# Smart Keys for Cyber-Cars: Secure Smartphone-based NFC-enabled Car Immobilizer

Christoph Busold, Ahmad-Reza Sadeghi, Christian Wachsmann
Intel CRI-SC, TU Darmstadt, Germany
christoph.busold@cased.de, ahmad.sadeghi@cased.de, christian.wachsmann@cased.de

Alexandra Dmitrienko
Fraunhofer SIT, Darmstadt, Germany
alexandra.dmitrienko@cased.de

Hervé Seudié, Majid Sobhani, Ahmed Taha
TU Darmstadt, Germany
herve.seudie@cased.de, majid.sobhani@cased.de, ahmed.taha@cased.de

## ABSTRACT

Smartphones have become very popular and versatile devices. An emerging trend is the integration of smartphones into automotive systems and applications, particularly access control systems to unlock cars (doors and immobilizers). Smartphone-based automotive solutions promise to greatly enhance the user's experience by providing advanced features far beyond the conventional dedicated tokens/transponders.

We present the first *open* security framework for secure smartphone-based immobilizers. Our generic security architecture protects the electronic access tokens on the smartphone and provides advanced features such as context-aware access policies, remote issuing and revocation of access rights and their delegation to other users. We discuss various approaches to instantiate our security architecture based on different hardware-based trusted execution environments, and elaborate on their security properties. We implemented our immobilizer system based on the latest Android-based smartphone and a microSD smartcard. Further, we support the algorithmic proofs of the security of the underlying protocols with automated formal verification tools.

## Categories and Subject Descriptors

D.4.6 [**Security and Protection**]: Access controls; K.6.5 [**Security and Protection**]: Authentication

## General Terms

Security, Design

## Keywords

Immobilizer; Mobile Security; Access Control; Delegation


*CODASPY'13,* February 18–20, 2013, San Antonio, Texas, USA.


## 1. INTRODUCTION

Today, smartphones are high performance platforms providing a wide range of features and have become an integral part of our daily life. The increasing computing and storage capabilities, the vast number and variety of apps available on app stores and new communication interfaces, such as Near Field Communication (NFC), provide many deployment possibilities for smartphones, including electronic ticketing [1], payment [27] and access control [46, 22]. In this context, an emerging trend is the integration of smartphones into modern automotive systems and applications such as access control to unlock, configure and start vehicles [41, 39, 47]. In particular, the NFC interface is well-suited for such applications due to its short nominal communication range (of a few centimeters) providing basic assurance of the user's physical proximity.

In this paper, we focus on smartphone-based NFC-enabled immobilizer systems. An electronic vehicle immobilizer is an anti-theft device that prevents starting the vehicle's engine unless the corresponding access token is (physically) present and authenticated. Currently, this access token is a transponder (i.e., an RFID chip) embedded into the mechanical car key or a contactless smartcard.

Smartphone-based immobilizer systems promise to enhance the user experience by providing a variety of appealing new features and enabling flexible applications beyond what is provided today by conventional transponder-based immobilizer systems. They do not require users to obtain a physical transponder but allow them to use their smartphone to remotely obtain electronic car keys (or access rights for the immobilizer). Moreover, access rights can be delegated to other users, revoked or bound to specific policies. In particular, automotive applications with a highly dynamic or large set of users, such as car sharing and fleet management, can highly benefit from smartphone-based immobilizers.

Despite the mentioned advantages for users, the core challenge concerns the security aspects of smartphone-based immobilizer systems. Smartphones are complex devices and appealing targets of attacks (e.g., by malware), especially when they are used in security-critical applications.

The traditional immobilizers used in practice are closed and proprietary systems and suffer from various security vulnerabilities, as recent attacks show [31, 33, 24, 50]. The reasons are conceptual protocol design flaws as well as the de-

ployment of insecure or weak cryptographic schemes. On the other hand, several prototypes of commercial smartphone-based and NFC-enabled immobilizer systems have been introduced recently [41, 39, 47], but without providing technical details or information on their security properties.

**Our goal and contribution.** We present an *open* smartphone-based immobilizer system architecture and the underlying security framework, which provides enhanced functional and security features and overcomes the security issues of the conventional immobilizer systems. In particular, our contribution is as follows:

*Framework for smartphone-based immobilizer systems.* Our framework considers the functional and security requirements on the protocols and the system architecture of a smartphone-based solution under realistic adversary models.

*Evaluation of existing security hardware.* We evaluate and discuss various instantiations of our security architecture using different approaches to establish trusted execution environments on smartphones. We discuss which security guarantees can be provided by these instantiations, under which assumptions, and how some of these assumptions can be fulfilled by leveraging the features of security hardware currently available on recent smartphones.

*Formal tool-based protocol analysis.* Our protocols design is based on the protocols in [22], which we adapt to the immobilizer system. Additionally, we analyze the security of these protocols using the automated verification tool ProVerif [15], which is complementary to the cryptographic security analysis in [22].

*Implementation.* We present an implementation of our immobilizer system on Android using the latest smartphone hardware and a secure microSD smartcard. We show that it is feasible to implement a secure NFC-enabled and smartphone-based immobilizer system. In particular, we discuss the conditions for the secure integration of enhanced features such as delegation under a strong but realistic adversary model, where the adversary has full control over the software on the smartphone platform. Hereby, we take the technical limitations of currently available security hardware for smartphones into account.

**Outline.** We present our framework for smartphone-based immobilizer systems and the requirement analysis in Section 2. We describe the platform security architecture in Section 3 and discuss the available secure hardware in Section 4. We present the implementation and evaluation of our solution in Section 5 and analyze its security in Section 6. Finally, we give an overview of related work in Section 7 and conclude in Section 8.

# 2. REQUIREMENT ANALYSIS

We first introduce the system and adversary model, our assumptions and present our objectives and requirements.

