Received 26 August 2015
Revised 3 October 2015
Accepted 1 February 2016

MOONACS: a mobile on-/offline NFC-based physical access control system

Dominik Gruntz
Institute for Mobile and Distributed Systems, UAS Northwestern Switzerland (FHNW), Windisch, Switzerland
Christof Arnosti
Bixi Systems AG, Liestal, Switzerland, and
Marco Hauri
Bixi Systems AG, Mels, Switzerland

Abstract
Purpose – The purpose of this paper is to present a smartphone-based physical access control system in which the access points are not directly connected to a central authorization server, but rather use the connectivity of the mobile phone to authorize a user access request online by a central access server. The access points ask the mobile phone whether a particular user has access or not. The mobile phone then relays such a request to the access server or presents an offline ticket. One of the basic requirements of our solution is the independence from third parties like mobile network operators, trusted service managers and handset manufacturers.
Design/methodology/approach – The authentication of the smartphone is based on public key cryptography. This requires that the private key is stored in a secure element or in a trusted execution environment to prevent identity theft. However, due to the intended independence from third parties, subscriber identity module (SIM)-based secure elements and embedded secure elements (i.e. separate hardware chips on the handset) were not an option and only one of the remaining secure element architectures could be used: host card emulation (HCE) or a microSD-based secure element.
Findings – This paper describes the implementation of such a physical access control system and discusses its security properties. In particular, it is shown that the HCE approach cannot solve the relay attack under conservative security assumptions and an implementation based on a microSD secure element is presented and discussed. Moreover, the paper also describes an offline solution which can be used if the smartphone is not connected to the access server. In this case, an access token is sent to the access point in response to an access request. These tokens are renewed regularly and automatically whenever the smartphone is connected.

This work has been carried out in a cooperation between Bixi Systems AG and the University of Applied Sciences Northwestern Switzerland. The authors would like to thank the Swiss Commission for Technology and Innovation (CTI) which cofinanced this project. Many thanks also go to Carlo Nicola at FHNW for his help in the protocol design and to Markus Freund at Bixi for the vital task of implementing the access point part of the protocol.
This article is an extended version of Arnosti C., Gruntz D. and Hauri M. Secure Physical Access with NFC-enabled Smartphones. In *Proceedings of the 13th International Conference on Advances in Mobile Computing & Multimedia (MoMM2015)*. ACM, Brussels, Belgium, December 2015, 140-148.

International Journal of Pervasive Computing and Communications
Vol. 12 No. 1, 2016
pp. 2-22
© Emerald Group Publishing Limited
1742-7371
DOI 10.1108/IJPCC-01-2016-0012

Originality/value – In this paper, a physical access control system is presented which operates as fast as existing card-based solutions. By using a microSD-based secure element (SE), the authors were able to prevent the software relay attack. This solution is not restricted to microSD-based SEs, it could also be implemented with SIM-based or embedded secure elements (with the consequence that the solution depends on third parties).

Keywords NFC, Secure element, Host card emulation, Physical access control system, Relay attack

Paper type Research paper

1. Introduction

The smartphone has become the central gadget in our life and makes our wallet more and more redundant. We use the phone for mobile payment, for mobile ticketing and soon it will replace all the smart cards we carry in our wallet.

This paper presents a physical access control system (PACS) in which the access card is replaced by a smartphone. The access points are not connected; they check whether access is granted by sending an access request to the mobile phone. The mobile phone either forwards such a request to the access server or, if it is not connected to the Internet, presents an access token stored on the mobile phone to the access point. Access is thus possible even if the mobile phone is not connected to the Internet.

The communication between the mobile phone and the access point is based on Near field communication (NFC). NFC is a short-range wireless technology that enables communication between a smartphone and an access point over a distance of approximately 10 cm or less. NFC operates at 13.56 MHz and can achieve (theoretical) transfer rates up to 424 kbit/s. A growing number of smartphones are equipped with NFC (NFC World, 2016).

We define a PACS as a system that controls access to physical resources like buildings, rooms or protected areas, using user-specific access control rules. A PACS supports two main activities: authentication and authorization. When a person requests access to a physical resource, it claims its identity and the authentication process verifies this claim. Authorization then defines whether access is granted for this identity or not. In a PACS, the authentication process checks whether the person:

- knows a secret (e.g. a password or a PIN);
- has something (e.g. a key or a token); or
- has a unique property (e.g. biometric properties).

PACS based on NFC and mobiles are not new. Such systems typically store the access rights on the subscriber identity module (SIM) card or on an embedded secure element and they depend on third-party suppliers, for example mobile network operators (MNOs), trusted service managers (TSMs), smartphone manufacturers like Google, Apple or Samsung or identity service providers like Legic IDConnect.

In this paper, we present a PACS which is independent of third-party providers. The smart card is replaced by the smartphone which is connected to the access server. However, as these phones are programmable and network connected, they provide a large surface for attacks (Janssen and Zandstra, 2014). In this paper, we describe and analyze the security of different authentication solutions developed in the context of a concrete PACS. In particular, we present a novel authentication process which prevents software proxy attacks.

The rest of the paper is organized as follows: in Section 2 we describe the general structure of our solution and the designed protocols in Section 3. The security properties are analyzed in Section 4 and a solution to the authorization problem is presented in Section 5. Finally, we describe further (physical) attacks and compare the different PACS approaches in Sections 6 and 7. After an overview of related approaches in Section 8, we conclude with the results in Section 9.

2. PACS models

A classical PACS consists of three components: a server, identity cards and access points (doors with electronic components). There exist two common interaction models for these components.

In the online access point model, the access point is connected with the server over a network connection. In this model, the card acts as an authentication-only component. The access point authenticates the card and accesses the server to get authorization information for the authenticated card. The server then decides whether access is permitted or not.

In the offline access point model, the access point has no connection to the server. Authorization data for a set of access points are stored on the card. Such authorization data contain access rights which are only valid for a limited time (which is defined individually by the server for each access point and each card) and have to be renewed periodically by the end-user. The access point is able to authenticate the card and to get the authorization data directly from it.

A PACS in which the identity cards are emulated by smartphones can follow both models. But as a smartphone is a connected device, a third model is possible, namely, a model where the offline access point is connected to the server using the smartphone as a proxy. This model combines the advantages of the other two models, i.e. the access points do not need a network connection (which means reduced infrastructure costs) but nevertheless support the verification of access rights at access time.

