SmartTokens: Delegable Access Control with NFC-enabled Smartphones
(Full Version)

Alexandra Dmitrienko[1], Ahmad-Reza Sadeghi[2], Sandeep Tamrakar[3], and Christian Wachsmann[4]

[1] Fraunhofer SIT Darmstadt, Germany
alexandra.dmitrienko@trust.cased.de

[2] Technische Universität Darmstadt & Fraunhofer SIT Darmstadt, Germany
ahmad.sadeghi@trust.cased.de

[3] Aalto University School of Science, Finland
sandeep.tamrakar@aalto.fi

[4] Technische Universität Darmstadt (CASED), Germany
christian.wachsmann@trust.cased.de

Abstract. Today's smartphones and tablets offer compelling computing and storage capabilities enabling a variety of mobile applications with rich functionality. The integration of new interfaces, in particular near field communication (NFC) opens new opportunities for new applications and business models, as the most recent trend in industry for payment and ticketing shows. These applications require storing and processing security-critical data on smartphones, making them attractive targets for a variety of attacks. The state of the art to enhance platform security concerns outsourcing security-critical computations to hardware-isolated Trusted Execution Environments (TrEE). However, since these TrEEs are used by software running in commodity operating systems, malware could impersonate the software and use the TrEE in an unintended way. Further, existing NFC-based access control solutions for smartphones are either not public or based on strong assumptions that are hard to achieve in practice. We present the design and implementation of a generic access control system for NFC-enabled smartphones based on a multi-level security architecture for smartphones. Our solution allows users to delegate their access rights and addresses the bandwidth constraints of NFC. Our prototype captures electronic access to facilities, such as entrances and offices, and binds NFC operations to a software-isolated TrEE established on the widely used Android smartphone operating system. We provide a formal security analysis of our protocols and evaluated the performance of our solution.

1 Introduction

Modern smartphones are equipped with a variety of communication interfaces and enable mobile access to many different services, including Internet, web services, e-mail, multi-media entertainment, navigation and location-based services. The integration of additional communication interfaces, in particular near field communication (NFC) [35], greatly enlarges the application area of smart devices. NFC-based access control systems on smartphones and commercial NFC-based applications for ticketing and payment are particularly promoted by industry.

Electronic access control tokens for smartphones offer a variety of appealing features: they can be distributed and revoked remotely, delegated by users, and may support context-aware and time-limited access control policies. There are already some commercial systems on the market, including electronic hotel room keys [1,7,15] that are sent to the customer via SMS or email, and electronic car keys [14,40]. These applications require storing and processing security-critical data on smartphones, raising risks of being targeted by attacks. However, the security properties of current solutions are unclear, in particular because their design and implementation details are not publicly available and most operating systems for smartphones are vulnerable to malware [31,32].

A vast amount of research (such as in [16,25,11]) has been performed on hardening platform security based on secure hardware that is already available in many smartphones, such as M-Shield [3] and ARM TrustZone [2] on Nokia devices. Existing security hardware typically provides a trusted execution environment (TrEE) that enforces secure and isolated execution of small programs. However, currently available TrEEs are typically resource-constrained and prevent the implementation of important security functionalities, such as secure user interfaces [11]. Further, even the verification of X.509 certificates within a TrEE is challenging and requires a number of subsequent invocations of the TrEE [11], which introduces additional performance overhead. Hence, practical security architectures built on top of existing TrEEs must rely on additional trusted components in the operating system.

The secure implementation of security critical NFC-based applications on smartphones, such as electronic payment, ticketing and access control systems, requires the underlying security architecture to isolate trusted and untrusted components to prevent leakage and unintended manipulation of security-critical data, such as authentication secrets. Furthermore, the underlying protocol design must consider the bandwidth constraints of NFC.

Contribution and Outline. We present the design and implementation of an access control system for NFC-enabled smartphones. The unique feature of our scheme is that users can delegate (part of) their access rights to other users without contacting a central token issuer. Our contributions are as follows:

Multi-level platform security architecture. Our SmartToken application runs on top of a security architecture that protects the underlying authentication secrets. The architecture combines a hardware-supported trusted execution environment (TrEE) to handle cryptographic keys with software-based isolation of trusted code controlling access to the TrEE (Section 2). The architecture provides a two-line defense against software attacks and a trade-off between security and resource constraints of common security hardware.

Delegatable SmartToken system. We present a generic token-based access control system for NFC-enabled smartphones that, in contrast to previous solutions, supports delegation of access rights without contacting a central token issuer and that addresses the bandwidth constraints of NFC (Section 3). Our solution is suitable for various applications, ranging from access control solutions for digital objects, such as electronic documents, to physical resources like rooms or cars. Further, we prove the security properties of our system (Section 4).

Reference implementation. We instantiate the SmartToken system for electronic access control tokens (Section 5). The implementation is based on TrustDroid [10], which extends the widely used Android smartphone operating system with software-based isolation of trusted and untrusted compartments. Further, we conceptually consider binding NFC operations to a hardware-based trusted execution environment (TrEE).

2 Multi-level Security Architecture

In this section we describe our multi-level security platform architecture, which we deploy to protect user credentials on the device.

2.1 Model and Requirement Analysis

In the following, we describe our system model, formulate security objectives and requirements, and define our trust and adversary model.

System Model. We consider mobile platforms that (1) run untrusted code, such as user applications downloaded from untrusted sources, (2) store user credentials, such as user passwords and cryptographic secrets that are used in cryptographic protocols to authenticate the user to some service provider, and that (3) run security-critical code that, e.g., operates on security sensitive data, such as cryptographic keys.

Security Objectives and Requirements. The objective of our overall solution is to prevent the adversary from being able to authenticate to a service provider. While attacks against the authentication protocols must be prevented by protocol design (Section 3), the platform security architecture must ensure (1) that the adversary cannot access user credentials stored on the platform and (2) that he cannot exploit or modify code using them. More specifically, the objective of the platform security architecture is to ensure *confidentiality* of and to enforce *access control* to credentials, i.e., that any application on the platform can use only those credentials that have been created or enrolled by this application before. This results in the following security requirements:

- *Confidentiality of user credentials:* User credentials created or enrolled by security-critical code must not be accessible by untrusted and other security-critical code while stored or used on the platform.
- *Code isolation:* Security-critical code that processes user credentials must be isolated from untrusted and other security-critical code on the platform.

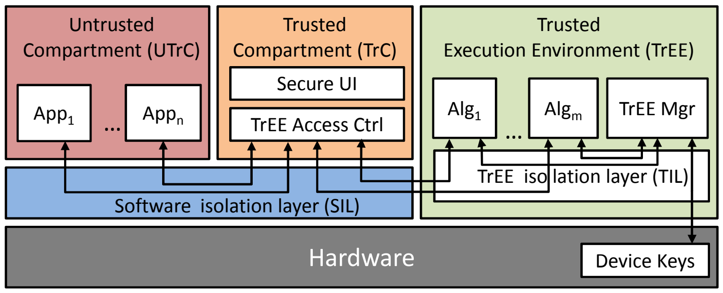

Untrusted Compartment (UTrC)
App1
...
Appn
Trusted Compartment (TrC)
Secure UI
TrEE Access Ctrl
Trusted Execution Environment (TrEE)
Alg1
...
Algm
TrEE Mgr
TrEE isolation layer (TIL)
Software isolation layer (SIL)
Hardware
Device Keys

Fig. 1. Generic multi-level platform security architecture

- *Code access control:* Only authorized code instances must be able to invoke execution of security-critical code that has access to user credentials.
- *Code integrity:* The integrity of security-critical code that has access to user credentials and the integrity of untrusted code that can invoke security-critical code must be preserved.

Trust and Adversary Model. The adversary can perform software attacks and install, modify or compromise arbitrary code on the device. However, he cannot access or modify the hardware of the platform and its trusted computing base, i.e., the code that enforces access control or isolation on the device.

2.2 Generic Security Architecture

Figure 1 illustrates our multi-level security platform architecture. At a high level, the execution environment of the device is split into three isolated compartments (Figure 1): an untrusted compartment UTrC, a trusted compartment TrC and a trusted execution environment TrEE. TrEE is isolated from the rest of the system by the underlying security hardware and protected against software-based attacks. However, TrEE is a resource-constrained component. UTrC is free of strict resource constraints and isolated from the untrusted compartment by means of software, which is less reliable compared to hardware-based isolation since isolation can be broken upon successful compromise of the software isolation layer. TrEE is used to run secure code that operates on user credentials, while TrC handles system components that exceed the capabilities of TrEE. Particularly, TrC provides a secure user interface SecureUI, which is used to collect security-sensitive user input (such as passwords) or to display output. Further, TrC includes the TrAC component, which enforces access control to the code running within TrEE.[5]

Security-sensitive applications are split into an untrusted host application App_i running in UTrC and one or more security-sensitive algorithms Alg_j that

[5] Note, that both SecureUI and TrAC have been shown to exceed resource-constraints of commodity TrEEs [27,11].

are executed by TrEE and that can be invoked by App_i when necessary (Figure 1). Communication between App_i and the algorithms Alg_j within TrEE is mediated by TrAC, which ensures that App_i can communicate only to those Alg_j App_i is supposed to communicate to. The software isolation layer verifies the integrity of host applications (e.g., by comparing the hash digest of the application binary to a reference value or by verifying the application's signature upon application loading) and reports it to TrAC, which then grants or denies access to the TrEE based on the integrity of the host application.

Algorithms executed within TrEE may belong to different host applications and thus are mutually untrusted. Thus, they are isolated from each other, which is enforced by the TrEE isolation layer. Furthermore, TrEE includes the TrEEMgr component, which has direct access to platform keys stored in secure memory and that provides a sealing/unsealing functionality to the algorithms. More specifically, TrEEMgr encrypts/decrypts user credentials with a key that is cryptographically bound to the platform key and the identity of the algorithm (such as the hash digest of its binary).

The trusted computing base of our architecture includes the software isolation layer, trusted compartment, TrEE isolation layer and the TrEE manager.

Fulfillment of the Security Requirements. Our security architecture achieves the security requirements described in Section 2.1: confidentiality of user credentials is ensured by a trusted TrEEMgr component, which stores user credentials only in an encrypted form and such that they can be decrypted only by authorized algorithms (sealing). Isolation of security-critical code from untrusted code is enforced by a hardware-isolated TrEE, while isolation from other security-critical code is provided by the trusted isolation layer within the TrEE. Access control to security-critical code is enforced by the TrAC component. The integrity of security-critical code is ensured by the sealing functionality of TrEEMgr, which ensures that user credentials can be decrypted only if the integrity of the algorithm is preserved. Integrity of untrusted code is enforced by the software isolation layer, which measures and verifies the application integrity upon loading the application and denies access to TrC if the application has been modified.

Our security architecture provides higher security guarantees than approaches using pure software-based isolation and solutions that rely only on hardware-based TrEEs (such as [28,19,25,11]), where the secure user interface and access control to the TrEE is typically outsourced to the untrusted commodity operating system that is vulnerable to various attacks.

2.3 Architecture Instantiation

Our security architecture can be instantiated based on different types of security hardware and different approaches to software-based isolation. For instance, the TrEE can be instantiated using ARM TrustZone [2], M-Shield [3], embedded or removable secure elements, such as SIM cards, universal integrated circuit cards (UICC), or secure memory cards (SMC). A detailed discussion of different types of hardware security modules can be found in [34].