## 2.1 System Model

Our system model is depicted in Figure 1 and involves a car manufacturer M, a car C, a car owner O and a car user U. The manufacturer M produces cars equipped with immobilizers, which are electronic control units that prevent unauthorized users from starting the car engine. Moreover,

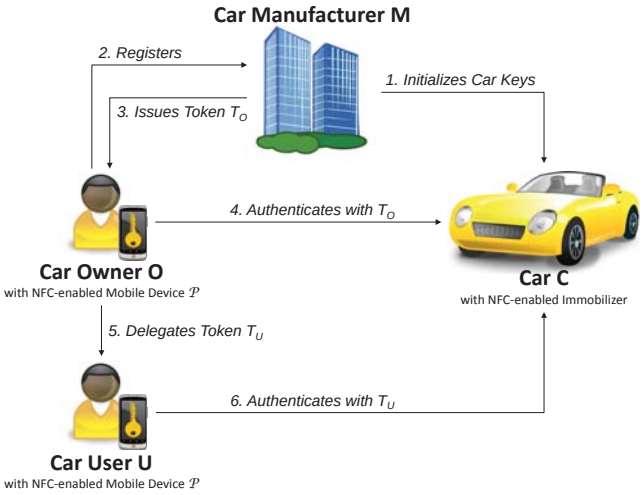


**Figure 1: System model**

M also represents car dealers and service stations authorized by the car manufacturer. The car owner O is a private person or a company that haspurchased the car and received an electronic access control token $T_O$ from M. The token is securely deployed and stored on the mobile platform $\mathcal{P}$ of O. The car user U is a person who is authorized by O to use the car. This can be a friend or a family member of the car owner, or an employee of the company owning the car. The authorization is given by means of issuing a delegated access control token $T_U$ which grants U access to C.

## 2.2 Adversary Model

**Communication channels.** The adversary $\mathcal{A}$ has full control over all communication channels except the channel between the car manufacturer M and the car C, which we assume to be secure.[1] Specifically, $\mathcal{A}$ can eavesdrop, inject, replay and modify all messages transmitted over the channels between O and M, O and U, O and C, and U and C.[2]

**Smartphone platform.** We assume that each mobile platform $\mathcal{P}$ consists of an untrusted host $\mathcal{H}$ and a trusted execution environment (TrEE) $\mathcal{S}$. $\mathcal{A}$ can perform software attacks against the untrusted host $\mathcal{H}$ and install, modify or compromise all software components installed on $\mathcal{P}$. However, we exclude hardware attacks and assume that $\mathcal{A}$ cannot compromise $\mathcal{S}$. We provide a detailed discussion on instantiations of TrEE on top of different security hardware available for mobile platforms in Section 4.

**Immobilizer.** We assume that the immobilizer C is trusted as in conventional (non-NFC-based) immobilizer systems. Specifically, we assume that an adversary who compromises the immobilizer can start the car without attacking the mo-

[1]A secure communication channel between the car C and the manufacturer M can be established using standard protocols such as SSL.

[2]Note, that we exclude relay attacks since the focus of this paper is a secure immobilizer system architecture for NFC-enabled smartphones. One possible solution to reduce the risk of relay attacks are distance bounding techniques, which can be combined with our scheme.

bile platform. Hence, we exclude attacks against the immobilizer component.

## 2.3 Objectives

As in traditional immobilizer systems, our main objective is to prevent unauthorized access to the immobilizer:

**O1: Access control.** Only authorized entities, namely O authorized by M and U authorized by O, should be able to unlock the immobilizer C.

Further, the performance, i.e., the time needed for authentication is a significant usability aspect, which is essential for a positive user experience:

**O2: Performance.** Authentication of O or U to C should be performed within an unnoticeable time interval [37, 8].

Moreover, the compatibility to existing smartphones is important to ensure the applicability of the solution in practice:

**O3: Compatibility.** An important requirement is the compatibility to commodity mobile platforms. The immobilizer system should be compatible with existing hardware and require no or only minor changes to the mobile operating system.

A smartphone-based immobilizer system should enable new appealing features, such as the remote issuing and revocation of electronic tokens, remote replacement of electronic keys in case of loss or theft of the mobile device, or provide mechanisms to ensure access revocation of former car owners in case of car re-sell. Hence, our additional objectives are as follows:

**O4: Remote issuing.** The car manufacturer M should be able to *remotely* (e.g., via the Internet) issue and deploy the electronic access token $T_O$ to the car owner O.

**O5: Remote revocation.** M should be able to *remotely* revoke access tokens $T_O$ issued t O. Moreover, revocation of $T_O$ by M should automatically revoke all delegated tokens $T_U$ issued by O.

Some other desirable enhanced features include token delegation and support for context-aware access policies:

**O6: Delegation.** A car owner O should be able to securely delegate her access rights to a third party U.

**O7: Policy-based access control.** A car owner O should be able to restrict access of delegated users to the car based on contextual information such as time and location.

As we discuss later in Section 6, off-the-shelf smartphone platforms and security hardware can be used to achieve objectives (O1) to (O5). However, due to the technical constraints of available security hardware and the limitations posed by some security hardware manufacturers, objectives (O6) and (O7) cannot be realized with the currently available commodity hardware.

## 2.4 Security Requirements

### *Protocol-specific requirements.*

The main security objective of an immobilizer system is the secure authentication of the car owner O (or the delegated user U) to the immobilizer C.

### *Platform-specific requirements.*

Mobile platforms typically host a mobile operating system that can potentially be compromised and expose all secrets stored on the platform. Hence, to achieve objective (O1), the security-sensitive data used in the underlying protocols must be protected against untrusted code. Therefore, we define the following security requirements on the underlying mobile platform:

**SR1: Secure storage.** Security-sensitive data should not be accessible by untrusted software components while stored on the platform.

**SR2: Isolation.** The system components operating on security-sensitive data must be trusted and isolated from the untrusted components.

Further, it has to be ensured that the security sensitive operations, such as authentication and delegation, are triggered by the user rather than by malware. Moreover, advanced use cases, such as delegation and policy-based access control, rely on security-critical user inputs, such as passwords and user-defined access-control policies. Hence, for these use cases we need an additional security requirement:

**SR3: Secure user interface.** The user (the car owner O or the car user U) should be able to securely communicate with the trusted components.

**Discussion.** To achieve isolation on the platform, one could apply virtualization-based approaches or use a hardened operating system that provides isolation properties (such as proposed in [22]). However, this requires changes to the underlying operating system and may be hard to achieve in practice (O3). Hence, our primary goal is to leverage the general purpose secure hardware available for commodity mobile platforms to establish a hardware-isolated trusted execution environment (TrEE). Although the technology to achieve the objectives (SR1) to (SR3) is partially available, it is not widespread. In particular, a secure user interface (SR3) can be realized only with certain types of secure hardware. We provide an overview of available secure hardware and discuss its features in more details in Section 4.

# 3. SYSTEM DESIGN

In this section, we present a secure NFC-based immobilizer system. The solution includes cryptographic protocols for the secure interaction between the involved entities and a mobile security architecture to protect security-sensitive data, such as the cryptographic secrets used in the protocols, when they are processed and stored on the mobile platform.