In our system, the smartphone is responsible for authentication. Authorization data are requested from the server by the smartphone after the initial contact between the access point and the smartphone (see Figure 1).

We also implemented the possibility to store access rights on the smartphone, similar to the offline access point model in a traditional PACS model. This offline ticket mode is used in cases where network access is not available to the mobile at the access point. The connectivity of the mobiles is used to update the authorization data on the mobile.

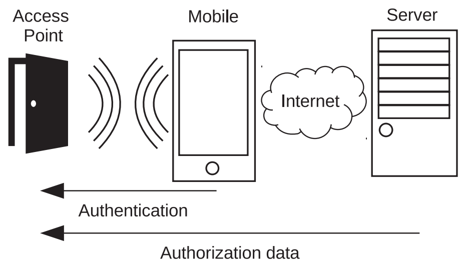


Figure 1.
Smartphone solution

3. Protocol design

As shown in Figure 1, the protocol scheme of our smartphone-based PACS consists of two protocols:

1. the authorization protocol to transmit authorization data between the server and the access point, using the smartphone as a relay; and
2. the authentication protocol to authenticate the smartphone to the access point.

In the authorization protocol, trust between the server and the access point is created with a symmetric key known to both end points prior to the transaction. The authorization answer sent by the server contains the public part of an RSA key pair owned by the mobile. In the authentication protocol, the mobile uses its RSA key to sign the authentication request. The access point can use the public key which it received in the trusted authorization answer to authenticate the mobile.

Both the authentication and the authorization protocols are initiated by the access point and the requests for both protocols are simultaneously sent from the access point to the smartphone using NFC. The answers are also simultaneously sent in the same NFC response from the smartphone back to the access point.

3.1 Authentication protocol

To authenticate the smartphone (M), the access point (A) sends a nonce to the smartphone. The smartphone signs this nonce with its private RSA key and sends the signature (sig) back to the access point:

Authentication protocol between the access point (A) and the smartphone (M)

1: NFC: $A \rightarrow M$: $Nonce_A$

2: NFC: $M \rightarrow A$: $sig_M(Nonce_A)$

The access point can then check if the signature was created using the key which corresponds to the public key sent and signed by the server in the authorization protocol.

3.2 Online authorization protocol

The authorization request sent by the access point (A) to the smartphone (M) consists of two segments: the access point ID and a nonce. This information is relayed to the server (S). By using transport layer security (TLS) for the transport, the ID of the smartphone is provided to the server in the form of an X.509 client certificate:

Online authorization protocol between the access point (A), the smartphone (M) and the server (S)

1: NFC: $A \rightarrow M$: $ID_A || Nonce_A$

2: Internet (TLS): $M \rightarrow S$: $ID_A || Nonce_A || Clientcert_M$

3: Internet (TLS): $S \rightarrow M$: $Access\ OK ||$
$sig_S(Pubkey_M || ID_A || Nonce_A || Access\ OK)$

4: NFC: $M \rightarrow A$: $Pubkey_M || ID_A || Nonce_A || Access\ OK ||$
$sig_S(Pubkey_M || ID_A || Nonce_A || Access\ OK)$

The server can decide whether the access request should be granted or denied based on the access point ID, the smartphone ID, current time and access rights stored and managed by the server.

The access point ID, the nonce and the information whether the access was granted is encoded in the authorization answer, together with the public key of the key pair stored on the smartphone. The access point can verify the identity of the phone with the public key, and it can validate the authorization answer by means of a cryptographic signature based on a common secret known by the access point and the server. The current time, which is sent by the server to the access point, can be used to adjust the real-time clock of the access point to prevent time drift. This is especially important, as with offline authorization (Section 3.3), the current time at the access point is used for the access decision.

To reduce the amount of data transferred between the server and the smartphone, all data already known by either side (access point ID, nonce and smartphone public key) are omitted. These data are re-attached by the smartphone before the whole packet is relayed back to the access point.

3.3 Offline authorization protocol

As an alternative to the authorization protocol described in Section 3.2, a ticket-based offline authorization protocol was implemented. The use case of this alternative protocol is mainly to enable access points which are not covered by wireless network connections, for example in cellars. As the tickets are time-based, a mobile must periodically update the stored tickets and an access point must have a real-time clock source (RTC) to implement the offline protocol.

An offline ticket consists of three main parts:

(1) information about the mobile (Mobile ID and the public key used to authenticate the mobile in the authentication protocol);

(2) access information (list of access points, validity timeframe and access times); and

(3) the signature provided by the server to prove the authenticity of the ticket (Figure 2).

To update the list of tickets stored on a mobile, the mobile sends a ticket update request to the server. By using TLS for the transport, the identity of the mobile is provided to the server in the form of an X.509 client certificate in the same way as in the online

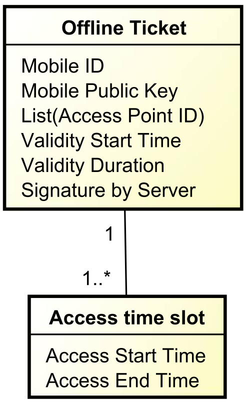


Figure 2.
Structure of an offline ticket

authorization protocol. The server now uses the identification of the mobile to create a list of all access points where the mobile has access during the ticket validity timeframe. For these access points, tickets are created consisting of the known information about the mobile (Mobile ID and public key) and the access information. Finally, each ticket is signed with the symmetric encryption key shared between the server and the access point. These tickets are then sent to the mobile, where they replace the previously stored tickets.

When an access request is initiated by touching the mobile to the NFC reader of the access point, the same authentication and authorization requests are sent to the smartphone as in the online protocol. Based on network connectivity, the smartphone then decides which protocol it uses. Online authorization is preferred if sufficient network connectivity between the mobile and the server is available. In the offline authentication protocol, the ticket belonging to the access point ID sent in the authorization request is presented to the access point by the mobile:

Offline authorization protocol between the access point (A) and the smartphone (M)

1: NFC: $A \rightarrow M$: $ID_A || Nonce_A$

2: NFC: $M \rightarrow A$: $Ticket_{A,M}$

The authorization nonce is ignored by the mobile. The access point can now analyze the authorization answer to grant or deny access.

To perform this analysis, the ticket signature provided by the server is checked first. If this check fails, the access request is ignored. Second, the access information is checked. The first simple check is to look if the access point ID is in the access point list of the ticket. If this is not the case, access is denied.