Software-enforced isolation can be implemented based on virtualization technology or hardened operating systems that enforce domain isolation by mandatory access control. Examples include the OKL4 microvisor [23], domain isolation based on security kernels [42], and the TrustDroid [10] security enhancement of the Android operating system.

Instantiation for Android doevices. We aim to instantiate our multi-level security architecture on Android-powered devices, since Android is the most popular smartphone operating system worldwide [18] and first NFC-enabled Android devices appear on the market. On the other hand, most secure NFC-based applications target Nokia smartphones, most probably since NFC-enabled Nokia smartphones are already available for some time and equipped with secure hardware. At the time of writing, we are not aware of any instantiation of a secure access control application for Android devices and aim to fill this gap.

To enforce the software isolation required by our architecture, we could follow the virtualization approach, e.g., based on the OKL4 microvisor that can run multiple instances of L4Android, as well as native applications. However, as supported by OKL4-based developments [17], a number of challenges has to be solved with regard to performance, power consumption and drivers portability before virtualization approaches become a practical solution for mobile devices. Thus, we opted for a more practical solution and adopted the TrustDroid security extensions [10] to enforce isolation.

TrustDroid applies a coloring approach to isolation that has its origins in information-flow theory [36]. Particularly, it uses the concept of application identifiers on Android and colors (tags) applications and application data upon application installation. Based on the assigned colors, TrustDroid organizes applications along with their data in logical domains. At runtime, communication across domains is prevented by means of mandatory access control applied on all communication channels between applications, including inter-process communication (IPC) calls, Linux sockets, file system access and local network connections. We extended TrustDroid to form isolated domains and enabled inter-domain communication through well-defined interfaces, as required by our architecture. The details of our implementation can be found in Section 5.

3 Smart Token System

We present a generic access control system that allows users to maintain their access credentials for different resources on their smartphone. One of the key features of our scheme is that users can delegate their credentials to other users without contacting a central token issuer. The system is applicable to various applications, ranging from access control solutions for digital objects, such as electronic documents, to physical resources like rooms and cars.

3.1 Overview

The entities in our system are at least a token issuer $\mathcal{I}$, a set of resources $\mathcal{R}$ (such as electronic documents or doors) and a set of users $\mathcal{U}$ (Figure 2). We denote the

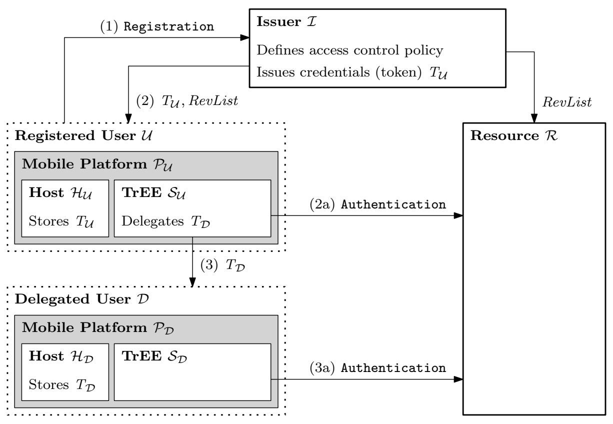

(1) Registration
Issuer $\mathcal{I}$
Defines access control policy
Issues credentials (token) $T_{\mathcal{U}}$
(2) $T_{\mathcal{U}}$, *RevList*
RevList
Registered User $\mathcal{U}$
Mobile Platform $\mathcal{P}_{\mathcal{U}}$
Host $\mathcal{H}_{\mathcal{U}}$
Stores $T_{\mathcal{U}}$
TrEE $\mathcal{S}_{\mathcal{U}}$
Delegates $T_{\mathcal{D}}$
(2a) Authentication
Resource $\mathcal{R}$
(3) $T_{\mathcal{D}}$
Delegated User $\mathcal{D}$
Mobile Platform $\mathcal{P}_{\mathcal{D}}$
Host $\mathcal{H}_{\mathcal{D}}$
Stores $T_{\mathcal{D}}$
TrEE $\mathcal{S}_{\mathcal{D}}$
(3a) Authentication

Fig. 2. SmartToken system overview

adversary with $\mathcal{A}$. Each $\mathcal{U}$ possesses a mobile platform $\mathcal{P}_{\mathcal{U}}$, such as a smartphone or tablet. $\mathcal{I}$ is a central authority that defines which $\mathcal{U}$ is allowed to access which $\mathcal{R}$. Further, $\mathcal{I}$ issues credentials (SmartTokens) $T_{\mathcal{U}}$ to each $\mathcal{U}$, which are used later by $\mathcal{U}$ to authenticate to $\mathcal{R}$. We distinguish between registered users and delegated users. A registered user $\mathcal{U}$ can delegate his token $T_{\mathcal{U}}$ to a delegated user $\mathcal{D}$, while a delegated user $\mathcal{D}$ cannot delegate his token $T_{\mathcal{D}}$.

Objectives. The objectives of our solution are as follows:

– *Access control.* Access to a resource $\mathcal{R}$ is granted only (1) to a registered user $\mathcal{U}$, who got a token $T_{\mathcal{U}}$ for $\mathcal{R}$ from issuer $\mathcal{I}$, and (2) to a delegated user $\mathcal{D}$, who got a token $T_{\mathcal{D}}$ for $\mathcal{R}$ from a registered user $\mathcal{U}$ with $T_{\mathcal{U}}$ for $\mathcal{R}$.
– *Delegation.* Issuer $\mathcal{I}$ can allow registered users to delegate (share) their tokens with other users.
– *Revocation.* Issuer $\mathcal{I}$ can revoke tokens of regular and/or delegated users. Revoking token $T_{\mathcal{U}}$ of a registered user $\mathcal{U}$ automatically revokes all delegated tokens $T_{\mathcal{D}}$ based on $T_{\mathcal{U}}$.

Note that our scheme provides basic protection against denial-of-service attacks that permanently prevent a user from using the SmartToken scheme. However, since the focus of this paper is delegatable authentication for NFC-enabled smartphones, we did not consider countermeasures against denial of-service attacks.

Protocols. Our scheme is composed of the following protocols:

– *System initialization:* Issuer $\mathcal{I}$ generates its authentication secrets and encryption keys. Moreover, $\mathcal{I}$ generates and initializes each resource $\mathcal{R}$ with an authentication secret and encryption key.
– *User registration:* User $\mathcal{U}$ registers its mobile platform $\mathcal{P}_{\mathcal{U}}$ with $\mathcal{I}$ and becomes a registered user.

– *Token issuing:* $\mathcal{I}$ generates and sends the authentication key, the delegation key and token $T_{\mathcal{U}}$ to the mobile platform $\mathcal{P}_{\mathcal{U}}$ of a registered user $\mathcal{U}$.
– *Token delegation:* A registered user $\mathcal{U}$ delegates its smart token (its access rights) to a user $\mathcal{D}$, who then becomes a delegated user.
– *User authentication:* $\mathcal{U}$ or $\mathcal{D}$ authenticate to $\mathcal{R}$. Access to $\mathcal{R}$ is granted or denied based on the result of the authentication protocol.
– *Token and user revocation:* $\mathcal{I}$ revokes one or all tokens of $\mathcal{U}$ by updating the revocation list *RevList* on each $\mathcal{R}$.

Our scheme is inspired by Kerberos [30], which is a widely deployed and extensively analyzed authentication protocol. Kerberos provides strong authentication for client/server applications based on symmetric cryptography. Our protocols follow a similar approach to distribute authentication secrets with tokens issued by a key distribution center (KDC), which corresponds to the issuer in our scheme. However, in contrast to Kerberos our scheme enables delegation of tokens by clients (mobile devices) without contacting the KDC. Further, tokens are bound to the identity and the platform of their user by means of a one-time password and a device-specific platform key, respectively.

Trust Model and Assumptions. We assume that each registered user $\mathcal{U}$ and each delegated user $\mathcal{D}$ possesses a mobile platform $\mathcal{P}$, which consists of an untrusted operating environment (host) $\mathcal{H}$ and a trusted execution environment (TrEE) $\mathcal{S}$ (Figure 2). In Section 5, we show how the TrEE can be implemented based on an isolated trusted software compartment. Further, we assume issuer $\mathcal{I}$, resource $\mathcal{R}$ and $\mathcal{S}$ to be trusted. Moreover, we assume that an authentic and confidential out-of-band channel between $\mathcal{I}$ and $\mathcal{U}$ is available once before the user registration protocol, and between $\mathcal{U}$ and $\mathcal{D}$ once before the token delegation protocol. Note that this is very natural since in many access control scenarios users typically have to prove their identity (e.g., by showing their identity card) to $\mathcal{I}$ during registration and/or will get a personal welcome letter with their access credentials from $\mathcal{I}$. Furthermore, $\mathcal{S}$ provides countermeasures against dictionary attacks.

Adversary Model. We consider adversaries $\mathcal{A}$ that have full control over the communication between $\mathcal{I}$, $\mathcal{R}$, $\mathcal{U}$ and $\mathcal{D}$, which means that $\mathcal{A}$ can eavesdrop, modify, insert, delete and re-route protocol messages.[6] Further, $\mathcal{A}$ can compromise the untrusted part $\mathcal{H}$ of the user's mobile platform $\mathcal{P}$ and gain access to all information stored in $\mathcal{H}$. However, as mentioned in assumptions, $\mathcal{A}$ cannot compromise issuer $\mathcal{I}$, resource $\mathcal{R}$ or TrEE $\mathcal{S}$ of $\mathcal{P}$. In particular, $\mathcal{A}$ cannot change the functionality of $\mathcal{S}$ and $\mathcal{A}$ cannot obtain any secret information stored in $\mathcal{S}$.

3.2 Notation and Preliminaries

We denote with $a \in_R A$ the uniform sampling of an element a from a set A. Let A be a probabilistic algorithm. Then $y \leftarrow \mathsf{A}(x)$ means that on input x, algorithm

[6] Note that we exclude relay attacks since the focus of this paper is delegatable authentication for NFC-enabled smartphones. Relay attacks can be mitigated by distance bounding techniques, which can be integrated into our scheme.

A assigns its output to variable y. Probability $\epsilon(l)$ is called *negligible* if for all polynomials $f(\,)$ it holds that $\epsilon(l) \leq 1/f(l)$ for all sufficiently large l. Further, ID_X is the unique identifier, sk_X the secret key, and pk_X the public key of entity X, respectively.

Encryption Schemes. An encryption scheme ES is a tuple of algorithms $(\mathsf{Genkey}, \mathsf{Enc}, \mathsf{Dec})$ where Genkey is the key generation, Enc is the encryption and Dec is the decryption algorithm. A public-key encryption scheme is said to be CPA-secure [21,4] if every probabilistic polynomial time (p.p.t.) adversary $\mathcal{A}$ has at most negligible advantage of winning the following security experiment: an algorithm $\mathcal{C}_{sk}^{\mathrm{CPA}}$ (CPA-challenger), generates an encryption key pk and decryption key sk using $\mathsf{Genkey}(1^l)$, chooses $b \in_R \{0,1\}$, encrypts $c_b \leftarrow \mathsf{Enc}(pk; m_b)$ and returns c_b to $\mathcal{A}$. Eventually, $\mathcal{A}$ must return a bit b' that indicates whether c_b encrypts m_0 or m_1. $\mathcal{A}$ wins if $b' = b$. Note that for symmetric encryption schemes $sk = pk$.