## 3.1 Mobile Platform Security Architecture

Our security architecture is depicted in Figure 2. The execution environment of the mobile platform is divided into two independent worlds: An untrusted host $\mathcal{H}$ and a trusted execution environment $\mathcal{S}$. The host runs on the general purpose processor of the mobile device, while the TrEE is established on top of secure hardware. Such secure hardware can be either embedded into the smartphone or attached to the mobile device via the standard communication interfaces, which does not require any changes to commodity platforms.

Depending on the type of the secure hardware used, the TrEE can either have a direct (secure) connection to the

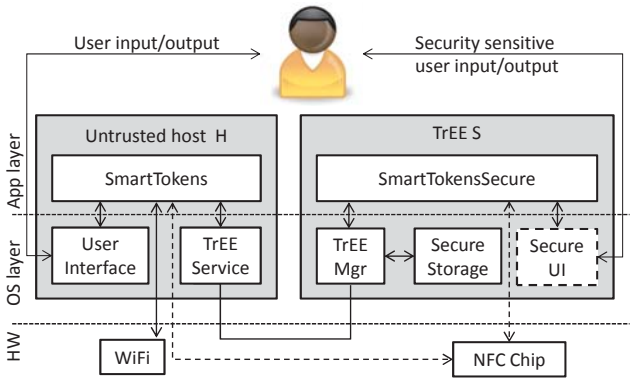


**Figure 2: Mobile platform security architecture. Dashed boxes indicate optional components that are not available on current security hardware. Further, the NFC chip can be either controlled by the smartphone OS or have a direct connection to the TrEE, indicated by dashed lines.**

NFC chip or must rely on the untrusted operating system to handle the NFC (illustrated as a dashed line in Figure 2). In contrast, the WiFi or the mobile network interface that is used for the communication with, e.g., the car manufacturer M, is always managed by the operating system.

The functionality of our system is realized within the SmartTokens app and the SmartTokensSecure component residing at the application level. The SmartTokens app manages the access control tokens and handles different protocols (such as registration, delegation and authentication), while the SmartTokensSecure component is only invoked to perform the computations involving security-critical data, such as the cryptographic secrets used in the protocols. All security-sensitive data is handled by the SmartTokensSecure component. This data never leaves the TrEE in cleartext and is securely stored in the SecureStorage component of $\mathcal{S}$.

The SmartTokens and SmartTokensSecure apps communicate via a channel established between both execution environments. The channel is handled by the TrEEService and TrEEMgr components residing at the operating system level. These components are responsible for multiplexing the communication between the different applications running on the phone. TrEEMgr additionally manages the access of different TrEE applications to the SecureStorage component, so that other applications, possibly residing within the TrEE, cannot access the cryptographic parameters of the SmartTokensSecure app.

The user input is handled by two components in the system: The user interface UI provided by the operating system residing within $\mathcal{H}$ and a secure user interface SecureUI based on the TrEE. UI is used for the ordinary interaction with the user, while all security-sensitive data, such as passwords and access control policies are handled by the SecureUI component. SecureUI is customized with a background picture or a unique paraphrase only known to the user and the TrEE, allowing the user to distinguish between SecureUI and UI.

Note, that SecureUI based on the TrEE can be provided only by certain types of secure hardware, as we discuss later in Section 4. Thus, this component is optional in our system architecture (indicated by a dashed box in Figure 2). The architecture instantiation without the SecureUI can achieve the basic objectives in Section 2.3, while the advanced objectives are achievable only in a relaxed adversary model, where the adversary cannot compromise the UI.

## 3.2 Protocol Design

Our protocol design is along the lines of the token-based access control system by Dmitrienko et al. [22]. This scheme consists of six protocols for initialization, user registration, token issuing, token delegation and the authentication protocol for registered and delegated users, respectively. In the following, we briefly describe these protocols.

**Initialization of TrEE.** The manufacturer of the trusted execution environment $\mathcal{S}$ installs the SmartTokensSecure app in $\mathcal{S}$ and initializes the SecureStorage of $\mathcal{S}$ with a unique decryption/encryption key pair $(sk_{\mathcal{P}}, pk_{\mathcal{P}})$ and a platform certificate $cert_{\mathcal{P}}$, which attests that $\mathcal{S}$ is a genuine trusted execution environment and that only $\mathcal{S}$ knows the secret decryption key $sk_{\mathcal{P}}$ that corresponds to the public encryption key $pk_{\mathcal{P}}$.

**Immobilizer initialization.** The car manufacturer M initializes the immobilizer C with an authentication secret $K^{\mathsf{C}}_{\mathrm{Auth}}$ and an encryption key $K^{\mathsf{C}}_{\mathrm{Enc}}$, which are both used in the authentication protocol.

**Owner registration.** Before purchasing a car, the car owner O registers her platform $\mathcal{P}$ with the car manufacturer M. In this process, M verifies the platform certificate $cert_{\mathcal{P}}$ of $\mathcal{P}$ and generates an authentication secret $K^{\mathsf{O,M}}_{\mathrm{Auth}}$ and decryption key $K^{\mathsf{O,M}}_{\mathrm{Enc}}$ for O, which are used later in the token issuing protocol. Further, M encrypts both keys with $pk_{\mathcal{P}}$ and sends the ciphertext back to $\mathcal{S}$, where the SmartTokensSecure app decrypts both keys and stores them in SecureStorage.

**Token issuing.** In this protocol, M generates an authentication key $K^{\mathsf{O}}_{\mathrm{Auth}}$, which is used in the authentication protocol to unlock the immobilizer C, and a delegation key $K^{\mathsf{O}}_{\mathrm{Del}}$ that is used in the delegation protocol to create delegated tokens. Both keys and the identity $ID_{\mathsf{O}}$ of O are authenticated with the immobilizer authentication secret $K^{\mathsf{C}}_{\mathrm{Auth}}$ and encrypted under the immobilizer encryption key $K^{\mathsf{C}}_{\mathrm{Enc}}$, i.e.,

$$\sigma_{\mathsf{M}} := \mathtt{MAC}\left(K^{\mathsf{C}}_{\mathrm{Auth}}; ID_{\mathsf{O}}, K^{\mathsf{O}}_{\mathrm{Auth}}, K^{\mathsf{O}}_{\mathrm{Del}}\right)$$
$$T_{\mathsf{O}} := \mathtt{Enc}\left(K^{\mathsf{C}}_{\mathrm{Enc}}; ID_{\mathsf{O}}, K^{\mathsf{O}}_{\mathrm{Auth}}, K^{\mathsf{O}}_{\mathrm{Del}}, \sigma_{\mathsf{M}}\right)$$

$T_{\mathsf{O}}$ is the *access token* of O for C and used later in the authentication protocol. Furthermore, M and the SmartTokensSecure app in $\mathcal{S}$ establish a secure channel based on $K^{\mathsf{O,M}}_{\mathrm{Auth}}$ and $K^{\mathsf{O,M}}_{\mathrm{Enc}}$, which is used to send $T_{\mathsf{O}}$ from M to SmartTokensSecure. Finally, SmartTokensSecure stores $K^{\mathsf{O}}_{\mathrm{Auth}}$, $K^{\mathsf{O}}_{\mathrm{Del}}$ and $T_{\mathsf{O}}$ in the SecureStorage of $\mathcal{S}$.