The access information contains a validity timeframe composed of a start time (UNIX-timestamp, rounded to the nearest minute) and a length in minutes. The access point can now check if the time of its real-time clock is inside of this validity timeframe. If this is not the case, access is denied.

If the current time is inside of the ticket validity timeframe, the access point has to check if access is currently granted. To check this, the access point has to iterate through the access time slots that are stored in the ticket. If one of the access time slots contains the current time of the access point, access is granted, otherwise denied.

Third, the authentication of the mobile is checked using the information about the mobile stored in the ticket. The authentication part of the protocol does not change between on- and offline authorization: The availability of the private part of a key pair on the mobile is still proven by signing a nonce sent from the access point to the mobile. This signature can be validated using the public key of the mobile, which is stored in the ticket signed by the (trustworthy) server. If this check fails, it is a sign of one of two attacks: either the ticket got stolen by an attacker listening to the NFC connection but not in the possession of the private key (wrong key used for the signature) or an old authentication answer was sent in a replay attack (right key used to sign the wrong nonce). If such an attack is detected, access is denied.

If no attack was detected by analyzing the sent data, the access point can grant or deny access based on the access state calculated from the access information.

The Mobile ID used in this offline authorization protocol is not directly related to the authorization process, but can be used by the access point for logging and debugging purposes.

3.4 Traffic reduction in the offline authorization protocol

As mobile network traffic is a limited and costly resource, several measures were taken to reduce necessary traffic at the moment of ticket updates. To get an idea of the possible impact of such reductions let us look at a typical PACS installation done by the business partner of this project. Such PACS systems may have up to 5,000 access points and up to 50,000 end-users. If we further assume that an end-user has access to approximately 4 per cent of all access points and that each offline ticket is valid for exactly one access point, we get a raw-ticket content traffic of about 20GB per day sent from the server to the mobiles. Different protocol and data format overheads can greatly increase traffic needs.

To reduce this traffic, mainly two techniques were implemented. First, data inside the signed ticket, which is already known to the mobile, are not delivered from the server to the mobile – similar to the optimization we implemented for the online authorization protocol. Second, after the server has a list of access profiles for the different access points, only one ticket is generated for several access points with the same access profile.

A very simple ticket as it is sent by a mobile to an access point has a raw content data size of at least 270 bytes. With omitting data already known to the mobile client, about 230 bytes can be saved per ticket sent from the server to the mobile. This allows up to 85 per cent of raw-ticket content traffic savings. The data already known to the mobile consist of the RSA-1536 public key of the mobile in PKCS#1 format and the Mobile ID (24 byte). As the signature of the ticket is server generated, the mobile must restore the exact content that the server omitted. In contrast to the online authorization protocol, the access point ID must be transferred with each ticket to enable the mobile to find the correct ticket to be presented to the access point at access time.

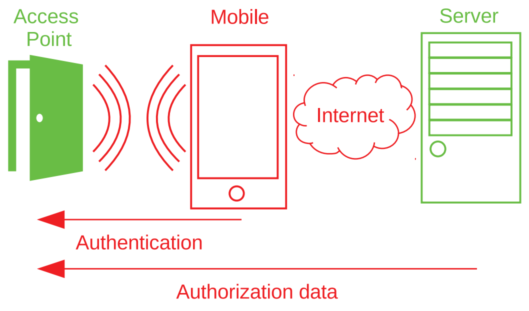
To consolidate tickets where the end-user has the same access profile, the server first creates a map data structure, where each access point ID is mapped to the specific access profile the requesting end-user has for this access point. This map is then grouped by equal access profiles. For each of these groups, one or more tickets are created containing multiple (or all) access point IDs associated to it. The number of access points which are combined into a single ticket is a balance between traffic generated between server and mobile (less tickets are preferred to save traffic) and traffic between mobile and access point (small tickets are preferred to get faster access times) and has to be decided individually for different installations.

4. Attack surfaces

Access control systems are security sensitive systems, a breach of security could lead to physical break-in and subsequently theft, sabotage or espionage. Because of this, security assumptions have to be conservative.

We assume that the smartphone and data transmission channels are insecure (red in Figure 3). Attackers with the right infrastructure can create man-in-the-middle scenarios to monitor or manipulate data sent over the Internet or over the NFC connection. Malware inadvertently installed on a smartphone allows an attacker to gain full control of the installed software, to examine stored data and to run applications.

To allow the access point to securely decide about granting or denying access, two conditions have to be met:



Note: Green components are assumed secure

Figure 3.
Attack surfaces

(1) the authentication of the smartphone has to be secure; and
(2) the authorization data transmitted from the server must not be tampered with and be currently valid.

In this chapter, we describe attacks directly related to the protocol. Further attacks are described in Section 6.

4.1 Attacking online authorization

An attacker with control over the network connection between the smartphone and the server can read and manipulate any transmitted data. To mitigate this type of attack, a TLS-secured connection is used. X.509 certificates are used to authenticate the smartphone and the server to each other. With this technology in place, an attacker can only read the encrypted data, and any manipulation would immediately be detected.

The authorization data sent by the server to the access point are relayed by the smartphone; thus, an attacker with control over the smartphone software can read and alter the transmitted data while it is passing through the smartphone. To allow detection of altered authentication data, the server and the access point share a common secret which is used to digitally sign the transmitted data. By reading the authentication data sent by the server, the only valuable additional information an attacker gains is whether the attacked smartphone has granted access to the access point at the time of transmission.

With the nonce which is sent to the server and back to the access point, a replay attack (in which an attacker sends the same answer multiple times to reuse old authorization data) can be detected. The access point simply compares the received nonce with the sent one to see if the answer corresponds to the current request. To check the integrity of the authorization answer, the access point can create a signature of the payload data and compare it to the signature sent by the server. If the signatures do not match, the message was either not signed by the correct server or changed in transit.

An attacker with control over the smartphone software, and thus over the TLS authentication used to verify the identity of the smartphone to the server, can send multiple authorization requests to the server to gain information about the access rights of the attacked smartphone.