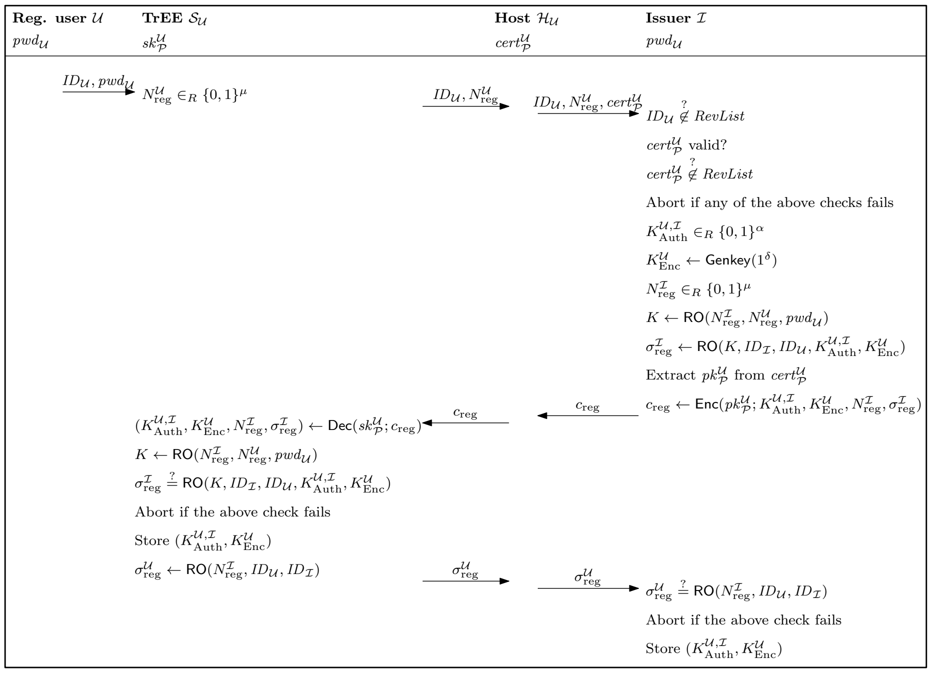
Random Oracles. A random oracle RO [6] is an oracle that responds with a random output to each given input. More precisely, RO starts with an empty look-up table Γ. When queried with input m, RO first checks if it already knows a value $\Gamma[m]$. If this is not the case, RO chooses $r \in_R \{0,1\}^\alpha$ and updates Γ such that $\Gamma[m] = r$. Finally, RO returns $\Gamma[m]$. Random oracles model the ideal security properties of cryptographic hash functions.

Note that our protocols use the MAC-then-encrypt paradigm [5], where for a given plaintext m, first the message digest $\sigma = \mathsf{RO}(m)$ is computed and then (m, σ) is encrypted with a CPA-secure encryption scheme.

3.3 Protocol Specification

System Initialization. Each mobile platform $\mathcal{P}$ has a unique platform key pair $(sk_{\mathcal{P}}, pk_{\mathcal{P}})$, where $sk_{\mathcal{P}}$ is only known to trusted execution environment (TrEE) $\mathcal{S}$ of platform $\mathcal{P}$. Further, host $\mathcal{H}$ of $\mathcal{P}$ stores a certificate $cert_{\mathcal{P}}$ issued by, e.g., the platform manufacturer, which contains $pk_{\mathcal{P}}$ and attests that $pk_{\mathcal{P}}$ is the public key of a genuine TrEE $\mathcal{S}$ and that $sk_{\mathcal{P}}$ is securely stored in and never leaves $\mathcal{S}$. Issuer $\mathcal{I}$ initializes the revocation list $RevList \leftarrow \emptyset$ and each resource $\mathcal{R}$ with $RevList$, a resource-specific authentication key $K_{\mathrm{Auth}}^{\mathcal{R}}$ and a resource-specific encryption/decryption key $K_{\mathrm{Enc}}^{\mathcal{R}}$.

User Registration. When a user $\mathcal{U}$ wants to register, $\mathcal{I}$ sends a new one-time password $pwd_{\mathcal{U}}$ to $\mathcal{U}$ over an authentic and confidential out-of-band channel. After that, $\mathcal{U}$ can register as follows (Figure 3): $\mathcal{U}$ sends its identifier $ID_{\mathcal{U}}$ and $pwd_{\mathcal{U}}$ to TrEE $\mathcal{S}_{\mathcal{U}}$ of its mobile platform $\mathcal{P}_{\mathcal{U}} = (\mathcal{H}_{\mathcal{U}}, \mathcal{S}_{\mathcal{U}})$. Then $\mathcal{S}_{\mathcal{U}}$ sends $ID_{\mathcal{U}}$ and a random $N_{\mathrm{reg}}^{\mathcal{U}}$ to host $\mathcal{H}_{\mathcal{U}}$, which sends both values and the platform certificate $cert_{\mathcal{P}}^{\mathcal{U}}$ to $\mathcal{I}$. Next, $\mathcal{I}$ verifies $cert_{\mathcal{P}}^{\mathcal{U}}$ and generates a new authentication secret $K_{\mathrm{Auth}}^{\mathcal{U},\mathcal{I}}$ and an encryption/decryption key $K_{\mathrm{Enc}}^{\mathcal{U}}$ for $\mathcal{U}$, which are used later in the token issuing protocol. Further, $\mathcal{I}$ derives a temporary authentication secret K



Reg. user $\mathcal{U}$
$pwd_{\mathcal{U}}$
TrEE $\mathcal{S}_{\mathcal{U}}$
$sk^{\mathcal{U}}_{\mathcal{P}}$
Host $\mathcal{H}_{\mathcal{U}}$
$cert^{\mathcal{U}}_{\mathcal{P}}$
Issuer $\mathcal{I}$
$pwd_{\mathcal{U}}$
$ID_{\mathcal{U}}, pwd_{\mathcal{U}}$
$N^{\mathcal{U}}_{\text{reg}} \in_R \{0,1\}^{\mu}$
$ID_{\mathcal{U}}, N^{\mathcal{U}}_{\text{reg}}$
$ID_{\mathcal{U}}, N^{\mathcal{U}}_{\text{reg}}, cert^{\mathcal{U}}_{\mathcal{P}}$
$ID_{\mathcal{U}} \overset{?}{\notin} RevList$
$cert^{\mathcal{U}}_{\mathcal{P}}$ valid?
$cert^{\mathcal{U}}_{\mathcal{P}} \overset{?}{\notin} RevList$
Abort if any of the above checks fails
$K^{\mathcal{U},\mathcal{I}}_{\text{Auth}} \in_R \{0,1\}^{\alpha}$
$K^{\mathcal{U}}_{\text{Enc}} \leftarrow \mathsf{Genkey}(1^{\delta})$
$N^{\mathcal{I}}_{\text{reg}} \in_R \{0,1\}^{\mu}$
$K \leftarrow \mathsf{RO}(N^{\mathcal{I}}_{\text{reg}}, N^{\mathcal{U}}_{\text{reg}}, pwd_{\mathcal{U}})$
$\sigma^{\mathcal{I}}_{\text{reg}} \leftarrow \mathsf{RO}(K, ID_{\mathcal{I}}, ID_{\mathcal{U}}, K^{\mathcal{U},\mathcal{I}}_{\text{Auth}}, K^{\mathcal{U}}_{\text{Enc}})$
Extract $pk^{\mathcal{U}}_{\mathcal{P}}$ from $cert^{\mathcal{U}}_{\mathcal{P}}$
$c_{\text{reg}} \leftarrow \mathsf{Enc}(pk^{\mathcal{U}}_{\mathcal{P}}; K^{\mathcal{U},\mathcal{I}}_{\text{Auth}}, K^{\mathcal{U}}_{\text{Enc}}, N^{\mathcal{I}}_{\text{reg}}, \sigma^{\mathcal{I}}_{\text{reg}})$
c_{reg}
c_{reg}
$(K^{\mathcal{U},\mathcal{I}}_{\text{Auth}}, K^{\mathcal{U}}_{\text{Enc}}, N^{\mathcal{I}}_{\text{reg}}, \sigma^{\mathcal{I}}_{\text{reg}}) \leftarrow \mathsf{Dec}(sk^{\mathcal{U}}_{\mathcal{P}}; c_{\text{reg}})$
$K \leftarrow \mathsf{RO}(N^{\mathcal{I}}_{\text{reg}}, N^{\mathcal{U}}_{\text{reg}}, pwd_{\mathcal{U}})$
$\sigma^{\mathcal{I}}_{\text{reg}} \overset{?}{=} \mathsf{RO}(K, ID_{\mathcal{I}}, ID_{\mathcal{U}}, K^{\mathcal{U},\mathcal{I}}_{\text{Auth}}, K^{\mathcal{U}}_{\text{Enc}})$
Abort if the above check fails
Store $(K^{\mathcal{U},\mathcal{I}}_{\text{Auth}}, K^{\mathcal{U}}_{\text{Enc}})$
$\sigma^{\mathcal{U}}_{\text{reg}} \leftarrow \mathsf{RO}(N^{\mathcal{I}}_{\text{reg}}, ID_{\mathcal{U}}, ID_{\mathcal{I}})$
$\sigma^{\mathcal{U}}_{\text{reg}}$
$\sigma^{\mathcal{U}}_{\text{reg}}$
$\sigma^{\mathcal{U}}_{\text{reg}} \overset{?}{=} \mathsf{RO}(N^{\mathcal{I}}_{\text{reg}}, ID_{\mathcal{U}}, ID_{\mathcal{I}})$
Abort if the above check fails
Store $(K^{\mathcal{U},\mathcal{I}}_{\text{Auth}}, K^{\mathcal{U}}_{\text{Enc}})$

Fig. 3. User registration protocol

from $pwd_{\mathcal{U}}$, computes authenticator $\sigma^{\mathcal{I}}_{\text{reg}}$ for $K^{\mathcal{U},\mathcal{I}}_{\text{Auth}}$ and $K^{\mathcal{U}}_{\text{Enc}}$, encrypts both keys and $\sigma^{\mathcal{I}}_{\text{reg}}$ with the platform key $pk^{\mathcal{U}}_{\mathcal{P}}$ of $\mathcal{S}_{\mathcal{U}}$, and sends the resulting ciphertext c_{reg} to $\mathcal{S}_{\mathcal{U}}$. On receipt of c_{reg} $\mathcal{S}_{\mathcal{U}}$ decrypts c_{reg} and, in case the verification of σ_{reg} is successful, stores $(K^{\mathcal{U},\mathcal{I}}_{\text{Auth}}, K^{\mathcal{U}}_{\text{Enc}})$. Then, $\mathcal{S}_{\mathcal{U}}$ sends authenticator $\sigma^{\mathcal{U}}_{\text{reg}}$ to $\mathcal{I}$, which verifies $\sigma^{\mathcal{U}}_{\text{reg}}$ and, in case the verification was successful, stores $(K^{\mathcal{U},\mathcal{I}}_{\text{Auth}}, K^{\mathcal{U}}_{\text{Enc}})$. In case $\mathcal{I}$ already stores an authentication secret and encryption/decryption key for $\mathcal{U}$, $\mathcal{I}$ deletes the old keys and stores the newly generated ones.

