Secure Physical Access with NFC-enabled Smartphones

This paper presents a smartphone-based physical access control system in which the access points are not directly connected to a central authorization 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. 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. In our solution we use the following secure element archiectures: Host Card Emulation (HCE) and a microSD-based secure element. We show that the HCE approach cannot solve the relay attack under conservative security assumptions and we present and discuss an implementation based on a microSD secure element that still allows the access points to connect to the authorization server upon every access albeit the access points are not connected with it.

Christof Arnosti, Dominik Gruntz, Marco Hauri | dominik.gruntz@fhnw.ch

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[1] 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 [12].

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 then 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 are not new. Such systems typically store the access rights on the 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, since these phones are programmable and network connected, they provide a large surface for attacks [8]. 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 the next two sections we describe the general structure of our solution and the designed protocols. The security properties are analyzed in the section „Attack Surfaces“ and a solution to the authorization problem is presented in the section „Separate Secure Element“. Finally, we describe further (physical) attacks and compare the different PACS approaches. After an overview of related approaches we conclude with the results.

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 au-

1 This is the submitted version of a paper which has been accepted for the 13th International Conference on Advances in Mobile Computing and Multimedia (MoMM15), 11 - 13 December 2015 in Brussels, Belgium.

thentication-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 the 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 is 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 has 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 since 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 don't 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 is 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), but we do not discuss this in this paper since the security implications do not change.

Protocol Design

As shown in Figure 1, the protocol scheme of our smartphone-based PACS consists of two protocols: the authentication protocol to authenticate the smartphone to the access point and the authorization protocol to transmit authorization data between the server and the access point, using the smartphone as a relay.

The authentication protocol is based on public-key cryptography. On the smartphone, an RSA key pair is stored. The public key of this key pair is known to the authorization server and is included in the authorization answer which is signed by the server. This signature links authorization and authentication by proving that the authentication response is made by the device the authorisation response belongs to.

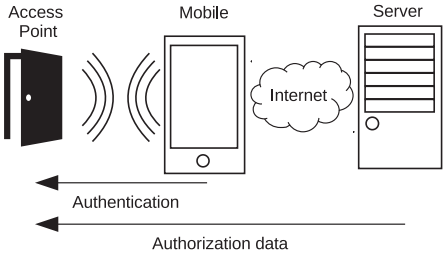

Figure 1: Smartphone solution

Both parts of the protocol are initiated by the access point and the requests for both parts are simultaneously sent from the access point to the smartphone using NFC. The answers are also simultaneously sent by the smartphone to the access point.

Authentication Protocol

To authenticate the smartphone (M), the access point (A) sends a nonce to the smartphone. The smartphone signs this nonce and sends the signature (sig) back to the access point. The access point then can check if the signature was created using the private key belonging to the same key pair as the public key sent and signed by the server in the authorization protocol.

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

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 SSL for the transport, the ID of the smartphone is provided to the server in the form of an X.509 client certificate.

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.

1. NFC: $A \rightarrow M: ID_A \,||\, Nonce_A$
2. Internet (TLS): $M \rightarrow S: ID_A \,||\, Nonce_A \,||\, Clientcert_M$
3. Internet (TLS): $S \rightarrow M: AccessOK \,||\, sig_S(Pubkey_M \,||\, ID_A \,||\, Nonce_A \,||\, AccessOK)$
4. NFC: $M \rightarrow A: Pubkey_M \,||\, ID_A \,||\, Nonce_A \,||\, AccessOK \,||\, sig_S(Pubkey_M \,||\, ID_A \,||\, Nonce_A \,||\, AccessOK)$

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.

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

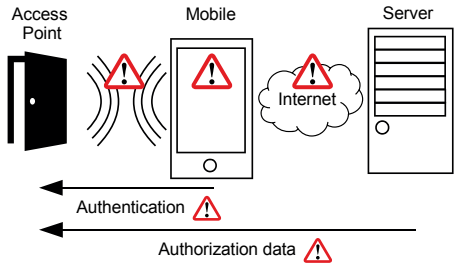

Figure 2: Attack surfaces: only the access point and the server assumed secure

data are re-attached by the smartphone before the whole packet is relayed back to the access point.

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 (marked with an insecure sign in Figure 2). 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: First, the authentication of the smartphone has to be secure; and second, the authorization data transmitted from the server must not be tampered with.

In the next two sections we describe attacks directly related to the protocol. Further attacks are described in the section „Physical Attacks“.

Attacking 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 is relayed by the smartphone, thus an attacker with control over the smartphone software can read and alter the transmitted data while it's 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 access rights 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 correspondents 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 don't 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.

Attacking Authentication

While detecting manipulation of authorization data is relatively easy since 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 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 [1]. Such key stores depends 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

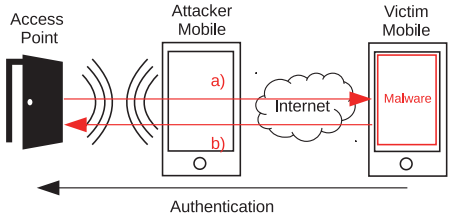

Figure 3: Software relay attack: a) the authorisation request, b) the authorisation answer

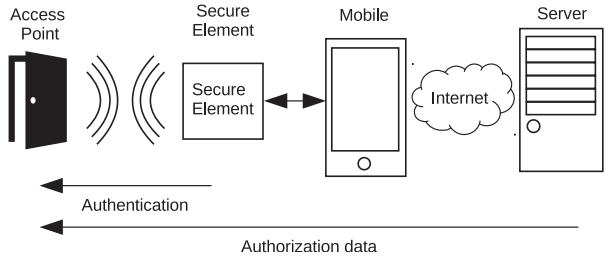

Figure 4: Use of a secure element for authentication

third-party applications can extract the private key [6, 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 by Michael Roland [14]) to execute the needed private key operations on the victim's smartphone at the time of access with a second smartphone (Figure 3). 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 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 executed on the smartphone CPU can access the result. Such a solution which still supports online authorization is described in the next section.