Finally, an attacker can craft a downgrade attack to force the access point to use the offline authorization protocol. As described in Section 3.3, the authorization request sent by the access point to the mobile is the same for the on- and offline authorization protocol, and the mobile decides based on connectivity if the authorization request can be answered by the server (preferred), or if an offline ticket should be presented. Using this possibility, an attacker could possibly gain access at a control point where the access rights were already revoked by presenting an old ticket as described in Section 4.2. To prevent a downgrade attack, an access point can be configured to not support offline access. As an additional measure, the server can be configured to not create offline tickets for specific access points and end-users. Without a valid offline ticket, the authorization request cannot be answered with an answer signed by a server and thus a downgrade to the offline protocol is impossible.

4.2 Attacking offline authorization

The offline authorization protocol provides by design less security than the online authorization protocol. The potential longevity of the tickets and the fact that they are stored on the smartphone (which we consider as insecure) provide additional attack surfaces, especially if an attacker has control over the mobile device. An organization implementing a PACS using mobiles must be aware of the additional security risks of the offline authorization protocol and decide if offline authorization should be enabled for specific mobiles at specific access points, and how long the tickets for different security requirements should be valid.

An attacker can read the content of the tickets stored on the mobile, and thus gain detailed information about access rights to all access points for which a ticket is stored on the mobile phone. If an attacker has access to a malware installed on an attacked smartphone, the attacker can continuously read the ticket contents and gain an access profile over a longer time span. One possibility to mitigate such attacks would be to encrypt the ticket contents instead of only signing it. In the current implementation, encryption of the ticket is not implemented because of two reasons:

(1) the mobile must be able to determine which ticket to present to the access point, and, thus, must be able to read the access point ID list.

(2) the traffic reductions by omitting information known to the mobile as described in Section 3.4 would not work with encrypted tickets.

One possibility to create stronger privacy would be to encrypt only the access time information in the ticket using the symmetric key known to the server and the access point, but leave the Mobile ID, mobile public key and the access point IDs unencrypted.

A security property of the tickets stored on the mobile is that they seem valid to the access point until the validity time frame is over. In the case that the access profiles on the server are changed, the server tries to update the tickets stored on the mobile as fast as possible, using push notifications to trigger an update of the stored tickets. There is no way for the server to notify the access points of tickets which should no longer be valid because of access profile changes, new access profiles are only effective after the mobile has actually performed the ticket update and overwritten the old data. This has two important implications:

(1) if a mobile has no network connection, no new rights can be downloaded, and thus the old tickets are still presented to the access point as valid.

(2) an attacker can abuse this property to present tickets which are still valid based on the validity time frame to an access point even after a ticket update, and, thus, may gain illegitimate access.

To lessen the possible impact of such an attack, the ticket validity time for a specific access point can be reduced accordingly.

4.3 Attacking authentication

While detecting manipulation of authorization data is relatively easy as both ends of the communication are trusted components, the authentication process is more difficult to handle. The security of the authentication protocol relies on the possibility to store a secret key in a secure storage on the smartphone.

When confronted with a simple smartphone software solution, an attacker with a system-level access can simply read the private key which is used to authenticate the smartphone against the access point. This private key can then be used on another device to impersonate the attacked smartphone.

Starting with version 4.3, Android provides support for hardware-based key stores (Android Open Source Project, 2016). Such key stores depend on the availability of security hardware (mostly integrated into the CPU of the smartphone) and on the software implementation of device manufacturers. If a hardware-based key store is present in a device, a smartphone application can use it to securely store the private key of a key pair and to execute private-key operations without the possibility that the Android OS or any third-party applications can extract the private key (Elenkov, 2014, pp. 178-180).

Even if a hardware-based key store is used, an attacker can apply a software relay attack on the key store (similar to the relay attack on a secure element described in Roland *et al.* (2013a) to execute the needed private key operations on the victim's smartphone at the time of access with a second smartphone (Figure 4). To achieve this goal, the attacker uses a smartphone to create a connection with the access point system. He then sends the authentication request by the Internet to a malware application on the victim's smartphone, where the malware can execute the necessary private-key operations to generate the signature needed for authentication and send back the result to the attacker's smartphone.

The software relay attack can only be solved using a separate trusted processing environment with its own NFC connection to securely authenticate the smartphone to the access point. This processing environment executes all private-key operations and sends the result directly to the access point without the possibility that any code

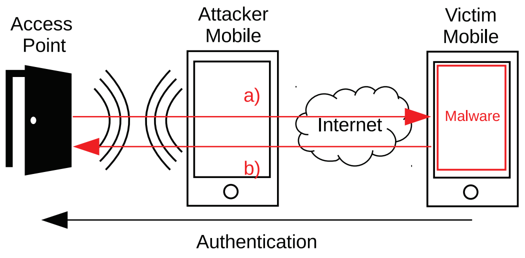


Figure 4. Software relay attack: (a) the authorization request; (b) the authorization answer

executed on the smartphone CPU can access the result. Such a solution which still supports online authorization is described in the next section.

5. Separate secure element

As discussed in the last section, a separate piece of hardware is needed which can communicate to the access point by NFC and to the smartphone (see Figure 5). Such a secure processing environment is typically called a secure element (SE) or trusted execution environment (TEE). By adding this piece of hardware, we gain another secure component in the system which can be used for authentication.

5.1 Secure elements

Most modern SEs or TEEs are very small computers which are used in many different security-sensitive applications – for example banking and credit cards, SIM cards, as an implementation for hardware-based key stores in Android phones, for access cards and also in smartphone NFC solutions. An SE contains at least a processing unit, program memory (typically flash memory), execution memory (RAM) and an interface to allow connections to other systems. Most SEs also contain cryptographic hardware to speed up the execution of cryptographic calculations. Typical interfaces to access the software of a smartcard are eight pin-plated contacts (banking cards), NFC (wireless banking cards) or soldering points (embedded SE).

While older secure element hardware was produced for special use cases, modern SEs contain an operating system with a standardized programming interface. An entity which wants to use secure elements – for example a bank – can develop an applet for their use case, deploy it to a number of secure elements, personalize the element (by executing special functionality of the applet to generate an ID or cryptographic secrets) and disable the programming functionality to guarantee data and application security. Only the issuer or a trusted third party can reprogram the secure element using cryptographically secured methods provided by the card operating system.

The most widespread programming interface for SEs is JavaCard. JavaCard is a slimmed-down Java variant specifically tailored to the security needs and low resources of a secure element. Special methods of persistence and transaction support are integrated in the language, and cryptographic methods are supported. Communication between a card terminal and the application on the card is standardized as ISO 7816. The protocol is a simple serial request/answer protocol which utilizes application protocol data units (APDU) to transfer information. A subset of these APDUs is standardized, others can be used in proprietary applications.