Token Issuing. The token issuing protocol is depicted in Figure 4: user $\mathcal{U}$ initiates the protocol at TrEE $\mathcal{S}_{\mathcal{U}}$ of its mobile platform $\mathcal{P}_{\mathcal{U}}$, which then sends $ID_{\mathcal{U}}$ and a random N_{iss} to $\mathcal{I}$. Next, $\mathcal{I}$ generates authentication secret $K^{\mathcal{U},\mathcal{R}}_{\text{Auth}}$, delegation secret $K^{\mathcal{U}}_{\text{Del}}$ and token $T_{\mathcal{U}}$ for $\mathcal{U}$, which are used later by $\mathcal{U}$ in the authentication and delegation protocols. Further, $\mathcal{I}$ computes σ_{iss} that authenticates $K^{\mathcal{U},\mathcal{R}}_{\text{Auth}}$, $K^{\mathcal{U}}_{\text{Del}}$ and $T_{\mathcal{U}}$, encrypts these keys, $T_{\mathcal{U}}$ and σ_{iss} with $K^{\mathcal{U}}_{\text{Enc}}$, and sends the resulting ciphertext c_{iss} to host $\mathcal{H}_{\mathcal{U}}$ of $\mathcal{P}_{\mathcal{U}}$, which passes c_{iss} to $\mathcal{S}_{\mathcal{U}}$. Next, $\mathcal{S}_{\mathcal{U}}$ decrypts c_{iss} and, in case the verification of σ_{iss} is successful, stores $(K^{\mathcal{U},\mathcal{R}}_{\text{Auth}}, K^{\mathcal{U}}_{\text{Del}})$. Eventually, $\mathcal{S}_{\mathcal{U}}$ sends $T_{\mathcal{U}}$ to $\mathcal{H}_{\mathcal{U}}$.

Authentication of Registered Users. The authentication protocol for registered users is depicted in Figure 5: user $\mathcal{U}$ initiates the protocol at TrEE $\mathcal{S}_{\mathcal{U}}$ of its mobile platform $\mathcal{P}_{\mathcal{U}}$, which sends an authentication request to resource $\mathcal{R}$.

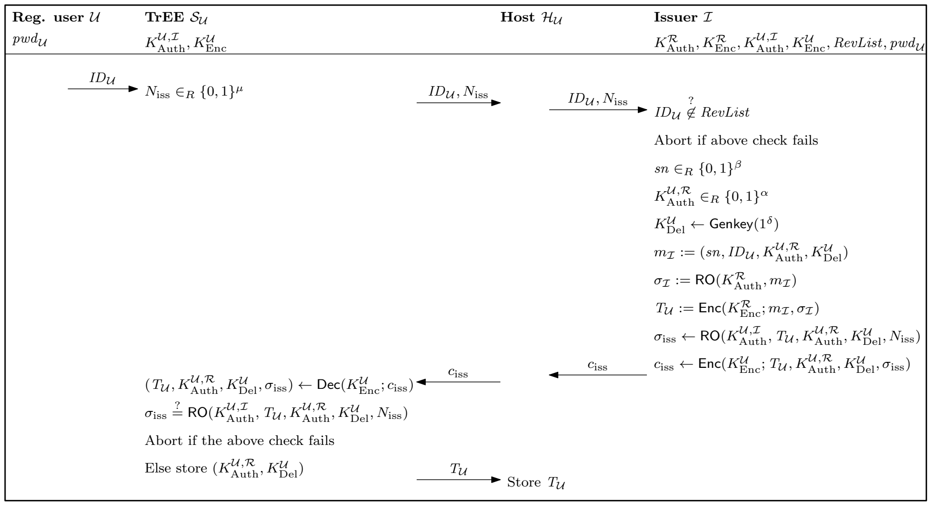

Reg. user $\mathcal{U}$
TrEE $\mathcal{S}_\mathcal{U}$
Host $\mathcal{H}_\mathcal{U}$
Issuer $\mathcal{I}$
$pwd_\mathcal{U}$
$K^{\mathcal{U},\mathcal{I}}_{\text{Auth}}, K^{\mathcal{U}}_{\text{Enc}}$
$K^{\mathcal{R}}_{\text{Auth}}, K^{\mathcal{R}}_{\text{Enc}}, K^{\mathcal{U},\mathcal{I}}_{\text{Auth}}, K^{\mathcal{U}}_{\text{Enc}}, RevList, pwd_\mathcal{U}$
$ID_\mathcal{U}$
$N_{\text{iss}} \in_R \{0,1\}^\mu$
$ID_\mathcal{U}, N_{\text{iss}}$
$ID_\mathcal{U}, N_{\text{iss}}$
$ID_\mathcal{U} \overset{?}{\notin} RevList$
Abort if above check fails
$sn \in_R \{0,1\}^\beta$
$K^{\mathcal{U},\mathcal{R}}_{\text{Auth}} \in_R \{0,1\}^\alpha$
$K^{\mathcal{U}}_{\text{Del}} \leftarrow \mathsf{Genkey}(1^\delta)$
$m_\mathcal{I} := (sn, ID_\mathcal{U}, K^{\mathcal{U},\mathcal{R}}_{\text{Auth}}, K^{\mathcal{U}}_{\text{Del}})$
$\sigma_\mathcal{I} := \mathsf{RO}(K^{\mathcal{R}}_{\text{Auth}}, m_\mathcal{I})$
$T_\mathcal{U} := \mathsf{Enc}(K^{\mathcal{R}}_{\text{Enc}}; m_\mathcal{I}, \sigma_\mathcal{I})$
$\sigma_{\text{iss}} \leftarrow \mathsf{RO}(K^{\mathcal{U},\mathcal{I}}_{\text{Auth}}, T_\mathcal{U}, K^{\mathcal{U},\mathcal{R}}_{\text{Auth}}, K^{\mathcal{U}}_{\text{Del}}, N_{\text{iss}})$
c_{iss}
$c_{\text{iss}} \leftarrow \mathsf{Enc}(K^{\mathcal{U}}_{\text{Enc}}; T_\mathcal{U}, K^{\mathcal{U},\mathcal{R}}_{\text{Auth}}, K^{\mathcal{U}}_{\text{Del}}, \sigma_{\text{iss}})$
c_{iss}
$(T_\mathcal{U}, K^{\mathcal{U},\mathcal{R}}_{\text{Auth}}, K^{\mathcal{U}}_{\text{Del}}, \sigma_{\text{iss}}) \leftarrow \mathsf{Dec}(K^{\mathcal{U}}_{\text{Enc}}; c_{\text{iss}})$
$\sigma_{\text{iss}} \overset{?}{=} \mathsf{RO}(K^{\mathcal{U},\mathcal{I}}_{\text{Auth}}, T_\mathcal{U}, K^{\mathcal{U},\mathcal{R}}_{\text{Auth}}, K^{\mathcal{U}}_{\text{Del}}, N_{\text{iss}})$
Abort if the above check fails
Else store $(K^{\mathcal{U},\mathcal{R}}_{\text{Auth}}, K^{\mathcal{U}}_{\text{Del}})$
$T_\mathcal{U}$
Store $T_\mathcal{U}$

Fig. 4. Token issuing protocol

Then $\mathcal{R}$ sends its identifier $ID_\mathcal{R}$ and a random N to $\mathcal{S}_\mathcal{U}$, which replies with $\sigma_\mathcal{U}$ to $\mathcal{H}_\mathcal{U}$ that sends $(\sigma_\mathcal{U}, T_\mathcal{U})$ to $\mathcal{R}$. Next, $\mathcal{R}$ decrypts $T_\mathcal{U}$ with $K^{\mathcal{R}}_{\text{Enc}}$ to obtain $K^{\mathcal{U},\mathcal{R}}_{\text{Auth}}$, verifies $\sigma_\mathcal{I}$ and $\sigma_\mathcal{U}$ using $K^{\mathcal{R}}_{\text{Auth}}$ and $K^{\mathcal{U},\mathcal{R}}_{\text{Auth}}$, respectively, and accepts only if both verifications are successful. Otherwise, $\mathcal{R}$ rejects.

Token Delegation. Registered user $\mathcal{U}$ and delegated user $\mathcal{D}$ establish a new one-time secret $pwd_\mathcal{D}$ over an authentic and confidential out-of-band-channel. Then, the token delegation protocol (Figure 6) starts: $\mathcal{D}$ sends its identifier $ID_\mathcal{D}$ and $pwd_\mathcal{D}$ to TrEE $\mathcal{S}_\mathcal{D}$ of its mobile platform $\mathcal{P}_\mathcal{D} = (\mathcal{S}_\mathcal{D}, \mathcal{H}_\mathcal{D})$, which then sends a random $N^{\mathcal{D}}_{\text{del}}$ to host $\mathcal{H}_\mathcal{D}$ that passes $(ID_\mathcal{D}, N^{\mathcal{D}}_{\text{del}})$ together with the platform certificate $cert^{\mathcal{D}}_{\mathcal{P}}$ of $\mathcal{P}_\mathcal{D}$ to host $\mathcal{H}_\mathcal{U}$ of the registered user's mobile platform $\mathcal{P}_\mathcal{U} = (\mathcal{S}_\mathcal{U}, \mathcal{H}_\mathcal{U})$. $\mathcal{H}_\mathcal{U}$ then sends $(ID_\mathcal{D}, N^{\mathcal{D}}_{\text{del}}, cert^{\mathcal{D}}_{\mathcal{P}})$ and token $T_\mathcal{U}$ of $\mathcal{U}$ to $\mathcal{S}_\mathcal{U}$. Next, $\mathcal{S}_\mathcal{U}$ verifies $cert^{\mathcal{D}}_{\mathcal{P}}$, generates authentication secret $K^{\mathcal{D}}_{\text{Auth}}$ for $\mathcal{D}$, computes authenticator $\sigma_\mathcal{U}$ and delegated token $T_\mathcal{D}$. Further, $\mathcal{S}_\mathcal{U}$ derives a temporary authentication secret K from $pwd_\mathcal{D}$ and uses K to compute authenticator σ_{del}. Moreover, $\mathcal{S}_\mathcal{U}$ encrypts $(K^{\mathcal{D}}_{\text{Auth}}, T_\mathcal{D}, T_\mathcal{U})$ with the platform key $pk^{\mathcal{D}}_{\mathcal{P}}$ of $\mathcal{S}_\mathcal{D}$ and sends the resulting ciphertext c_{del} to $\mathcal{S}_\mathcal{D}$. Next, $\mathcal{S}_\mathcal{D}$ decrypts and, in case the verification of σ is successful, stores $K^{\mathcal{D}}_{\text{Auth}}$ and sends $(T_\mathcal{D}, T_\mathcal{U})$ to $\mathcal{H}_\mathcal{D}$, which are used later in the authentication protocol.

Authentication of Delegated Users. Authentication of delegated users is similar to authentication of registered users (Figure 5). The only difference is that a delegated user $\mathcal{D}$ sends in addition to its delegated token $T_\mathcal{D}$ also the token $T_\mathcal{U}$ of user $\mathcal{U}$ that created $T_\mathcal{D}$. Further, $\mathcal{R}$ first decrypts $T_\mathcal{U}$ to obtain $K^{\mathcal{U}}_{\text{Del}}$, which is then used to decrypt $K^{\mathcal{D}}_{\text{Auth}}$ from $T_\mathcal{D}$. The rest of the authentication protocol is the same as in Figure 5.