**Car owner authentication.** The authentication protocol is depicted in Figure 3: O uses the SmartTokens app to initiate an authentication request sent from O's mobile platform $\mathcal{P}$ to the immobilizer C. Then C sends its identifier $ID_{\mathsf{C}}$ and a random $N$ (the bit-length $\mu$ of $N$ is a security-critical parameter [22]) to SmartTokensSecure on $\mathcal{P}$, which replies with $\sigma_{\mathsf{O}}$ to the SmartTokens app in $\mathcal{H}$ that eventually sends $(\sigma_{\mathsf{O}}, T_{\mathsf{O}})$ to C. Next, C decrypts $T_{\mathsf{O}}$ with $K^{\mathsf{C}}_{\mathrm{Enc}}$ to obtain $K^{\mathsf{O}}_{\mathrm{Auth}}$, verifies $\sigma_{\mathsf{M}}$ and $\sigma_{\mathsf{O}}$ using $K^{\mathsf{C}}_{\mathrm{Auth}}$ and $K^{\mathsf{O}}_{\mathrm{Auth}}$, respec-

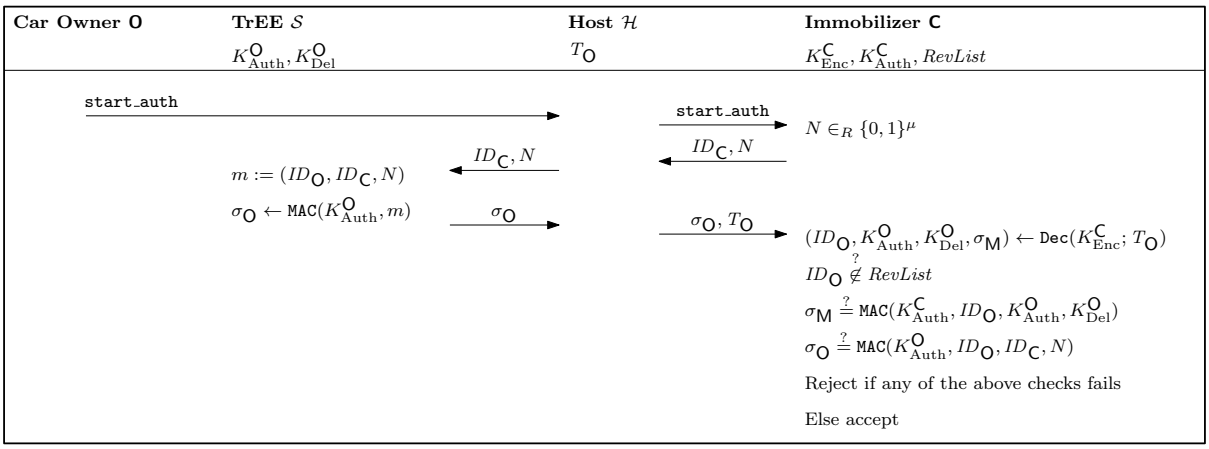


**Figure 3: Authentication protocol**

tively, and accepts only if both verifications are successful. Otherwise, C rejects.

**Token delegation.** The car owner O delegates her access rights to another car user U by creating a delegated token $T_{\mathsf{U}}$. Specifically, SmartTokensSecure in the TrEE of O's platform creates an authentication secret $K^{\mathsf{U}}_{\text{Auth}}$ for U, authenticates it along with the identifier $ID_{\mathsf{U}}$ of U using $K^{\mathsf{O}}_{\text{Auth}}$ and encrypts all data under the delegation key $K^{\mathsf{O}}_{\text{Del}}$:

$$\sigma_{\mathsf{O}} := \texttt{MAC}\left(K^{\mathsf{O}}_{\text{Auth}}; ID_{\mathsf{U}}, K^{\mathsf{U}}_{\text{Auth}}\right)$$
$$T_{\mathsf{U}} := \texttt{Enc}\left(K^{\mathsf{O}}_{\text{Del}}; ID_{\mathsf{U}}, K^{\mathsf{U}}_{\text{Auth}}, \sigma_{\mathsf{O}}\right)$$

Furthermore, O and U establish a secure channel based on the platform key of U's TrEE, which is used to transfer $T_{\mathsf{U}}$ to SmartTokensSecure in the TrEE of U's device.

**Car user authentication.** For the authentication of a delegated car user U, U's TrEE sends to the immobilizer C two tokens, $T_{\mathsf{O}}$ and $T_{\mathsf{U}}$. C first decrypts $T_{\mathsf{O}}$ to obtain $K^{\mathsf{O}}_{\text{Del}}$, which is then used to decrypt $K^{\mathsf{U}}_{\text{Auth}}$ from $T_{\mathsf{U}}$. The rest of the authentication protocol is the same as in Figure 3. Note that $T_{\mathsf{U}}$ is linked to $T_{\mathsf{O}}$ since $T_{\mathsf{U}}$ is a ciphertext that can be decrypted only with the delegation key $K^{\mathsf{O}}_{\text{Del}}$ of O, which is included in $T_{\mathsf{O}}$.

**Remote revocation.** The remote revocation of tokens is realized in a form of revocation lists deployed by M to the car immobilizer C. Revocation of $T_{\mathsf{O}}$ by M automatically revokes all delegated tokens $T_{\mathsf{U}}$ issued by O, since $T_{\mathsf{O}}$ is involved in the authentication of each delegated user U.

**Secure user input.** The token delegation protocol requires a secure input from the car owner O to the SmartTokensSecure app. Specifically, the car user O must approve the delegation of her access rights and specify the corresponding access control policy for the car user U.

## 4. SECURE HARDWARE

In the following, we analyze the features of the available security hardware for smartphones and discuss which of them are most appropriate for the realization of a secure smartphone-based immobilizer system.

Most known security hardware available for modern smartphones includes ARM TrustZone [4], MShield [11], smartcards, SIM-cards and secure microSD cards [26]. A comparison of the corresponding features is depicted in Table 1. Note that all considered security hardware provides secure storage (SR1) and isolation (SR2).

