-first-public-trial-singapore-nutonomy>. Accessed: 2020-05-08.
- [45] Marko Wolf and Timo Gendrullis. 2011. Design, Implementation, and Evaluation of a Vehicular Hardware Security Module, In *Information Security and Cryptology – ICISC 2011*. SpringerLink 7259.
- [46] W. Zeng, M. Khalid, and S. Chowdhury. 2015. A qualitative comparison of FlexRay and Ethernet in vehicle networks. In *2015 IEEE 28th Canadian Conference on Electrical and Computer Engineering (CCECE)*. 571–576.

A A PRACTICAL EXAMPLE OF A POLICY PATH

Figure 9 gives an example of a policy path. It suits the use case of a car being rented to a customer $\mathcal{U}$ for five days. The key K_{Fn}^C is required to start the car's engine. A usage policy and an offline delegation policy for $\mathcal{U}$ are issued, signed by the backend of the rental company and attached to K_{Fn}^C 's TPM2_PolicyAuthorize. Both embed the TPM2_PolicyCounterTimer command into the policy to implement a time constraint for $\mathcal{U}$. Furthermore, a revocation mechanism is implemented by a TPM2_PolicyNV. *Counter* is one of the TPM's NVRAM indices used as a monotonic counter. Revocation is done by incrementing *Counter* until it is greater than *RevokeCounter*. The usage policy ends with TPM2_PolicySigned to check $\mathcal{U}$'s authentication. The offline delegation policy ends with TPM2_PolicyAuthorize. This allows sub-policies signed by $\mathcal{U}$.

$\mathcal{D}$'s usage sub-policy is issued and signed by $\mathcal{U}$ and attached to the offline delegation policy. It restricts $\mathcal{D}$'s access to two of the five days by including another TPM2_PolicyCounterTimer and requires an authentication of $\mathcal{D}$ with TPM2_PolicySigned.

Figure 9: An example policy path

B EXAMPLES OF JSON-ENCODED POLICIES

In this section two policy templates and a corresponding instantiated and authorized policy are shown. These policies are JSON-encoded according to the TSS 2.0 JSON Data Types and Policy Language [41]. For better readability and brevity, long values, in particular PEM keys, digests and signatures, are abbreviated and some more insignificant fields are left out.

The policy templates are shown in Listing 1 and Listing 2. Listing 1 shows a user policy and Listing 2 shows an offline delegation policy template consisting of the offline delegation policy and the

corresponding sub-policy. Both policy templates activate the same features since the features field is identical.

Listing 1: Basic User policy template

```

1 {
2   "description": "uid:231432546464, features
3     :3f25664de646a4221e3563f46",
4   "policy": [{
5     "type": "NV",
6     "nvPath": "/nv/Owner/ctr",
7     "operandB": "00000000000001d8",
8     "operation": "signed_lt"
9   },
10  {
11    "type": "CounterTimer",
12    "operandB": "5000",
13    "operation": "signed_lt"
14  },
15  {
16    "type": "Signed",
17    "keyPEM": "-----BEGIN PUBLIC KEY[...]"
18    ,
19    "keyPEMhashAlg": "SHA256"
20  }
21 ]

```

Listing 2: Offline delegation policy and sub-policy template

```

1 {
2   "description": "uid:231432546465,
3     features:3f25664de646a4221e3563f46",
4   "policy": [{
5     "type": "POLICYAUTHORIZE",
6     "policyRef": [ 1, 2, 3, 4, 5 ],
7     "keyPEM": "-----BEGIN PUBLIC[...]"
8   }, {
9     "type": "NV",
10    "nvPath": "/nv/Owner/ctr",
11    "operandB": "00000000000001d8",
12    "operation": "signed_lt"
13  }
14 ]
15 {
16   "description": "uid:231432546465, features
17     :3f25664de646a4221e3563f46",
18   "policy": [{
19     "type": "CounterTimer",
20     "operandB": "5000",
21     "operation": "signed_lt"
22   }, {
23     "type": "Signed",
24     "keyPEM": "-----BEGIN PUBLIC KEY[...]",
25     "keyPEMhashAlg": "SHA256"
26   }
27 ]
28 }

```

Listing 3 shows the corresponding instantiated and authorized user policy from the template shown in Listing 1. This policy is close to the policies that are actually transmitted during our protocols and used to report the transmission sizes. Compared to its template, it is enriched with additional data, like parameters of the NV memory and the used cryptographic algorithms. Since it is an authorized policy, it also has a signature field with the corresponding public key that will be used for verification during the policy processing.