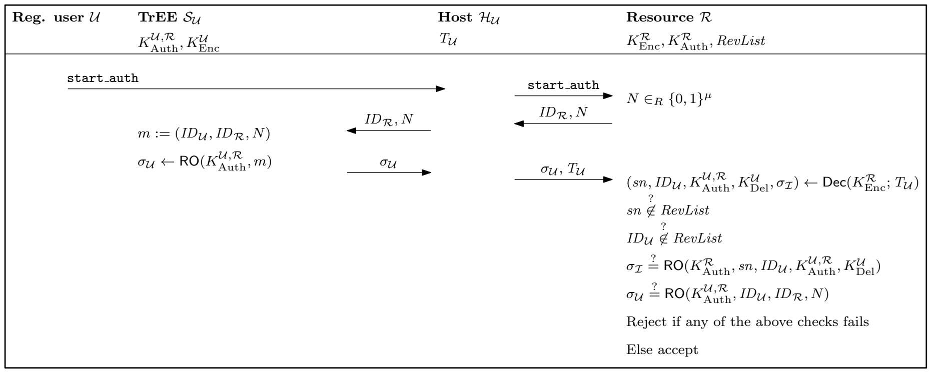

Reg. user $\mathcal{U}$
TrEE $\mathcal{S}_\mathcal{U}$
$K^{\mathcal{U},\mathcal{R}}_{\text{Auth}}, K^{\mathcal{U}}_{\text{Enc}}$
Host $\mathcal{H}_\mathcal{U}$
$T_\mathcal{U}$
Resource $\mathcal{R}$
$K^{\mathcal{R}}_{\text{Enc}}, K^{\mathcal{R}}_{\text{Auth}}, RevList$
start_auth
start_auth
$N \in_R \{0,1\}^\mu$
$ID_\mathcal{R}, N$
$ID_\mathcal{R}, N$
$m := (ID_\mathcal{U}, ID_\mathcal{R}, N)$
$\sigma_\mathcal{U} \leftarrow \mathsf{RO}(K^{\mathcal{U},\mathcal{R}}_{\text{Auth}}, m)$
$\sigma_\mathcal{U}$
$\sigma_\mathcal{U}, T_\mathcal{U}$
$(sn, ID_\mathcal{U}, K^{\mathcal{U},\mathcal{R}}_{\text{Auth}}, K^{\mathcal{U}}_{\text{Del}}, \sigma_\mathcal{I}) \leftarrow \mathsf{Dec}(K^{\mathcal{R}}_{\text{Enc}}; T_\mathcal{U})$
$sn \overset{?}{\notin} RevList$
$ID_\mathcal{U} \overset{?}{\notin} RevList$
$\sigma_\mathcal{I} \overset{?}{=} \mathsf{RO}(K^{\mathcal{R}}_{\text{Auth}}, sn, ID_\mathcal{U}, K^{\mathcal{U},\mathcal{R}}_{\text{Auth}}, K^{\mathcal{U}}_{\text{Del}})$
$\sigma_\mathcal{U} \overset{?}{=} \mathsf{RO}(K^{\mathcal{U},\mathcal{R}}_{\text{Auth}}, ID_\mathcal{U}, ID_\mathcal{R}, N)$
Reject if any of the above checks fails
Else accept

Fig. 5. Authentication protocol for registered users

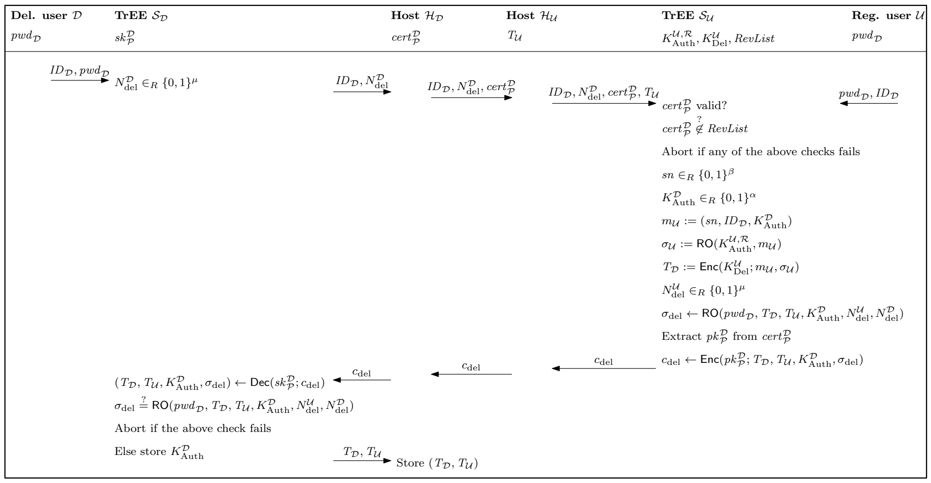

Del. user $\mathcal{D}$
$pwd_\mathcal{D}$
TrEE $\mathcal{S}_\mathcal{D}$
$sk^{\mathcal{D}}_{\mathcal{P}}$
Host $\mathcal{H}_\mathcal{D}$
$cert^{\mathcal{D}}_{\mathcal{P}}$
Host $\mathcal{H}_\mathcal{U}$
$T_\mathcal{U}$
TrEE $\mathcal{S}_\mathcal{U}$
$K^{\mathcal{U},\mathcal{R}}_{\text{Auth}}, K^{\mathcal{U}}_{\text{Del}}, RevList$
Reg. user $\mathcal{U}$
$pwd_\mathcal{D}$
$ID_\mathcal{D}, pwd_\mathcal{D}$
$N^{\mathcal{D}}_{\text{del}} \in_R \{0,1\}^\mu$
$ID_\mathcal{D}, N^{\mathcal{D}}_{\text{del}}$
$ID_\mathcal{D}, N^{\mathcal{D}}_{\text{del}}, cert^{\mathcal{D}}_{\mathcal{P}}$
$ID_\mathcal{D}, N^{\mathcal{D}}_{\text{del}}, cert^{\mathcal{D}}_{\mathcal{P}}, T_\mathcal{U}$
$pwd_\mathcal{D}, ID_\mathcal{D}$
$cert^{\mathcal{D}}_{\mathcal{P}}$ valid?
$cert^{\mathcal{D}}_{\mathcal{P}} \overset{?}{\notin} RevList$
Abort if any of the above checks fails
$sn \in_R \{0,1\}^\beta$
$K^{\mathcal{D}}_{\text{Auth}} \in_R \{0,1\}^\alpha$
$m_\mathcal{U} := (sn, ID_\mathcal{D}, K^{\mathcal{D}}_{\text{Auth}})$
$\sigma_\mathcal{U} := \mathsf{RO}(K^{\mathcal{U},\mathcal{R}}_{\text{Auth}}, m_\mathcal{U})$
$T_\mathcal{D} := \mathsf{Enc}(K^{\mathcal{U}}_{\text{Del}}; m_\mathcal{U}, \sigma_\mathcal{U})$
$N^{\mathcal{U}}_{\text{del}} \in_R \{0,1\}^\mu$
$\sigma_{\text{del}} \leftarrow \mathsf{RO}(pwd_\mathcal{D}, T_\mathcal{D}, T_\mathcal{U}, K^{\mathcal{D}}_{\text{Auth}}, N^{\mathcal{U}}_{\text{del}}, N^{\mathcal{D}}_{\text{del}})$
Extract $pk^{\mathcal{D}}_{\mathcal{P}}$ from $cert^{\mathcal{D}}_{\mathcal{P}}$
$c_{\text{del}} \leftarrow \mathsf{Enc}(pk^{\mathcal{D}}_{\mathcal{P}}; T_\mathcal{D}, T_\mathcal{U}, K^{\mathcal{D}}_{\text{Auth}}, \sigma_{\text{del}})$
c_{del}
c_{del}
c_{del}
$(T_\mathcal{D}, T_\mathcal{U}, K^{\mathcal{D}}_{\text{Auth}}, \sigma_{\text{del}}) \leftarrow \mathsf{Dec}(sk^{\mathcal{D}}_{\mathcal{P}}; c_{\text{del}})$
$\sigma_{\text{del}} \overset{?}{=} \mathsf{RO}(pwd_\mathcal{D}, T_\mathcal{D}, T_\mathcal{U}, K^{\mathcal{D}}_{\text{Auth}}, N^{\mathcal{U}}_{\text{del}}, N^{\mathcal{D}}_{\text{del}})$
Abort if the above check fails
Else store $K^{\mathcal{D}}_{\text{Auth}}$
$T_\mathcal{D}, T_\mathcal{U}$
Store $(T_\mathcal{D}, T_\mathcal{U})$

Fig. 6. Token delegation protocol

Token and User Revocation. To revoke a token $T_\mathcal{U}$ (or all tokens of user $\mathcal{U}$), sn (or $ID_\mathcal{U}$) is added to $RevList$.

4 Security Analysis

The security goal of the authentication scheme in Section 3.3 is *token authentication*, which means that only registered and delegated users, whose smartphone has a valid token T and knows the corresponding authentication secret K_{Auth}, can make an honest resource $\mathcal{R}$ accept. This can be formalized by a security experiment $\mathbf{Exp}^{\text{Auth}}_{\mathcal{A}}(q) = out^{\pi}_{\mathcal{R}}$, where a probabilistic polynomial time (p.p.t.) adversary $\mathcal{A}$ must make an honest resource $\mathcal{R}$ to authenticate $\mathcal{A}$ either as a registered user $\mathcal{U}$ or delegated user $\mathcal{D}$ by returning $out^{\pi}_{\mathcal{R}} = 1$ in some instance π of one of the authentication protocols (Section 3.3). Following the approach by Canetti et al. [12], $\mathcal{A}$ can arbitrarily interact a limited number of times q

with $\mathcal{I}$, $\mathcal{U}$, $\mathcal{D}$ and their mobile platforms $\mathcal{P} = (\mathcal{H}, \mathcal{S})$ and knows all information stored on $\mathcal{H}$. However, since we do not consider relay attacks, $\mathcal{A}$ is not allowed to just forward all messages from $\mathcal{S}$ to $\mathcal{R}$ in instance π. Hence, at least some of the protocol messages that made $\mathcal{R}$ accept must have been (partly) computed by $\mathcal{A}$ without knowing the secrets of $\mathcal{S}$. Note that, as discussed in Section 3.1, by assumption $\mathcal{A}$ does not know any value, including intermediate computation results, stored in $\mathcal{S}$ at any time and can only obtain the messages sent to $\mathcal{S}$ and its responses.

Definition 1. *A token-based authentication scheme achieves token authentication if for every p.p.t. adversary* $\mathcal{A}$ $\Pr\left[\mathbf{Exp}_{\mathcal{A}}^{\mathrm{Auth}}(q) = 1\right]$ *is negligible in* q.

Theorem 1. *The authentication scheme in Section 3.3 achieves token authentication (Definition 1) in the random oracle model under the assumption that the underlying encryption schemes are CPA-secure (Section 3.2).*

We give a proof sketch here, while the full proof can be found in Appendix A.

Proof (Sketch). Assume by contradiction that $\mathcal{A}$ is an adversary with non-negligible success probability. We show that $\mathcal{A}$ can be used to construct an adversary $\mathcal{B}$ that violates the definition of the underlying random oracle RO or the CPA-security of the underlying encryption schemes (Section 3.2). More detailed, $\mathcal{B}$ simulates the protocols in Section 3.3 according to their specification except that $\mathcal{B}$ simulates all ciphertexts and tokens by encrypting random plaintexts. Following the approach by Shoup [37], we show that the CPA-security of the underlying encryption schemes ensures that the simulation by $\mathcal{B}$ has a negligible effect on $\mathcal{A}$'s success probability. $\mathcal{A}$ is allowed to arbitrarily interact with RO and the simulation by $\mathcal{B}$. Eventually, in protocol session π, $\mathcal{A}$ responds to message $(ID_{\mathcal{R}}, N)$ generated by $\mathcal{B}$ with $out_{\pi}^{\mathcal{A}}$, which is used by $\mathcal{B}$ to compute either a collision for RO or to predict the output of RO with non-negligible probability, which violates the definition of RO (Section 3.2). Hence, RO and the CPA-security of the underlying encryption schemes ensure that there is no p.p.t. $\mathcal{A}$ that violates token authentication (Definition 1) with non-negligible probability. $\square$

5 SmartTokens Reference Implementation

In this section, we describe the implementation of the SmartToken design presented in Section 3.3 based on the security architecture described in Section 2. We exemplarily consider the scenario, where a company plays the role of issuer $\mathcal{I}$, while users $\mathcal{U}$ correspond to employees and delegated users $\mathcal{D}$ to temporary visitors or other employees. The resources $\mathcal{R}$ are the company premises, including buildings and rooms.