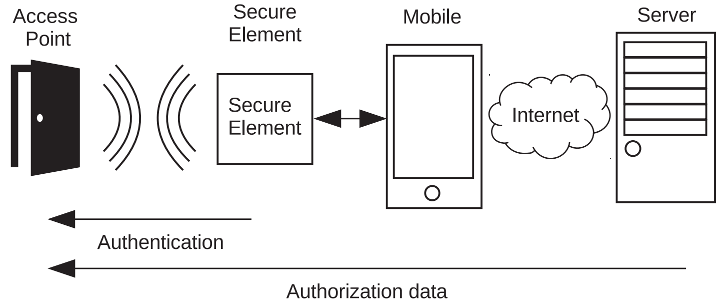


Figure 5.
Use of a secure element for authentication

For our project, we analyzed different programmable models of SEs which can be used in combination with a smartphone. One requirement was that the SE contains two separate interfaces, an NFC interface to communicate to the access point and an interface to communicate with the smartphone. Also, the application on the card must have the possibility to determine which communication channel is used to prevent the execution of the authentication method by software running on the smartphone CPU. We found such an SE in the form of a microSD card with a built-in NFC transceiver: the CredenSE 2.10J developed by DeviceFidelity (DeviceFidelity, 2013). The SE embedded in this microSD card can be accessed from a mobile phone through a specific application programming interface (API).

5.2 PACS with a microSD-SE

In the PACS we designed and implemented, we had to meet two security properties:

(1) the smartphone authentication; and

(2) the transfer of authorization data from the server to the access point.

To achieve this goal, we implemented a JavaCard applet which allows to authenticate the card to the access point and to relay data to the smartphone and subsequently to the server as shown in Figure 6.

All connections to the SE have to be initiated by either the smartphone or the access point, and the SE does not support communication with both endpoints at the same time. Due to these circumstances, we implemented a stateful JavaCard applet to relay the information to the smartphone and back. The NFC transceiver of the SE we used in the project has to be activated by an application running on the smartphone using a driver software. The driver software also allows to register a callback listener which gets notified when the secure element NFC transceiver enters or leaves the electromagnetic field of an NFC reader.

To use the PACS, the end-user has to activate the NFC transceiver of the SE by using the smartphone application which we developed. In the next few paragraphs, we will describe the different phases of the process which is used to authenticate and authorize a phone to an access point. The transaction is also described as sequence diagram in Figure 7.

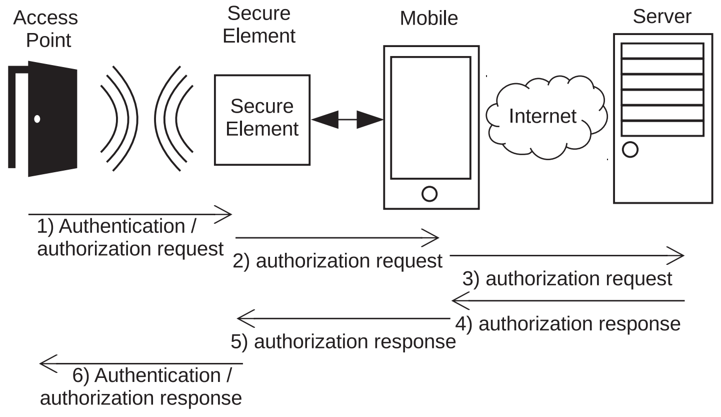

Access Point
Secure Element
Mobile
Server
Secure Element
Internet
1) Authentication / authorization request
2) authorization request
3) authorization request
4) authorization response
5) authorization response
6) Authentication / authorization response

Figure 6. Overview of the protocol of a PACS with a microSD-SE

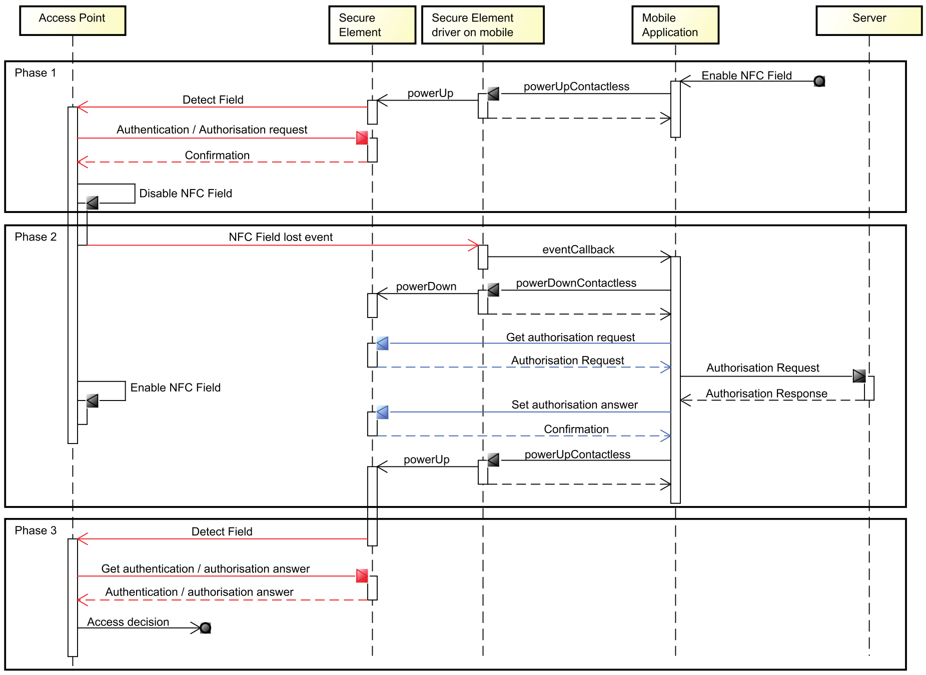


Figure 7.
Sequence diagram of an access request using the secure element solution for secure authentication

In the first phase of the process, after the smartphone user activates the NFC interface and touches the access point, the access point initiates the connection and sends the authorization and authentication request to the JavaCard applet which enters a second state. The access point then has to disable the NFC field to indicate the completion of the first phase to the mobile.