Listing 3: Instantiated and authorized basic user policy

```

1 {
2   "description": "uid:231432546464,
3     features:3f25664de646a4221e3563f46",
4   "policyDigests": [{
5     "hashAlg": "SHA256",
6     "digest": "7c22cdb6b04db9c[...]"
7   }],
8   "policyAuthorizations": [{
9     "type": "pem",
10    "policyRef": [1, 2, 3, 4, 5],
11    "key": "-----BEGIN PUBLIC KEY[...]",
12    "signature": "3045022100c5[...]"
13  }],
14   "policy": [{
15     "type": "POLICYNV",
16     "policyDigests": [{
17       "hashAlg": "SHA256",
18       "digest": "5e957fb[...]"
19     }],
20     "nvIndex": 25165824,
21     "nvPublic": {
22       "size": 0,
23       "nvPublic": {
24         "nvIndex": 25165824,
25         "nameAlg": "SHA256",
26         "attributes": {
27           "TPM2_NT": "COUNTER"
28         },
29         "PPWRITE": 0,
30         "OWNERWRITE": 0,
31         [...]
32       },
33       [...]
34     },
35     "operandB": "0000000000000002",
36     "operation": "SIGNED_LT"
37   }, {
38     "type": "POLICYCOUNTERTIMER",
39     "policyDigests": [{
40       "hashAlg": "SHA256",
41       "digest": "d74af43[...]"
42     }],
43     "operandB": "5000",
44     "operation": "SIGNED_LT"
45   }, {
46     "type": "POLICYSIGNED",
47     "policyDigests": [{

```



```

46         "hashAlg": "SHA256",
47         "digest": "7c22cdb[...]"
48     }],
49     "keyPublic": {
50         "type": "ECC",
51         "nameAlg": "SHA256",
52         "objectAttributes": {
53             [...]
54             "sign": 1
55         },
56         [...]
57         "parameters": {
58             [...]
59             "scheme": {
60                 "scheme": "ECDSA",
61                 "details": {
62                     "hashAlg": "SHA2
63                     56"
64                 },
65                 "curveID": "NIST_P256",
66                 [...], [...]
67             },
68             "keyPEMhashAlg": "SHA256",
69         }
70     }

```

C PROTOTYPE ENVIRONMENT

Figure 10 shows our prototype environment. It consists of an Intel NUC, two Raspberry Pis and two Android smartphones. The NUC represents the backend server ($\mathcal{B}$) that issues the policies. The two Android smartphones represent the tokens ($\mathcal{T}_U$ and $\mathcal{T}_D$) of user $\mathcal{U}$ and $\mathcal{D}$ that are used to download the policies from $\mathcal{B}$. The Raspberry Pis represent the two in-vehicle ECUs FC and RE. FC is the function controller with the attached TPM 2.0. The TPM 2.0 validates the presented policies and unlocks the feature keys. They are used in our use cases to unlock features on both the FC and the remote ECU (RE) or to establish a secure online connection with $\mathcal{B}$.

$\mathcal{T}_U$ has an exemplary policy without restrictions and $\mathcal{T}_D$ has a delegated policy with a time limitation of 5 days and a limited feature set without online connectivity so that $\mathcal{D}$ cannot consume data volume of $\mathcal{U}$.

With our system, restrictions already defined by $\mathcal{B}$ may only be further limited by $\mathcal{U}$ but not widened. This is a desired feature since $\mathcal{B}$ can for example enforce that safety-critical Advanced Driver-Assistance Systems (ADAS) like the pedestrian monitor or the lane keep assistant may not be turned off in a delegated policy.

Figure 10: Prototype Environment