5.1 Instantiation of the Multi-level Platform Security Architecture

In our current implementation, we instantiated a modified multi-level security architecture that slightly differs from the one described in Section 2. The reason

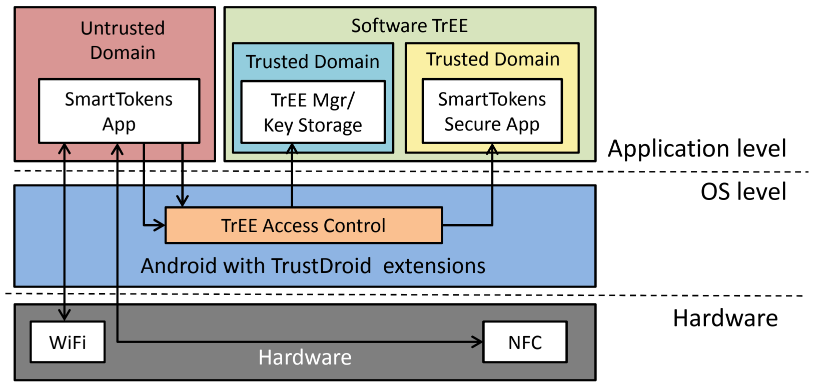


Fig. 7. Implemented Platform Security Architecture

is that we could not identify any Android device featuring both NFC and security hardware that can be used by third party developers. In particular, we could not find Android devices with M-Shield or ARM TrustZone, while Android platforms with SIM cards or universal integrated circuit cards (UICC) do not allow accessing the secure hardware. Moreover, there seems to be no Android device on the market that provides both an NFC interface and a microSD slot, which would have allowed using a removable secure memory card (SMC) as TrEE. However, we envision the availability of such devices in the near future and designed our implementation such that it can be easily ported to these security modules upon availability.

Due to this temporal limitation, our current prototype uses software-based isolation to establish a trusted execution environment (TrEE) on the device. The refined security platform architecture is depicted in Figure 7. It builds on the top of TrustDroid [10], a security framework that enhances the standard Android operating system with mandatory access control at all operating system levels, wich allows to establish isolated compartments (or domains) on the device. Further, TrustDroid allows to define inter-domain communication rules by specifying system-centric security policies.[7] We realized acces control to the TrEE as a security service of TrustDroid (thus, it resides at the level of the operating system), while TrEE is realized as a number of application-level isolated compartments. One TrEE-based compartment contains the TrEE Manager, while other compartments are intended to run secure code associated with host applications running in an untrusted compartment.

[7] TrustDroid applies very simple rules that restrict inter-domain communication. However, the TrustDroid framework itself allows defining more sophisticated security policies, e.g., to prevent application-level privilege escalation attacks [9,8].

Implementation Details. We implemented the SmartToken scheme in Section 3.3 on Nexus S smartphones running Android 2.3.3 patched with TrustDroid security extensions. The prototype implementation of resources uses a commodity NFC reader (ACS ACR 122 U) connected to a Linux PC running Ubuntu Oneiric.

NFC communication mode. We implemented our protocols using Android NFC card reader and writer APIs, which provide direct access to different NFC tag technologies using tag-specific application protocol data unit (APDU) command and response structures. Specifically, we use the ISO Dep Android API that allows direct access to smartcard properties and read/write operations according to the widely used ISO 14443-4 standard for contactless smartcards. The NFC reader emulates NFC Forum type 4 contactless smartcards that communicate according to ISO 14443-4. We used libnfc open source libraries[8] for accessing the NFC reader from the Linux PC. The implementation of the token authentication and user delegation protocol (Figure 5 and 6) uses ISO/IEC 7816-4 and ISO/IEC 7816-8 specific APDUs. ISO/IEC 7816-4 defines a standard interface for identifying applications and accessing files and data on smartcards, while ISO/IEC 7816-8 defines commands for security operations on smartcards. Further, we implemented an application on the Linux PC emulating the resource in the token authentication protocols.

Primitives and parameter sizes. Random oracle RO is implemented as HMAC based on SHA-1, where $\alpha = 160$. For the symmetric encryption scheme ES we used AES, i.e., $\delta = 128$. To achieve CPA-security (Section 3.2), which is required by the MAC-then-encrypt paradigm [5] used in our protocols and our security proof, AES is used in CBC mode with random padding. The public-key encryption scheme is implemented based on RSA with random padding, which means that platform keys are $2,048$ bit RSA keys. Further, we use $\beta = 64$ for token serial numbers sn and $\mu = 128$ for nonces. All identifiers ID are random 64 bit strings. For the one-time passwords pwd used in the user registration (Figure 3) and token delegation protocol (Figure 6), we use $\rho = 128$. Note that long passwords can be encoded in a barcode or data-matrix that can be printed on the user's welcome letter and scanned with the smartphone's camera. For delegated users, the barcode can be shown on the display of the registered user's smartphone and scanned by the camera of the delegated user's phone.

Performance Analysis. We measured the time required to complete an authentication protocol session between the NFC reader and the phone for a registered user and a delegated user. Table 1 shows the time for exchanging different protocol messages and the overall authentication session completion time. The average data transmission rate between NFC reader and phone is around 10 kbps. Our measurements show that it requires about 540 ms to complete an authentication session for a registered user and about 565 ms for a delegated user.

[8] www.libnfc.org

Table 1. Transmission times for authentication protocol messages (units are in milliseconds with 95% confidence interval)

User Type	**Connection Estb.**	**Start Msg.**	**Reading** $(ID_{\mathcal{R}}, N)$	**Sending** $(\sigma_{\mathcal{D}}, T_{\mathcal{D}}, T_{\mathcal{U}})$	**Session Time**
Registered	245.17(± .18)	42.19(±.52)	59.6 (± .51)	98.4 (± .53)	441.8(± .54)
Delegated	245.17(± .18)	42.19(±.52)	59.6 (± .51)	121.6 (± .54)	473.55 (± .54)

6 Related Work

There are several NFC-based applications for smartphones, including key storage and management, payment and ticketing systems, and remote attestation.

Key storage and management. Mantoro et al. [29] propose a scheme to protect the cryptographic keys of a PC by securely storing them in the SIM card of an NFC-enabled phone. However, the scheme protects only against offline attacks aiming to recover keys from the PC memory and is vulnerable to runtime attacks since keys are uploaded to the PC when used and thus can be accessed by malware. Noll et al. [33,26] propose a key management architecture that uses a SIM card to securely manage the authentication secrets of a smartphone. They describe several use cases, including an NFC-based access control system that allows distributing electronic keys via SMS. However, the security of their scheme is unclear since the use case is only sketched and neither protocols nor a security analysis is provided.

NFC-based payment systems. Chen et al. [13] propose an NFC-based mobile payment system leveraging SIM-based authentication capabilities of GSM networks. However, their scheme requires all involved parties to be subscribed to the same mobile operator, which is not always guaranteed in practice and not required in our scheme. Another NFC-based mobile payment system by Kadambi et al. [25] is based on payment authorization tokens that are used to authorize transactions. Their scheme protects privacy-sensitive user data, such as credit card numbers, even against merchants. Although their solution uses secure hardware, access to the secure environment is controlled by a commodity operating system that may be vulnerable to various attacks [31,32]. In contrast, in our scheme access control to the TrEE is enforced by trusted software components. Gauthier et al. [19] propose an offline payment system based on digital vouchers that can be transferred from one to another device over NFC. However, their scheme heavily relies on public-key operations resulting in low performance, while our scheme uses only efficient symmetric techniques and tackles the bandwidth limitations of the NFC interface (for the protocol running over NFC). The Merx system [38] provides a solution for delegated electronic payments. Its system model involves four parties: (1) a customer, (2) a concierge, (3) a merchant and (4) a bank, which can be mapped to the entities of our model as follows: (1) a user, (2) a delegated user, (3) a resource and (4) an issuer, respectively. The system requires online interactions between merchant and bank on each purchase, which is common for payment systems. However, when mapped into our use case, this scheme would require an online connection between the issuer and the resource upon each access of the user to the resource, which is highly undesirable in our use case and not required by our scheme.

NFC-based ticketing systems. Tamrakar et al. [39] present an NFC-based authentication scheme for electronic transport tickets. However, their scheme is vulnerable to replay attacks and assumes the mobile device to be equipped with a trusted time source, which is hard to achieve in practice and not required in our scheme. Ghìron et al. [20] present a prototype implementation of an NFC-enabled ticketing system. However, their work focuses on usability rather than security aspects.

NFC-enabled remote attestation. Toegl et al. [24,41] propose verifying the integrity of public terminals, such as ticket vending machines, using NFC-enabled smartphones. Their scheme requires terminals to be equipped with NFC-enabled TPMs, which are not conform to the latest TPM specification [22] and not available on the market.

7 Conclusion and Future Work

We present the design of a token-based access control system for NFC-enabled smartphones that can be used in many applications. The scheme allows users to delegate (part of) their access rights to other smartphone users without involvement of a central authority (a token issuer). Our scheme considers the bandwidth constraints of NFC by using only symmetric cryptographic primitives for the protocols running over NFC. We provide a formal security analysis of our scheme and instantiate it in the application scenario, where access control tokens are used as electronic door keys. We propose an implementation of our system for Android-powered Nexus S smartphones. Our performance analysis shows that authentication can be performed within 474 ms. Furthermore, we present a multi-level security architecture to protect the underlying authentication secrets of our protocols. The architecture combines a hardware-assisted trusted execution environment (TrEE) with software-based isolation and overcomes the drawbacks of existing solutions.

Future work includes extending the implementation of our multi-level security architecture for Android-based smartphones with security hardware, when these devices are available on the market. Moreover, we are implementing the token-based access control system and the multi-level security architecture on Nokia C7 phones, which feature an NFC interface and ARM TrustZone security hardware.

Acknowledgements. We thank our anonymous reviewers for their helpful comments and Raphael Friedrich, Stephan Heuser and Daniel Steinmetzer for supporting the implementation. This work has been supported in part by the European Commission through the FP7 programme under contract 238811 UNIQUE.