In the second phase of the process, the smartphone application initiates the connection to the SE after having received the callback that the NFC field of the reader has disappeared (because the reader shut down the field). The smartphone application establishes a connection to the SE and sends a command to ask for the authorization request. While still holding the connection to the SE, this request is forwarded to the server by the smartphone application and – after having received the authorization answer from the server – the answer is sent back to the JavaCard applet which stores it and enters the third phase. The smartphone now utilizes the driver software of the SE to enable the NFC transceiver and the access point gets a chance to detect the presence of the secure element.

In the third phase, the connection is again initiated by the access point. The access point can now read the answers for both the authentication and authorization requests sent in the first phase. The answer to the authorization request is the one sent by the smartphone in phase 2, the answer to the authentication request is computed by the JavaCard applet at this point in time. With these answers, the access point has all necessary information to validate the authentication and authorization of the smartphone.

The authentication request is transferred by NFC directly from the access point to the SE. As the SE is responsible for creating the authentication response and as the answer is directly transferred back to the access point by NFC, an attacker controlling the smartphone software cannot execute any public key operations nor read the private key. With the SE not allowing such operations by the smartphone, software relay attacks as described in Section 4.3 are not possible. By intercepting the connection between the access point and the secure element, an attacker with sufficient infrastructure can execute a hardware relay attack as described in Section 6.

The transaction duration is increased by using this method because of the way the callback on NFC field events is implemented in the software driver. Internal in the driver, a loop checks the NFC field status every 0.5 seconds and notifies the callback listener on change of the NFC field status. We manage to trick the driver into performing this check every 0.1 seconds. With this change in frequency, the additional time compared to a software-only solution (thus, the maximal time needed until the callback is called plus the time needed for turning off and on the NFC transceiver of the secure element) is around 150 ms.

6. Physical attacks

In this section, we describe attacks where an attacker needs to gain physical access to the smartphone used in the PACS. There are two main attacks in this category: theft and the so called hardware relay attack (Francis *et al.*, 2011). Both of these attacks are also possible with a classic NFC card-based PACS.

To execute a hardware relay attack, an attacker needs two NFC-capable smartphones. These smartphones are connected – for example via the Internet – and act as a proxy for NFC-transmitted data. With this setup, an attacker can extend the reach of the NFC transaction. To use this often-described attack (Francis *et al.*, 2011; Lee, 2012; Arnosti and Gruntz, 2014; Maass *et al.*, 2015), the attacker places one of the smartphones at the reader, and the other on the victim's smartphone (see Figure 8). The smartphone placed at the access point now forwards all requests sent by the reader to the second attacker smartphone. The requests then get forwarded to the attacked smartphone and the answers are sent back using the same way. As this attack is focused at the NFC transmission reach, it works for the off- and online authorization protocol.

When a smartphone is stolen, the attacker has the same possibilities as with the hardware relay attack (except that the relay infrastructure is not necessary). In our microSD-SE-based PACS, physical attacks and theft are addressed with the following two risk-reducing factors.

First, with the online protocol, the authorization data are loaded from the server at access time, an attacker cannot use a stolen smartphone after the theft was detected and the access rights got revoked server-side. This does not hold for the

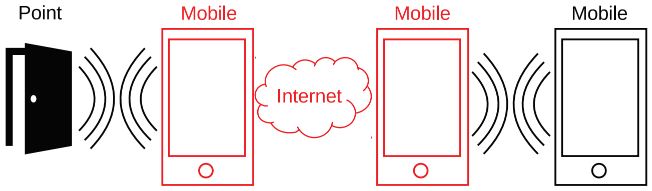


Figure 8. Hardware relay attack: an attacker uses two smartphones to extend the reach of the NFC transaction

offline protocol, i.e. an attacker can possibly gain access to access points as long as the ticket is valid.

Second, the end-user needs to manually start the NFC transaction (in particular for the microSD-SE-based solution). For this, he has to unlock the smartphone and use a button inside of an installed application (but it is also possible to use a widget to place this button on the home screen of the smartphone). Because of the limited possibilities of the used secure element, it is necessary that the SEs NFC interface is activated prior to a transaction, but the decision that this has to be done manually by the end-user is a security feature. The end-user (or the institution issuing the smartphone) can decide to configure security features like a display lock on the smartphone to complicate unauthorized use of the smartphone. If an attacker wants to perform a physical attack, he needs to activate the NFC interface of the secure element, and thus first needs to be able to operate the smartphone application. If the HCE mode of current Android operating systems is used, it is possible to implement the service in a way that only the screen must be enabled or unlocked – if only enabling the screen is necessary, security is obviously greatly decreased but usability greatly increased.

However, if it is possible for an attacker to attack a smartphone at the same time physically and digitally, he can start an NFC transaction software-wise and use the NFC connection either in a hardware relay attack or directly at the access point. Such an attack works as long as it is not detected and the access rights are still valid on the server. To mitigate the effects of such an attack, a PACS needs to rely on additional authentication technologies beyond simply checking the ownership of a smartphone. Examples of such technologies are checks of biometric features like fingerprints or checks of knowledge like a PIN at the access point.

In the case that an attacker manages to perform a hardware relay attack, he can read and manipulate the data sent in between the access point and the smartphone. The design of the protocol (as discussed in Section 3) guarantees that an attacker cannot manipulate the data undetected. Any data the attacker can read while eavesdropping this connection is either public, valid for only one transaction or of no added value. In particular, the information whether access is granted for the attacked smartphone at the time of the interception has no added value as the attacker could simply watch if the physical access point allows access or not.

7. PACS comparison

A classical PACS works with cards, but there exist systems where the card is emulated by a UICC-SE (SIM card) or an SE embedded in a smartphone. In this paper, we have shown how such a smartphone-based PACS can be implemented independent of mobile network operators, the services of a trusted service managers and handset manufacturers, both with HCE and with a microSD-SE.

In this section, we compare the following four PACS variants (these variants correspond to the columns in Table I):

(1) *Card*: With this variant, we refer to a card-only solution. Such a system could be based on MIFARE-DESFire cards which are ISO 14443-4 compliant and which are accessible over NFC.