**ARM TrustZone.** ARM TrustZone [4] allows for establishment of a trusted execution environment (TrEE) that could provide a secure user interface (SR3), eventually fulfilling all of our platform-related security requirements. However, TrustZone is available only on a few platforms, such as Apple's iPhone. Further, it is usually deactivated or locked by the phone manufacturer and cannot be used by third party applications running on the phone. Although the TrustZone API is public and software emulators are available, only selected third party developers get access to TrustZone development boards.

**MShield.** MShield is a closed TrEE that can be used by third party developers with the help of the on-board credentials (ObC) security framework [11, 34]. However, MShield is hardly available in practice and can only be found in a few high-end Nokia smartphones.

**SIM-cards.** SIM-cards are the most widespread TrEEs and available on every phone. However, SIM-cards are typically closed systems controlled by the network operators. Hence, a solution based on SIM-cards would be available only to customers of the particular network operator controlling the SIM-card.

**Embedded secure elements.** Embedded secure elements are available on some recent NFC-enabled Android phones, such as the Samsung Nexus S and the Samsung Galaxy Nexus. However, they are locked and can only be used with the Google Wallet payment system [28].

Combining the secure element and the NFC interface into one chip allows them to operate passively, i.e., without the power supply of the smartphone platform. In this mode, the secure element and NFC chips use the RF field generated by the NFC reader as power supply, a feature NFC inherited from contactless smartcards. Provided that the NFC chip in the phone supports this mode, it could be used to run the authentication protocol (Figure 3) even when the battery of the phone is depleted. On the protocol side, only a minor

**Table 1: Comparison of current secure smartphone hardware**

| Security hardware | Secure storage (SR1) | Isolation (SR2) | Secure user interface (SR3) | Open to 3rd parties | Availability |
|---|---|---|---|---|---|
| ARM TrustZone [4] | + | + | + | - | some phones, e.g., iPhone (not activated) |
| TI MShield [11] | + | + | - | with ObC [34] | some phones, e.g., Nokia |
| SIM-card | + | + | - | - | every phone |
| Embedded SE | + | + | - | - | some phones, e.g., Samsung Nexus S and Galaxy Nexus |
| Secure microSD card [26] | + | + | - | + | any phone with microSD slot |
| Secure microSD card with NFC [49] | + | + | - | + | any phone with microSD slot & NFC antenna |

modification would be required: The tokens must be stored in the secure element to make them available to the TrEE when the host is powered down.

**Secure microSD cards.** A promising approach is using secure microSD cards, which are microSD memory cards that include a secure element. They can be used in every smartphone with a microSD card slot. Some of these cards include an NFC chip that uses an NFC antenna integrated in the microSD card or can be connected to an external NFC antenna built into the phone [49]. Such microSD cards enable solutions that reach a large number of users because they can even be used on phones without an integrated NFC interface. However, microSD cards with integrated NFC are hardly available and most stock phones are not equipped with NFC antennas.

ARM TrustZone seems to be the most suitable TrEE for our architecture, since it satisfies all platform-related security requirements, while all other TrEE types do not provide a secure user interface. However, developments for TrustZone are currently limited to development boards, thus we have to consider other types of TrEEs for our prototype.

Our implementation of the smartphone-based immobilizer system uses an NFC-enabled smartphone with a secure microSD card, which seems to be the most applicable configuration in practice. Due to unavailability of a secure user interface when using only a microSD card, our prototype implementation achieves only the objectives (O1) - (O5), while achieving objectives (O6) - (O7) requires either a more enhanced TrEE or a weaker adversary model. We provide an evaluation and detailed discussion of this aspect in Section 6.

# 5. IMPLEMENTATION AND EVALUATION

We now describe our implementation and present the performance measurements for the authentication protocols. Figure 4 shows the hardware setup we used.

**Smartphone.** We implemented the immobilizer application on Android using an NFC-enabled Samsung Galaxy S3 smartphone. The NFC hardware of the Galaxy S3 comes with a built-in secure element used for the Google Wallet electronic payment system [28]. However, this secure element is locked and cannot be used for custom applications such as our immobilizer system. Therefore, we use a Giesecke & Devrient Mobile Security Card 1.0, which is a microSD smart card that allows installation of custom applications. The underlying smart card operating system complies to the JavaCard 2.2.2 and GlobalPlatform 2.2.1 specifications and provides all primitives required by the protocols: Public-key encryption (RSA), symmetric encryption (AES in CBC mode), a cryptographic hash function (SHA1) and a cryptographic random number generator.

The UI component of our architecture is represented by the keyboard and display drivers already present in the Android OS. The TrEEService implementation is based on the smart card API provided by the Seek-for-Android project [44]. It enables access to smartcards via APDUs as defined in ISO7816. The Galaxy S3 stock firmware already contains this smart card API for the built-in secure element, however, it is not enabled for the Mobile Security card. Therefore, we had to replace the firmware of the phone with a custom build of CyanogenMod9 [20] based on Android 4.0.3 where we included the smart card API patches (version 2.3.2). However, Seek-for-Android plans to release a plugin-in terminal for the Mobile Security Card, which can be used with the existing API and thus would not require a custom firmware.

Our implementation of the communication protocols and the SmartTokens application are based on [22]. However, they implemented the TrEE in software as an operating system service on top of a hardened Android OS. Our system in contrast does not require a hardened OS but utilizes a hardware isolated TrEE. The functionality of the SmartTokensSecure component in our architecture (Section 3.2) is implemented in a JavaCard applet that uses the secure storage of the smart card. Moreover, we implemented an interface to the smart card that is compatible to the existing SmartTokens application.

**Immobilizer.** The immobilizer implementation uses a setup similar to commercially available immobilizer systems [40, 7]. Specifically, we use an Arduino Uno [5], which is a commercial development board based on a 8 bit Atmel AVR microcontroller with 32 KB memory clocked at 16 MHz. The Arduino is connected via a Serial Peripheral Interface (SPI) to an NFC interface [38] based on the PN532 controller [42].

**Figure 4: Prototype setup consisting of an Arduino Uno board with NFC shield, a secure microSD smart card and a Samsung Galaxy S3 smartphone (left to right).**

**Table 2: Performance measurements**

| User Type | Challenge Time (ms) | Response Time (ms) | Verification Time (ms) | Session Time (ms) |
|---|---|---|---|---|
| Owner | 72.05 | 384.90 | 24.00 | 480.95 |
| User | 72.20 | 544.85 | 41.65 | 658.70 |

The implementation of the immobilizer uses the cryptographic primitives provided by the AVR-Crypto-Lib [10], which is an open source library optimized for AVR microcontrollers. Furthermore, we adapted the NFC library provided with the PN532 NFC controller so that the NFC hardware emulates a contactless smartcard according to the NFC Forum type 4 and ISO14443-4 specifications [32, 2].