References

1. VingCard Elsafe's NFC locking solution wins prestigious gaming industry technology award, http://www.hotel-online.com/News/PR2011_3rd/Aug11_VingCardHOT.html
2. Alves, T., Felton, D.: TrustZone: Integrated hardware and software security. Information Quaterly 3(4) (2004)
3. Azema, J., Fayad, G.: M-Shield mobile security technology: making wireless secure. Texas Instruments White Paper (2008), http://focus.ti.com/pdfs/wtbu/ti_mshield_whitepaper.pdf
4. Bellare, M., Desai, A., Pointcheval, D., Rogaway, P.: Relations among notions of security for public-key encryption schemes. In: CRYPTO. vol. 1462, pp. 26–45 (1998)
5. Bellare, M., Namprempre, C.: Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In: Okamoto, T. (ed.) Advances in Cryptology (ASIACRYPT). LNCS, vol. 1976, pp. 531–545. Springer Berlin/Heidelberg (2000)
6. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: ACM Conference on Computer and Communications Security (ACM CCS). pp. 62–73. ACM, New York, NY, USA (1993)
7. Brown, C.: NFC room keys find favour with hotel guests, http://www.nfcworld.com/2011/06/08/37869/nfc-room-keys-find-favour-with-hotel-guests/
8. Bugiel, S., Davi, L., Dmitrienko, A., Fischer, T., Sadeghi, A.R.: Xmandroid: A new android evolution to mitigate privilege escalation attacks. Technical Report TR-2011-04, Technische Universität Darmstadt (2011)
9. Bugiel, S., Davi, L., Dmitrienko, A., Fischer, T., Sadeghi, A.R., Shastry, B.: Towards taming privilege-escalation attacks on Android. In: 19th Annual Network & Distributed System Security Symposium (NDSS) (2012)
10. Bugiel, S., Davi, L., Dmitrienko, A., Heuser, S., Sadeghi, A.R., Shastry, B.: Practical and lightweight domain isolation on Android. In: ACM CCS Workshop on Security and Privacy in Mobile Devices (SPSM). ACM Press (2011)
11. Bugiel, S., Dmitrienko, A., Kostiainen, K., Sadeghi, A.R., Winandy, M.: TruWalletM: Secure web authentication on mobile platforms. In: The Second International Conference on Trusted Systems (INTRUST) (2010)
12. Canetti, R., Krawczyk, H.: Analysis of Key-Exchange protocols and their use for building secure channels. In: Advances in Cryptology (EUROCRYPT). LNCS, vol. 2045, pp. 453–474. Springer Berlin/Heidelberg, Berlin, Heidelberg (2001)
13. Chen, W., Hancke, G.P., Mayes, K.E., Lien, Y., Chiu, J.H.: NFC mobile transactions and authentication based on GSM network. In: International Workshop on Near Field Communication (NFC). pp. 83–89. IEEE Computer Society, Washington, DC, USA (2010)
14. Clark, S.: NXP launches NFC car key, http://www.nfcworld.com/2011/06/22/38196/nxp-launches-nfc-car-key/
15. Clark, S.: VingCard launches NFC room key system for hotels, http://www.nfcworld.com/2011/06/28/38366/vingcard-launches-nfc-room-key-system-for-hotels/
16. Costan, V., Sarmenta, L., van Dijk, M., Devadas, S.: The trusted execution module: Commodity general-purpose trusted computing. In: Smart Card Research and Advanced Application Conference (2008)

17. Davi, L., Dmitrienko, A., Kowalski, C., Winandy, M.: Trusted virtual domains on OKL4: Secure information sharing on smartphones. In: ACM Workshop on Scalable Trusted Computing (ACM STC). ACM Press (2011)
18. Gartner Inc.: http://www.gartner.com/it/page.jsp?id=1689814 (2011)
19. Gauthier, V.D., Wouters, K.M., Karahan, H., Preneel, B.: Offline NFC payments with electronic vouchers. In: ACM Workshop on Networking, Systems, and Applications for Mobile Handhelds (MobiHeld). pp. 25–30. ACM, New York, NY, USA (2009)
20. Ghìron, S.L., Sposato, S., Medaglia, C.M., Moroni, A.: NFC ticketing: A prototype and usability test of an NFC-based virtual ticketing application. In: International Workshop on Near Field Communication (NFC). pp. 45–50. IEEE Computer Society, Washington, DC, USA (2009)
21. Goldwasser, S., Micali, S.: Probabilistic encryption. Journal of Computer and System Sciences 28, 270–299 (1984)
22. Trusted Computing Group: TPM Main Specification, Version 1.2 rev. 103 (2007), https://www.trustedcomputinggroup.org
23. Heiser, G., Leslie, B.: The OKL4 microvisor: Convergence point of microkernels and hypervisors. In: ACM Asia-pacific Workshop on Systems (APSys). pp. 19–24. ACM, New York, NY, USA (2010)
24. Hutter, M., Toegl, R.: A trusted platform module for near field communication. In: International Conference on Systems and Networks Communications (ICSNC). pp. 136–141. IEEE Computer Society, Washington, DC, USA (2010)
25. Kadambi, K.S., Li, J., Karp, A.H.: Near-field communication-based secure mobile payment service. In: International Conference on Electronic Commerce (ICEC). pp. 142–151. ACM, New York, NY, USA (2009)
26. Kalman, G., Noll, J., UniK, K.: SIM as secure key storage in communication networks. In: International Conference on Wireless and Mobile Communications (ICWMC) (2007)
27. Kostiainen, K., Asokan, N., Afanasyeva, A.: Towards User-Friendly Credential Transfer on Open Credential Platforms Applied Cryptography and Network Security. Lecture Notes in Computer Science, vol. 6715, chap. 23, pp. 395–412. Springer Berlin / Heidelberg, Berlin, Heidelberg (2011), http://dx.doi.org/10.1007/978-3-642-21554-4_23
28. Kostiainen, K., Ekberg, J.E., Asokan, N., Rantala, A.: On-board credentials with open provisioning. In: ACM Symposium on Information, Computer, and Communications Security (ASIACCS). pp. 104–115. ACM (2009)
29. Mantoro, T., Milisic, A.: Smart card authentication for Internet applications using NFC enabled phone. In: International Conference on Information and Communication Technology for the Muslim World (ICT4M) (2010)
30. Massachusetts Institute of Technology: Kerberos: The network authentication protocol. http://web.mit.edu/kerberos/
31. McAfee Labs: McAfee threats report: Second quarter 2011. http://www.mcafee.com/us/resources/reports/rp-quarterly-threat-q2-2011.pdf (2011)
32. McAfee Labs: McAfee threats report: Third quarter 2011. http://www.mcafee.com/us/resources/reports/rp-quarterly-threat-q3-2011.pdf (2011)
33. Noll, J., Lopez Calvet, J.C., Myksvoll, K.: Admittance services through mobile phone short messages. In: International Multi-Conference on Computing in the Global Information Technology. pp. 77–82. IEEE Computer Society, Washington, DC, USA (2006)

34. Reveilhac, M., Pasquet, M.: Promising secure element alternatives for NFC technology. In: International Workshop on Near Field Communication (NFC). pp. 75–80. IEEE Computer Society, Washington, DC, USA (2009)
35. Robertson, T.: Eight industries that will benefit from NFC technology, https://www.x.com/devzone/articles/eight-industries-will-benefit-nfc-technology
36. Rushby, J.M.: Design and verification of secure systems. In: ACM Symposium on Operating Systems Principles (SOPS) (1981)
37. Shoup, V.: Sequences of games: A tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332 (2004), http://eprint.iacr.org/2004/332
38. Soghoian, C., Aad, I.: Merx: Secure and privacy preserving delegated payments. In: Chen, L., Mitchell, C., Martin, A. (eds.) Trusted Computing, Lecture Notes in Computer Science, vol. 5471, chap. 14, pp. 217–239. Springer Berlin / Heidelberg, Berlin, Heidelberg (2009)
39. Tamrakar, S., Ekberg, J.E., Asokan, N.: Identity verification schemes for public transport ticketing with NFC phones. In: ACM workshop on Scalable Trusted Computing (STC). pp. 37–48. ACM, New York, NY, USA (2011)
40. Telecom Innovation Laboratories: Mobile Wallet turns cell phones into digital car keys (2011), http://www.laboratories.telekom.com/public/English/Newsroom/news/Pages/digitaler_Autoschluessel_Mobile_Wallet.aspx
41. Toegl, R., Hutter, M.: An approach to introducing locality in remote attestation using near field communications. J. Supercomput. 55(2), 207–227 (2011)
42. Zhang, X., Acıiçmez, O., Seifert, J.P.: A trusted mobile phone reference architecture via secure kernel. In: ACM workshop on Scalable Trusted Computing (ACM STC). pp. 7–14. ACM, New York, NY, USA (2007)

A Proof of Theorem 1

In this section, we prove Theorem 1, which states that the authentication scheme in Section 3.3 achieves token authentication (Definition 1) in the random oracle model under the assumption that the underlying encryption schemes are CPA-secure (Section 3.2).

Proof. Assume by contradiction that $\mathcal{A}$ is an adversary s.t. $\Pr\left[\mathbf{Exp}_{\mathcal{A}}^{\mathrm{Auth}}(q) = 1\right]$ is non-negligible in q. We show that $\mathcal{A}$ can be used to construct an adversary $\mathcal{B}$ that violates the definition of the underlying random oracle RO or the CPA-security of the underlying encryption schemes (Section 3.2).

More detailed, $\mathcal{B}$ simulates all protocols in Section 3.3 according to their specification except that $\mathcal{B}$ sets $c_{\mathrm{reg}} := \mathsf{Enc}(pk_{\mathcal{P}}^{\mathcal{U}}; R_{\mathrm{reg}})$, $T_{\mathcal{U}} := \mathsf{Enc}(K_{\mathrm{Enc}}^{\mathcal{R}}; R_{\mathrm{iss},1})$, $c_{\mathrm{iss}} := \mathsf{Enc}(K_{\mathrm{Enc}}^{\mathcal{U}}; R_{\mathrm{iss},2})$, $T_{\mathcal{D}} := \mathsf{Enc}(K_{\mathrm{Del}}^{\mathcal{U}}; R_{\mathrm{del},1})$ and $c_{\mathrm{del}} := \mathsf{Enc}(pk_{\mathcal{P}}^{\mathcal{D}}; R_{\mathrm{del},2})$, where R_{reg}, $R_{\mathrm{iss},1}$, $R_{\mathrm{iss},2}$, $R_{\mathrm{del},1}$ and $R_{\mathrm{del},2}$ are random plaintexts with the same length as their original counterparts specified in Section 3.3. Further, $\mathcal{B}$ maintains a list $\Gamma_{\mathcal{A}}$ of all queries by $\mathcal{A}$ to random oracle RO and responses of RO. $\mathcal{A}$ is allowed to arbitrarily interact with RO and the simulation by $\mathcal{B}$, where the total number of queries of $\mathcal{A}$ is limited by q. Eventually, in some protocol session π, $\mathcal{A}$ responds to message $(ID_{\mathcal{R}}, N)$ generated by $\mathcal{B}$ either with $out_{\pi}^{\mathcal{A},\mathcal{U}} = (\sigma_{\mathcal{A}}, T_{\mathcal{A},\mathcal{U}})$ or $out_{\pi}^{\mathcal{A},\mathcal{D}} = (\sigma_{\mathcal{A}}, T_{\mathcal{A},\mathcal{D}}, T_{\mathcal{A},\mathcal{U}})$. Note that, in the following we consider only case $out_{\pi}^{\mathcal{A},\mathcal{D}}$ since the proof of case $out_{\pi}^{\mathcal{A},\mathcal{U}}$ is similar. Depending on the output of $\mathcal{A}$, $\mathcal{B}$ distinguishes three cases:

Case 1: $\mathcal{A}$ reuses $out_{\pi}^{\mathcal{A},\mathcal{D}} = (\sigma_{\mathcal{A}}, T_{\mathcal{A},\mathcal{D}}, T_{\mathcal{A},\mathcal{U}})$ from a previous authentication protocol instance π' as response to $(ID_{\mathcal{R}}, N')$. In this case, $\mathcal{B}$ decrypts $(sn, ID_{\mathcal{U}}, K_{\mathrm{Auth}}^{\mathcal{U},\mathcal{R}}, K_{\mathrm{Del}}^{\mathcal{U}}, \sigma_{\mathcal{I}}) \leftarrow \mathsf{Dec}(K_{\mathrm{Enc}}^{\mathcal{R}}; T_{\mathcal{A},\mathcal{U}})$ and $(sn, ID_{\mathcal{D}}, K_{\mathrm{Auth}}^{\mathcal{D}}, \sigma_{\mathcal{U}}) \leftarrow \mathsf{Dec}(K_{\mathrm{Del}}^{\mathcal{U}}; T_{\mathcal{A},\mathcal{D}})$, and returns $\left[\left((K_{\mathrm{Auth}}^{\mathcal{D}}, ID_{\mathcal{D}}, ID_{\mathcal{R}}, N), \sigma_{\mathcal{A}}\right), \left((K_{\mathrm{Auth}}^{\mathcal{D}}, ID_{\mathcal{D}}, ID_{\mathcal{R}}, N'), \sigma_{\mathcal{A}}\right)\right]$ as a collision of RO.