(2) *SE*: This variant stands for all solutions which depend on third parties like MNOs, TSMs or handset manufacturers. Examples are the solution from Kaba

Table I. Comparison between different PACS

Criteria	*Card*		*SE* UICC or embedded		*HCE* Software-only		*microSD-SE* SE and software	
Third-party independence (MNO/TSM/manufacturer)	(+)	No dependency	(−)	MNO/TSM dependency	(+)	No dependency	(+)	No dependency
Key security (security of key storage)	(+)	Secure	(+)	Secure	(0)	Device dependent	(+)	Secure
Hardware relay attack (theft/physical security)	(−)	Always as NFC is always on	(−)	Always as NFC is always on	(+)	On unlocked device only	(+)	After user interaction only
Software relay attack (software proxy)	(+)	Not possible	(+)	Not possible	(−)	Insecure	(+)	Secure
Time to open (usability)	(+)	System dependent	(+)	System dependent	(+)	Same performance as card	(0)	Like HCE + 150 ms
Online authorization (for offline access points)	(−)	Not possible	(0)	System dependent	(+)	Possible	(+)	Possible
Offline access tickets (for offline mobiles)	(−)	Terminal	(0)	Over MNO or terminal	(+)	Online over smartphone	(+)	Online over smartphone

(2016) which is hosted by Legic Connect or Tapit, a solution from Swisscom (2015). Tapit has been denounced.

(3) *HCE*: This is the solution described in Section 3 which is implemented without using a SE.

(4) *microSD-SE*: The microSD-SE approach uses a separate SE to solve some security problems of a HCE-only solution as shown in Section 5. In our project, we used a microSD-SE from DeviceFidelity.

We performed the comparison along the following criteria, and for all implementation variants we marked each criterion in Table I either with a + sign (positive), a − sign (negative) or with a 0 (neutral):

- *Third-party independence*: Except for the SE variant, all other variants are independent of a third party, i.e. the PACS service provider has full control over the technology and can provide its own applications.
- *Key security*: Under this criterion, we compared how secure user credentials can be stored. For the HCE variant, such credentials can be stored in a hardware-based key store if such a feature is provided by the phone hardware.
- *Hardware relay attack*: A hardware relay attack can be executed if the (emulated) card is accessible without further interaction. This is obviously the case for cards, but also for the SE-variant (for usability reasons, otherwise the access token could not be used if the phone has been turned off). For the HCE and the microSD-SE variants the hardware relay attack can only be executed if the device has been unlocked (and a special application has been started in addition for the microSD-SE variant). HCE on the other hand is vulnerable to this attack.
- *Software relay attack*: Obviously, the card-only variant is immune to software attacks, and for the SE-variant the software relay attack is also not possible if the system is implemented properly, thus does not allow that private key operations are executed from the phone host. This is the condition which is also met by the microSD-SE which we used in the implementation of our project.
- *Time to open*: We expect that all solutions show a comparable timing, except for the microSD-SE variant which takes about 150 ms longer than the HCE variant.
- *Online authorization*: The online authorization is possible for the HCE and the microSD-SE approaches. We do not know any systems where an UICC-SE or an embedded SE is used while also providing online authorization data via NFC, and we do not know if all requirements are met to enable such an implementation.
- *Offline access rights*: For card-only systems, access rights can be stored on the cards (in addition to the authentication credentials), but then the user has to reload this card at specific terminals. For the UICC-SE approach, a reload of access tokens can be performed over a terminal or over MNO-specific technologies. It would also be possible to load access rights to an UICC-SE

using the smartphone's Internet connection, but the SE would have to distinguish the access paths to prevent the software relay attack.

According to the criteria we used in this comparison, the microSD-SE approach has a lot of advantages in the security and usability criteria. Financially, the microSD approach is relatively expensive, as the cards need to be bought for every user. As the same infrastructure can be used for the HCE variant, the higher financial investment directly correlates to higher security.

8. Related work

Several NFC-based access control systems for smartphones have been described and implemented, but most of them are not public. NFC-based PACS typically either use a UICC-SE (Kaba, 2016, Swisscom, 2015) or they are HCE-based (Telcred AB, 2015).

Before HCE was available in the Android framework, an alternative was to use the inverse reader mode (Saminger *et al.*, 2013). Systems that adopted this approach are, for example, AirKey[1] from EVVA (www.evva.at) and NFC Porter[2] from IMA (www.nfcporter.com). These systems both store their credentials on the mobile phone for offline use (Roland and Langer, 2013).

Most UICC-SE solutions follow the online access point model described in Section 2, that is the access points are typically connected to the authentication server, and over these connected access points the data stored on the SE can be updated securely.

All HCE-based solutions suffer from possible software relay attacks. The same holds for all other access control solutions which store the credentials in a SE, but use other communication technologies like Bluetooth Smart (BLE) to connect to the access point and to the authentication server and, thus, use an application running on the smartphone to move authentication data. A system which follows this approach is HID's Mobile Access[3](www.hidglobal.com). The relay attack problem is often mitigated by additional security checks, such as the need to enter a PIN or the check of biometric features directly at the access point.

The general structure of our microSD-SE-based solution follows the model described in Dmitrienko *et al.* (2012), but that model explicitly excludes relay attacks as the focus is on delegatable authentication for NFC-enabled smartphones. To mitigate relay attacks, distance-bounding techniques are proposed. These techniques determine an upper bound on the round-trip time of request-response pairs (Drimer and Murdoch, 2007). However, this approach cannot be applied to online solutions where the access server has to be connected before the response is sent back to the access point.

9. Results

We have presented a smartphone-based PACS in which the access points communicate with the access server over the smartphone connected to the access point. This relay approach allows different attacks, in particular the hardware and the software relay attack. The hardware relay attack can be mitigated by protecting the smartphone with a screen lock.