**Performance Evaluation.** We evaluated the performance of our implementation of the authentication protocol running between the smartphone and the immobilizer. For this purpose, we made the following measurements: (1) the time required to start the authentication mechanism and to get the challenge from the immobilizer after the NFC connection has been established, (2) the time required by the phone to send the response to the immobilizer, (3) the time required by the immobilizer to verify the phone's response, and (4) the time required for the complete authentication protocol. The results of our measurements averaged over 20 protocol runs are shown in Table 2. Our solution requires 480.95 ms ($\pm$26.95 ms) for the whole authentication process of the car owner and 658.70 ms ($\pm$74.30 ms) for the authentication of the car user, which is sufficient to provide positive user experience [37].

# 6. SECURITY CONSIDERATIONS

In this section we discuss our security architecture with respect to the security objectives outlined in Section 2.4.

## 6.1 Protocol Analysis

As mentioned in Section 3.2 our protocols are adapted from [22]. While security proofs of the protocols in cryptographic models are shown in [22], we additionally analyze the security of the protocols using automated verification tools. In this context prominent verification tools are ProVerif [15], Scyther [17] and Avispa [6]. We use ProVerif since it provides the largest feature set compared to other tools [18, 19, 21]. In contrast to other verification tools, ProVerif allows modeling of any known cryptographic function using the applied pi calculus specification language [3]. The corresponding protocol specifications were successfully verified by ProVerif, additionally supporting the cryptographic proofs and providing the confidence in fulfillment of the security objectives.

### *Formal Verification using ProVerif.*

ProVerif takes as input a formal description of the protocols, the adversary model and the security objectives based on the applied pi calculus specification language and proves or disproves the claimed security properties. This formal specification includes the following components:

**Agents.** Agents represent the protocol participants (car manufacturer $\mathsf{M}$, car owner $\mathsf{O}$, car user $\mathsf{U}$ and immobilizer $\mathsf{C}$), which have different *roles* such as being the sender/receiver of a protocol message.

**Events.** Events model the actions performed by the agents, such as computations and sending/receiving messages.

**Communication channels.** The communication channels are used to transfer messages between the agents.

**Security properties.** The security properties specify the confidentiality and authentication objectives of the protocols (Section 2.4). We use the common ProVerif specification of authentication based on the correspondence property [16], which states that every successful execution of a protocol step implies that the involved entities were honest and followed the protocol specification.

**Adversary.** The adversary and its capabilities are modeled as an agent that aims to violate the security objectives. Note that ProVerif uses the Dolev-Yao adversary model [23] by default, which corresponds to the adversary model in Section 2.2. Further, we assume that the car manufacturer $\mathsf{M}$, the immobilizer $\mathsf{C}$ and the trusted execution environments $\mathcal{S}$ of all mobile platforms are trusted while all other agents are untrusted.

Based on the formal specification, ProVerif proves or disproves the security properties. Specifically, the confidentiality property is checked by searching for protocol states where the adversary learns at least one of the cryptographic secrets. The authentication goal is checked by searching for protocol states where a message originating from the adversary is accepted by the immobilizer even though the adversary does not know the underlying authentication secrets.

The detailed formal specification, proof and result of the formal verification of the protocols using ProVerif can be found in the full version of this paper.

## 6.2 Analysis of the Security Architecture

Our security architecture leverages the underlying security hardware to satisfy the requirements (SR1) and (SR2). Particularly, it relies on a separate processor providing an isolated execution environment and a dedicated secure memory. Further, to satisfy (SR2), we ensure that the security-sensitive data is never available in plain text to untrusted code (as detailed in Section 3.2). Therefore, the code running on the untrusted host cannot access any secrets which are stored and processed inside the trusted execution environment (required by the protocols to achieve objective (O1), as we showed in Section 6.1.).

## 6.3 Advanced Security Features

Advanced objectives such as secure delegation (O6) and policy-based access control (O7) require a secure user interface to handle security-sensitive user input (as discussed in Section 3.2). Particularly, token delegation relies on a password-based authentication of the delegated user U against the car owner O before the delegated token is issued. Without a secure user interface, the password can be intercepted by malware and redirected to a malicious device that can impersonate U and receive the delegated token $T_U$. Further, context-aware access control requires the car owner O to define access control policies during the delegation process. When entered via an untrusted user interface, the access policy can be manipulated by a malware without consent of the car owner.

As we discussed in Section 4, among the currently available secure hardware ARM TrustZone seems to be capable of providing a secure user interface. However, currently it is not freely programmable. Thus, in the following, we discuss possible alternatives to a secure user interface on the phone that can provide a reasonable trade-off between the available technologies and the desired security features.

**NFC proximity.** NFC proximity provides a means to input a single bit of information directly into the TrEE. Particularly, a user can authorize or invoke a security sensitive operation such as token delegation or authentication by tapping his phone to the NFC reader. However, the following aspects have to be considered when using NFC proximity to authorize security sensitive operations: First, in case the NFC interface is handled by the operating system, the malware can emulate the proximity to the NFC reader by pretending to receive data from the NFC interface. Thus, NFC-based proximity can be reliably provided only by TrEEs that feature a direct connection to the NFC chip so that the untrusted operating system cannot spoof the communication. Examples of such secure hardware are SIM-cards, embedded secure elements and secure microSD cards with an integrated NFC chip. Moreover, a powerful adversary with specialized equipment may extend the nominal NFC range of 10–20 cm up to 1–10 meters [29] and trigger NFC without consent of the user. Furthermore, NFC-based proximity may potentially be subject to relay attacks [25]. In this case it can be applied only in a weaker adversary model that excludes these kinds of attacks.

NFC proximity enables secure delegation over the NFC interface. Particularly, the car owner O can authenticate the user U based on a visual contact, while both platforms can use a secure channel based on a key established via NFC[3]. However, NFC proximity cannot be used to securely enter context-aware access control policies.

**Car on-board computer.** Modern car on-board computers (e.g., head units) typically have large displays and input devices that can be leveraged as user interface. Moreover, it might be reasonable to assume that the car on-board computer (and hence its user interface) is trusted, since attacks on car on-board computers are much less common than attacks on mobile devices[4]. Moreover, the attacker controlling the car on-board computer most likely will have good chances to start the car engine directly by attacking its internal infrastructure.