Case 2: $\mathcal{A}$ creates (forges) a new token $T_{\mathcal{A},\mathcal{D}}$ that has never been used in any previous protocol instance. In this case $\mathcal{B}$ decrypts $(sn, ID_{\mathcal{D}}, K_{\mathrm{Auth}}^{\mathcal{D}}, \sigma_{\mathcal{A},\mathcal{U}}) \leftarrow \mathsf{Dec}(K_{\mathrm{Del}}^{\mathcal{U}}; T_{\mathcal{A},\mathcal{D}})$ and, when $\sigma_{\mathcal{A},\mathcal{U}}$ is *not* in $\Gamma_{\mathcal{A}}$, $\mathcal{B}$ returns $\sigma_{\mathcal{A},\mathcal{U}}$ as a prediction of $\mathsf{RO}(K_{\mathrm{Auth}}^{\mathcal{U},\mathcal{R}}, sn, ID_{\mathcal{D}}, K_{\mathrm{Auth}}^{\mathcal{D}})$.

Case 3: $\mathcal{A}$ creates (forges) a new token $T_{\mathcal{A},\mathcal{U}}$ that has never been used in any previous protocol instance. In this case, $\mathcal{B}$ decrypts $(sn, ID_{\mathcal{U}}, K_{\mathrm{Auth}}^{\mathcal{U},\mathcal{R}}, K_{\mathrm{Del}}^{\mathcal{U}}, \sigma_{\mathcal{A},\mathcal{I}}) \leftarrow \mathsf{Dec}(K_{\mathrm{Del}}^{\mathcal{U}}; T_{\mathcal{A},\mathcal{U}})$ and, when $\sigma_{\mathcal{A},\mathcal{I}}$ is *not* in $\Gamma_{\mathcal{A}}$, $\mathcal{B}$ returns $\sigma_{\mathcal{A},\mathcal{I}}$ as a prediction of $\mathsf{RO}(K_{\mathrm{Auth}}^{\mathcal{R}}, sn, ID_{\mathcal{U}}, K_{\mathrm{Auth}}^{\mathcal{U},\mathcal{R}}, K_{\mathrm{Del}}^{\mathcal{U}})$.

Now we show that in any case $\mathcal{B}$ violates the definition of RO (Section 3.2) with non-negligible probability. We start by showing that the simulation by $\mathcal{B}$ is indistinguishable from the protocols in Section 3.3 and has a negligible effect on the success probability $p_{\mathcal{A}}^{(0)}(q) := \Pr\left[\mathbf{Exp}_{\mathcal{A}}^{\mathrm{Auth}}(q)\right]$ of $\mathcal{A}$. Following the approach by Shoup [37], we define six games, $\mathfrak{G}_0$ to $\mathfrak{G}_5$, where $\mathfrak{G}_0$ corresponds to the protocol specification and $\mathfrak{G}_5$ to the simulation by $\mathcal{B}$:

– In $\mathfrak{G}_0$ $\mathcal{A}$ interacts with the authentication scheme specified in Section 3.3

- $\mathfrak{G}_1$ is as $\mathfrak{G}_0$ except that $c_{\text{reg}} = \mathsf{Enc}(pk_{\mathcal{P}}^{\mathcal{U}}; R_{\text{reg}})$ is used
- $\mathfrak{G}_2$ is as $\mathfrak{G}_1$ except that $T_{\mathcal{U}} = \mathsf{Enc}(K_{\text{Enc}}^{\mathcal{R}}; R_{\text{iss},1})$ is used
- $\mathfrak{G}_3$ is as $\mathfrak{G}_2$ except that $c_{\text{iss}} = \mathsf{Enc}(K_{\text{Enc}}^{\mathcal{U}}; R_{\text{iss},2})$ is used
- $\mathfrak{G}_4$ is as $\mathfrak{G}_3$ except that $T_{\mathcal{D}} = \mathsf{Enc}(K_{\text{Del}}^{\mathcal{U}}; R_{\text{del},1})$ is used
- $\mathfrak{G}_5$ is as $\mathfrak{G}_4$ except that $c_{\text{del}} = \mathsf{Enc}(pk_{\mathcal{P}}^{\mathcal{D}}; R_{\text{del},2})$ is used

With $p_{\mathcal{A}}^{(j)}(q)$ we denote the probability that $\mathcal{A}$ succeeds in game $\mathfrak{G}_j$. Next, we show that $|p_{\mathcal{A}}^{(0)}(q) - p_{\mathcal{A}}^{(5)}(q)|$ is negligible in q by showing that $|p_{\mathcal{A}}^{(i)}(q) - p_{\mathcal{A}}^{(i+1)}(q)|$ is negligible for $0 \leq i \leq 4$. We start with $i = 1$: Assume by contradiction and w.l.o.g. that $p_{\mathcal{A}}^{(0)}(q)$ is non-negligible and $p_{\mathcal{A}}^{(1)}(q)$ is negligible. This allows constructing an adversary $\mathcal{B}_1$ that wins against CPA-challenger $\mathcal{C}_{sk_{\mathcal{P}}^{\mathcal{U}}}^{\text{CPA}}$ (Section 3.2): $\mathcal{B}_1$ simulates the protocols in Section 3.3 according to their specifications but sets c_{reg} to the output of $\mathcal{C}_{pk_{\mathcal{P}}^{\mathcal{U}}}^{\text{CPA}}$ on input of $m_0 = (K_{\text{Auth}}^{\mathcal{U},\mathcal{I}}, K_{\text{Enc}}^{\mathcal{U}}, N_{\mathcal{I}}, \sigma_{\text{reg}})$ created according to the protocol specification, and a random plaintext $m_1 = R_{\text{reg}}$. In case $\mathcal{A}$ wins $\mathbf{Exp}_{\mathcal{A}}^{\text{Auth}}$, $\mathcal{B}_1$ returns $b' = 0$ as a guess for the random choice b of $\mathcal{C}_{sk_{\mathcal{P}}^{\mathcal{U}}}^{\text{CPA}}$. Note that, depending on b, $\mathcal{A}$ either obtains the encryption of m_0, which corresponds to $\mathfrak{G}_0$, or the encryption of m_1, which corresponds to $\mathfrak{G}_1$. Since by assumption $p_{\mathcal{A}}^{(0)}(q)$ is non-negligible, $\mathcal{B}_1$ can distinguish between the encryption of m_0 and m_1 with non-negligible probability, which contradicts CPA-security (Section 3.2) of the underlying encryption scheme. In turn, this means that CPA-security ensures that $|p_{\mathcal{A}}^{(0)}(q) - p_{\mathcal{A}}^{(1)}(q)|$ is negligible. The proofs of negligibility of $|p_{\mathcal{A}}^{(i)}(q) - p_{\mathcal{A}}^{(i+1)}(q)|$ for $1 \leq i \leq 4$ are similar to case $i = 0$ and omitted due to space restrictions. Consequently, it follows that in case $p_{\mathcal{A}}^{(0)}(q)$ is non-negligible, then $p_{\mathcal{A}}^{(5)}(q)$ must be non-negligible.

Now we show that if $p_{\mathcal{A}}^{(5)}(q)$ is non-negligible, then $\mathcal{B}$ can violate the properties of RO with non-negligible probability in each of the three cases considered above. Note that in case 1 by assumption $\mathcal{A}$ computes $\sigma_{\mathcal{A}} = \mathsf{RO}(K_{\text{Auth}}^{\mathcal{D}}, ID_{\mathcal{D}}, ID_{\mathcal{R}}, N)$ with non-negligible probability $p_{\mathcal{A}}^{(5)}(q)$. Moreover, $\Pr[N = N'] = 2^{-\mu}$ since $N, N' \in_R \{0,1\}^{\mu}$. Thus, $\mathcal{B}$ returns a collision of RO with non-negligible probability $(1 - 2^{-\mu}) \cdot p_{\mathcal{A}}^{(5)}(q)$, which contradicts the definition of RO (Section 3.2). Further, in case 2 by assumption $\mathcal{A}$ computes a new $T_{\mathcal{A},\mathcal{D}}$ such that $\sigma_{\mathcal{A},\mathcal{U}} = \mathsf{RO}(K_{\text{Auth}}^{\mathcal{U},\mathcal{R}}, sn, ID_{\mathcal{U}}, K_{\text{Auth}}^{\mathcal{D}})$. Note that, when $\sigma_{\mathcal{A},\mathcal{U}}$ is in $\Gamma_{\mathcal{A}}$, this implies that $\mathcal{A}$ knows $K_{\text{Auth}}^{\mathcal{U},\mathcal{R}}$. However, $K_{\text{Auth}}^{\mathcal{U},\mathcal{R}}$ has never been used by $\mathcal{B}$ except in queries to RO, which means that all messages observed by $\mathcal{A}$ are independent of $K_{\text{Auth}}^{\mathcal{U},\mathcal{R}}$. Moreover, $K_{\text{Auth}}^{\mathcal{U},\mathcal{R}} \in_R \{0,1\}^{\alpha}$. Hence, $\sigma_{\mathcal{A},\mathcal{U}}$ is in $\Gamma_{\mathcal{A}}$ with negligible probability $2^{-\alpha}$. In turn this means that $\sigma_{\mathcal{A},\mathcal{U}}$ is a prediction of $\mathsf{RO}(K_{\text{Auth}}^{\mathcal{R}}, sn, ID_{\mathcal{U}}, K_{\text{Auth}}^{\mathcal{U},\mathcal{R}}, K_{\text{Del}}^{\mathcal{U}})$ with non-negligible probability $(1 - 2^{-\alpha}) \cdot p_{\mathcal{A}}^{(5)}(q)$, which contradicts the definition of RO (Section 3.2). The proof of case 3 is similar to case 2 and omitted due to space restrictions. In case 3 $\sigma_{\mathcal{A},\mathcal{I}}$ is a prediction of $\mathsf{RO}(K_{\text{Auth}}^{\mathcal{R}}, sn, ID_{\mathcal{U}}, K_{\text{Auth}}^{\mathcal{U},\mathcal{R}}, K_{\text{Del}}^{\mathcal{U}})$ with non-negligible probability $(1 - 2^{-\alpha}) \cdot p_{\mathcal{A}}^{(5)}(q)$, which again contradicts the definition of RO (Section 3.2).

Summing up, random oracle RO and the CPA-security of the underlying encryption schemes ensure that there is no p.p.t. $\mathcal{A}$ such that $\Pr\left[\mathbf{Exp}_{\mathcal{A}}^{\mathrm{Auth}}(q)\right]$ is negligible in q, which concludes the proof. $\square$