We have shown that the software relay attack can be prevented with a microSD-based SE which communicates directly to the access point. For the online authorization, the microSD-SE must be able to communicate with the server over the

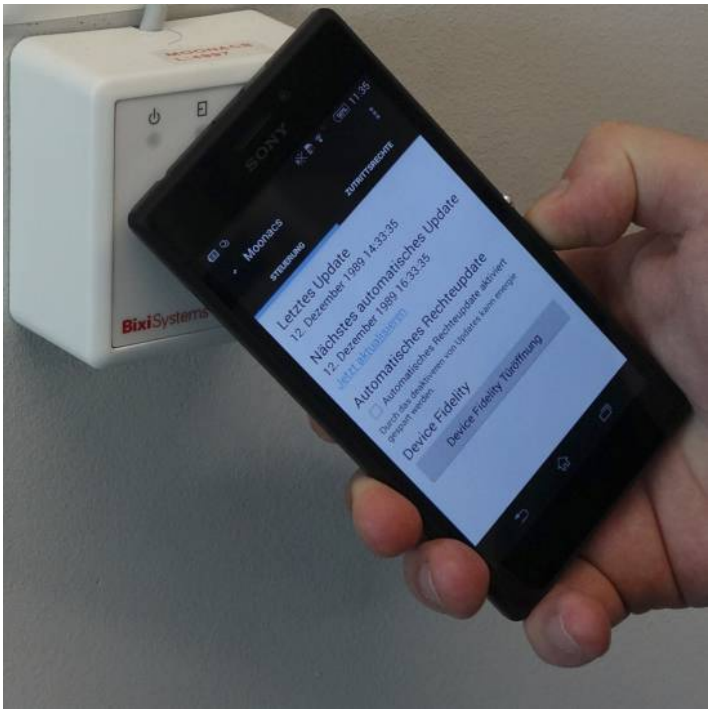

BixiSystems
SONY
Moonacs
STEUERUNG
ZUTRITTSRECHTE
Letztes Update
12. Dezember 1989 14:33:35
Nächstes automatisches Update
12. Dezember 1989 16:33:35
Jetzt aktualisieren
Automatisches Rechteupdate
Automatisches Rechteupdate aktiviert
Durch das deaktiveren von Updates kann energie gespart werden
Device Fidelity
Device Fidelity Türöffnung

Plate 1.
Implementation of our PACS in action

smartphone. Our contribution is to show that such an approach can be implemented and that the speed is still acceptable. A picture of the implemented solution in action is shown in Plate 1.

A drawback of the microSD-SE approach beyond additional costs is that the microSD-card is typically provisioned by a single service provider (in our case, this would be the provider of the PACS). An end-user wanting to use microSD-SE-based PACS from multiple service providers would have to switch microSD-cards.

A PACS usually has to support several levels of security. With the solution presented in this paper, the same infrastructure and the same protocols can be used for the HCE and the microSD-SE variants. The less-secure HCE variant could be rolled out to most of the users who have access to a building, and users having access to high-security areas inside that building could use the microSD-SE-based solution.

Notes

1. available at: www.evva.at
2. available at: www.nfcporter.com
3. available at: www.hidglobal.com

References

Android Open Source Project (2016), *Android Keystore System*, available at: https://developer.android.com/training/articles/keystore.html

Arnosti, C. and Gruntz, D. (2014), "Man-in-the-middle: analyse des datenverkehrs bei NFC-zahlungen", *IMVS Fokus Report*, Vol. 8 No. 1, pp. 24-31.

DeviceFidelity (2013), *CredenSE 2.10J Classic is NFC Card-Emulation and Certified Java Card SE in a MicroSD*, available at: http://devifi.netfirms.com/devifi.com/assets/DeviceFidelity_CredenSE.pdf

Dmitrienko, A., Sadeghi, A.-R., Tamrakar, S. and Wachsmann, C. (2012), "SmartTokens: delegable access control with nfc-enabled smartphones", *International Conference on Trust & Trustworthy Computing (TRUST)*, Vienna, Springer, Vol. 7344, pp. 219-238.

Drimer, S. and Murdoch, S.J. (2007), "Keep your enemies close: distance bounding against smartcard relay attacks", *Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium, Berkeley, CA*, pp. 257-274.

Elenkov, N. (2014), *Android Security Internals: An In-Depth Guide to Android's Security Architecture*, 1st ed., No Starch Press, San Francisco, CA.

Francis, L., Hancke, G., Mayes, K. and Markantonakis, K. (2011), "Practical relay attack on contactless transactions by using NFC mobile phones", *Cryptology ePrint Archive*, Report 2011/618, available at: http://eprint.iacr.org/2011/618

Janssen, T. and Zandstra, M. (2014), *HCE Security Implications*, UL Transaction Security White Paper, 8 January..

Kaba (2016), *Mobile Access Solutions*, available at: www.kaba.com/en/kaba/innovation/654636/mobile-access-solutions.html (accessed 18 March 2016).

Lee, E. (2012), *NFC Hacking: The Easy Way*, Presentation held at the DEFCON 20 conference in Las Vegas, July 26-29.

Maass, M., Müller, U., Schons, T., Wegemer, D. and Schulz, M. (2015), "NFCGate: an NFC relay application for android", *Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks, New York, NY*, available at: http://doi.org/10.1145/2766498.2774984

NFC World (2016), *NFC Phones: The Definitive List*, available at: www.nfcworld.com/nfc-phones-list

Roland, M. and Langer, J. (2013), "Comparison of the usability and security of NFC's different operating modes in mobile devices", *E&i Elektrotechnik und Informationstechnik*, Vol. 130 No. 7, pp. 201-206.

Roland, M., Langer, J. and Scharinger, J. (2013a), "Applying relay attacks to google wallet", *Proceedings of the 5th International Workshop on Near Field Communication (NFC2013), Zürich, Switzerland, February*.

Saminger, C., Grünberger, S. and Langer, J. (2013), "An NFC ticketing system with a new approach of an inverse reader mode", in *Proceedings of the 5th International Workshop on Near Field Communication (NFC2013), Zürich, Switzerland, February*.

Swisscom (2015), *The Swiss Wallet of Tomorrow*, available at: www.tapit.ch/en (accessed 20 August 2015).

Telcred AB (2015), *A New Approach to Access Control*, available at: http://telcred.com (accessed 20 August 2015)

Further reading

Arnosti, C., Gruntz, D. and Hauri, M. (2015), "Secure physical access with NFC-enabled smartphones, in Chen, L., Steinbauer, M., Khalil, I. and Anderst-Kotsis, G. (Eds)", *Proceedings of the 13th International Conference on Advances in Mobile Computing and Multimedia (MoMM15)*, *Brussels*, *11-13 December*, *ACM*, pp. 140-148.

Coskun, V., Ozdenizci, B. and Ok, K. (2013), “A survey on near field communication (NFC) technology”, *Wireless Personal Communication*, Vol. 71 No. 3, pp. 2259-2294.

Google (2016), *Android Keystore System*, available at: https://developer.android.com/training/articles/keystore.html

Corresponding author
Dominik Gruntz can be contacted at: dominik.gruntz@fhnw.ch