The system that leverages the car user interface can achieve both, secure delegation and context-aware access control, as the car user interface can be used to enter the password required for the delegation, as well as to define context-aware security policies in a secure way.

# 7. RELATED WORK

In this section, we discuss existing immobilizer systems in practice and in literature. Further, we give an overview on related work regarding access control with smartphones.

### *NFC-based Immobilizers.*

NXP Semiconductors presented the prototype of an NFC-based immobilizer system [41]. The security of this approach relies on the secure element of the smartphone. However, it is unclear how this secure element is instantiated and whether this approach requires new phones with special security hardware. Furthermore, in contrast to our approach, this system does not consider delegation and may not be usable in advanced use cases such as car sharing applications.

NFC-based immobilizer systems supporting car sharing were proposed independently by the automotive component suppliers Valeo [39] and Continental [47]. They collaborated with network operators (Orange and Deutsche Telecom, respectively) and used SIM-cards as secure execution environments for the protection of their electronic car keys. While SIM-cards are available on each mobile platform, they are controlled by different network operators. Hence the solutions by Valeo and Continental would probably be available only to the customers of those network providers. Furthermore, there is no public information on the security mechanisms used in these solutions.

### *Transponder-based Immobilizers.*

Lemke et al. [35] present a system model and requirement analysis for electronic immobilizer systems that use dedicated hardware tokens. The proposed model does not capture advanced use cases such as delegation and thus cannot be applied to our system. Moreover, their model is concerned with the security aspects of the immobilizer, while we focus on the security of the user's mobile device and the protection of the authentication secrets from the untrusted mobile OS.

The first open specification of a security protocol stack for

[3]Note that the NFC link is commonly assumed not being susceptible to man-in-the-middle attacks [29]. Hence, when combined with visual authentication, it can be used for secure authenticated key establishment.

[4]More than 25,000 new malicious Android apps have been discovered in Q1 and Q2 of 2012 [51], while attacks on on-board computers are not widespread and very involved [43]

transponder-based immobilizer systems has been published by Atmel [9, 36]. Tillich et al. [48] uncovered several vulnerabilities in this stack and proposed fixes. This demonstrates the advantages of open approach to security design, which we also follow.

The first attempt of using public-key cryptography in immobilizer systems has been made by Heyszl et al. [30]. They showed that it is feasible to implement lightweight elliptic curve cryptography on resource-constrained transponders. However, since the amount of data to be transferred in their authentication protocol exceeds the constrained bandwidth of the NFC interface, their scheme is not appropriate for NFC-based immobilizer systems.

### *Delegable Access Control With Smartphones.*

Our work is along the lines of the SmartToken system by Dmitrienko et al. [22], which enables NFC-enabled smartphones to maintain electronic access control tokens that can be delegated to other users. Specifically, we adapt the protocols of the SmartToken scheme to the immobilizer use case and provide a tool-based security verification of these protocols. Further, we introduce a new platform security architecture that, in contrast to the architecture of the SmartToken system, does not rely on trusted operating system components and can be built on top of commodity mobile operating systems.

Another smartphone-based access control system supporting delegation of access rights has been presented by Bauer et al. [14, 12, 13]. The system uses Bluetooth instead of NFC and is based on public-key cryptography. Delegation is expressed in form of a digitally signed certificate, while our delegable access control tokens carrying policies are based on symmetric cryptography to minimize the communication overhead and to meet the bandwidth constrains of NFC[5]. Further, we consider the mobile platform security aspects of protecting authentication secrets, while the work by Bauer et al. considers more complicated (particularly, role-based) access control policies and usability issues.

## 8. CONCLUSION

We presented the first open security framework and instantiation for smartphone-based NFC-enabled immobilizers. Unlike the conventional, closed and proprietary immobilizer systems that suffer from various vulnerabilities, our open approach allows the independent evaluation of our solution by the research community. Our framework consists of a set of secure protocols (adapted from [22]) and a security architecture for the mobile platform. We analyze the security of the underlying protocols using automated formal verification tools. Moreover, we analyze the security of our architecture and discuss which objectives can be achieved using off-the-shelf secure hardware for mobile platforms. We show that available hardware allows remote issuing and remote revocation of electronic tokens, which cannot be achieved with classical (transponder-based) immobilizer systems. Further, we outline approaches to achieve more advanced security features, such as secure delegation and context-aware access control.

[5]While the nominal NFC transmission rate is 106 kpbs, the bandwidth usable in practice is only about 10 kbps [45].

## 9. REFERENCES


[1] MIFARE4Mobile.org. `http://mifare4mobile.org/files/1213/3283/4766/12-03-20_NFC_Ticketing_Europe_2012.pdf`, 2012.

[2] Near Field Communication Forum. `http://www.nfc-forum.org/home/`.

[3] M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *28th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'01)*. ACM, 2001.

[4] T. Alves and D. Felton. TrustZone: Integrated hardware and software security. *Information Quaterly*, 3(4), 2004.

[5] Arduino. `http://www.arduino.cc/`.

[6] A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuellar, P. Hankes Drielsma, P.-C. Heám, J. Mantovani, S. Mödersheim, D. von Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Viganò, and L. Vigneron. The AVISPA tool for the automated validation of internet security protocols and applications. In *17th International Conference on Computer Aided Verification (CAV'05)*. Springer, 2005.

[7] Atmel. Car access. `http://www.atmel.com/applications/automotive/car_access/default.aspx`.

[8] ATMEL. Automotive Compilation Volume 7 December 2010. `http://www.atmel.com/Images/atmel_autocompilation_vol7_dec2010.pdf`, 2010.

[9] Atmel. Open source immobilizer protocol stack. `http://www.atmel.com/dyn/products/tools_card.asp?tool_id=17197`, 2010. registration required.

[10] AVR cryptographic library. Set of cryptographic primitives for Atmel AVR microcontrollers. `https://www.das-labor.org/wiki/AVR-Crypto-Lib`.

[11] J. Azema and G. Fayad. M-Shield mobile security technology: Making wireless secure. Texas Instruments white paper, 2008. `http://focus.ti.com/pdfs/wtbu/ti_mshield_whitepaper.pdf`.

[12] L. Bauer, L. Cranor, R. W. Reeder, M. K. Reiter, and K. Vaniea. Comparing access-control technologies: A study of keys and smartphones. Technical report, 2007.

[13] L. Bauer, L. F. Cranor, M. K. Reiter, and K. Vaniea. Lessons learned from the deployment of a smartphone-based access-control system. In *3rd symposium on Usable privacy and security (SOUPS'07)*. ACM, 2007.

[14] L. Bauer, S. Garriss, J. M. McCune, M. K. Reiter, J. Rouse, and P. Rutenbar. Device-enabled authorization in the Grey system. In *8th International Conference on Information Security (ISC'05)*. Springer-Verlag, 2005.

[15] B. Blanchet. An efficient cryptographic protocol verifier based on Prolog rules. In *14th IEEE Computer Security Foundations Workshop (CSFW'01)*. IEEE Computer Society, 2001.

[16] B. Blanchet. From secrecy to authenticity in security protocols. In *9th International Static Analysis Symposium (SAS'02)*. Springer Verlag, 2002.

[17] C. Cremers. *Scyther - Semantics and Verification of Security Protocols*. Ph.D. dissertation, Eindhoven University of Technology, 2006.

[18] C. Cremers. Unbounded verification, falsification, and characterization of security protocols by pattern refinement. In *15th ACM conference on Computer and communications security (CCS'08)*. ACM, 2008.

[19] C. Cremers, P. Lafourcade, and P. Nadeau. Comparing state spaces in automatic protocol analysis. In *Formal to Practical Security*. Springer Berlin Heidelberg, 2009.

[20] CyanogenMod. http://www.cyanogenmod.com/.

[21] N. Dalal, J. Shah, K. Hisaria, and D. Jinwala. A comparative analysis of tools for verification of security protocols. *IJCNS*, 3(10):779–787, 2010.

[22] A. Dmitrienko, A.-R. Sadeghi, S. Tamrakar, and C. Wachsmann. SmartTokens: Delegable access control with NFC-enabled smartphones. In *5th International Conference on Trust & Trustworthy Computing (TRUST'12)*, 2012.

[23] D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–207, 1983.

[24] A. Francillon, B. Danev, and S. Čapkun. Relay attacks on passive keyless entry and start systems in modern cars. In *Network and Distributed System Security Symposium (NDSS)*, 2011.

[25] L. Francis, G. Hancke, K. Mayes, and K. Markantonakis. Practical NFC peer-to-peer relay attack using mobile phones. In *6th International Conference on Radio Frequency Identification: Security and Privacy Issues (RFIDSec'10)*. Springer-Verlag, 2010.

[26] Giesecke & Devrient Secure Flash Solutions. The Mobile Security Card SE 1.0 offers increased security. http://www.gd-sfs.com/the-mobile-security-card/mobile-security-card-se-1-0/.

[27] Google. http://www.google.com/wallet/.

[28] Google Wallet. http://www.google.com/wallet/, 2012.

[29] E. Haselsteiner and K. Breitfuß. Security in Near Field Communication (NFC). Strengths and weaknesses. In *Workshop on RFID Security*, 2006.

[30] J. Heyszl and F. Stumpf. Asymmetric cryptography in automotive access and immobilizer systems. 9th Embedded Security in Cars Conference, 2011.

[31] S. Indesteege, N. Keller, O. Dunkelman, E. Biham, and B. Preneel. A practical attack on KeeLoq. In *27th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'08)*. Springer-Verlag, 2008.

[32] International Organization for Standardization. *International Standard ISO/IEC 14443-4. Identification cards – Contactless integrated circuit cards – Proximity cards.*

[33] M. Kasper, T. Kasper, A. Moradi, and C. Paar. Breaking KeeLoq in a flash: On extracting keys at lightning speed. In *2nd International Conference on Cryptology in Africa (AFRICACRYPT'09)*. Springer, 2009.

[34] K. Kostiainen, J.-E. Ekberg, N. Asokan, and A. Rantala. On-board credentials with open provisioning. In *4th ACM Symposium on Information, Computer, and Communications Security (ASIACCS'09)*. ACM, 2009.

[35] K. Lemke, A.-R. Sadeghi, and C. Stüble. An open approach for designing secure electronic immobilizers. In *Information Security Practice and Experience (ISPEC'05)*, 2005.

[36] P. Lepek. Configurable, secure, open immobilizer implementation. In *Embedded Security in Cars*, 2010.

[37] R. Näätänen, O. Syssoeva, and R. Takegata. Automatic time perception in the human brain for intervals ranging from milliseconds to seconds. *Psychophysiology*, 41(4):660–663, 2004.

[38] NFC Shield. Near Field Communication interface for Arduino. http://www.seeedstudio.com/wiki/NFC_Shield.

[39] NFC World. Orange and Valeo demonstrate NFC car key concept, 2010. http://www.nfcworld.com/2010/10/07/34592/orange-and-valeo-demonstrate-nfc-car-key-concept/.

[40] NXP. Car access and immobilizers. http://www.nxp.com/products/automotive/car_access_immobilizers/.

[41] NXP. NXP and Continental demonstrate the world's first concept car embedding NFC at Mobile World Congress, 2011. http://www.nxp.com/news/press-releases/2011/02/nxp-and-continental-demonstrate-the-world-s-first-concept-car-embedding-nfc-at-mobile-world-congress.html.

[42] PN532 Near Field Communication (NFC) controller. NXP Semiconductors. http://www.nxp.com/products/identification_and_security/reader_ics/nfc_devices/series/PN532.html.

[43] Scientific American. Hack My Ride: Cyber Attack Risk on Car Computers , 2011. http://www.scientificamerican.com/article.cfm?id=hack-my-ride.

[44] Secure Element Evaluation Kit for the Android platform. http://code.google.com/p/seek-for-android/.

[45] S. Tamrakar, J.-E. Ekberg, and N. Asokan. Identity verification schemes for public transport ticketing with NFC phones. In *ACM workshop on Scalable Trusted Computing (STC'11)*. ACM, 2011.

[46] Telcred. secure offline access control with NFC. http://www.telcred.com/, 2012.

[47] Telecom. Deutsche Telekom and automotive supplier Continental demonstrated car keys, 2011. http://www.telekom.com/innovation/connectedcar/81840.

[48] S. Tillich and M. Wójcik. Security analysis of an open car immobilizer protocol stack. 10th International Conference on Applied Cryptograpy and Network Security (ACNS'12), 2012.

[49] Tyfone. Tyfone to license SideTap MicroSD NFC and Secure Element Card technologies to AboMem, 2011. http://tyfone.com/newsroom/?p=541.

[50] R. Verdult, F. Garcia, and J. Balasch. Gone in 360 seconds: Hijacking with Hitag2. In *21st USENIX Security Symposium*, 2012.

[51] ZDNet. Android malware numbers explode to 25,000 in June 2012. http://www.zdnet.com/android-malware-numbers-explode-to-25000-in-june-2012-7000001046/, 2012.