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 4). 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.

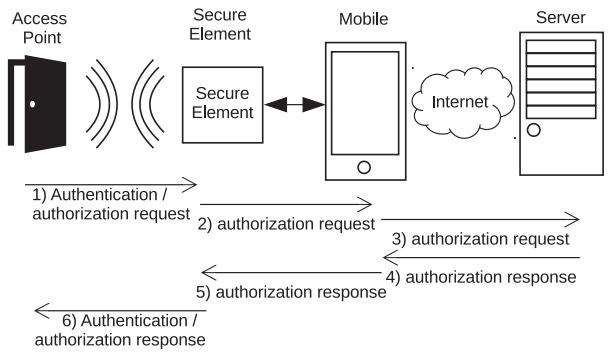

Figure 5: Overview of the protocol of a PACS with a microSD-SE

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 8 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 are standardized, others can be used in proprietary applications.

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

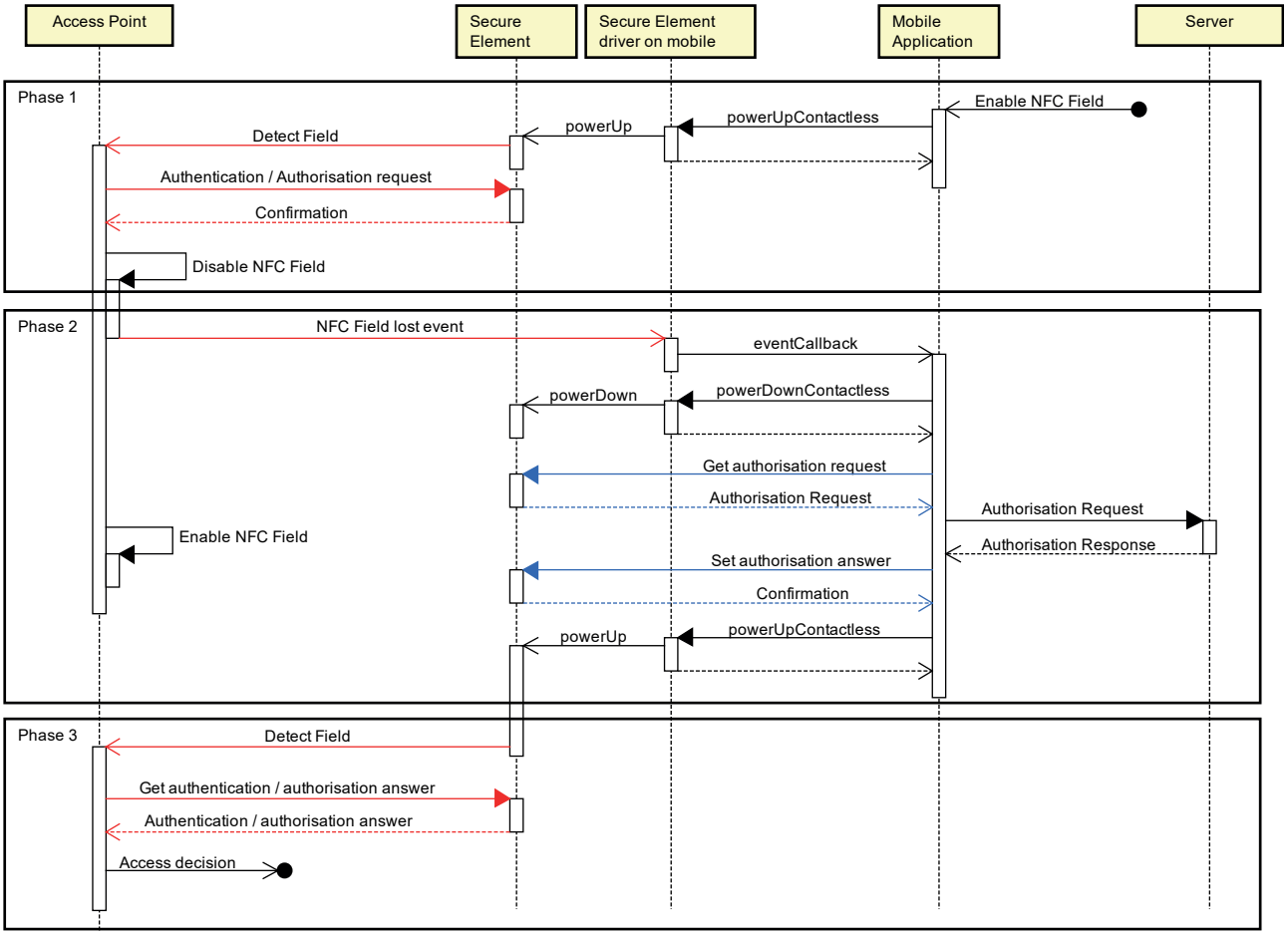

Figure 6: Sequence diagram of an access request using the secure element solution for secure authentication

transceiver: the CredenSE 2.10J developed by DeviceFidelity [3]. The SE embedded in this microSD card can be accessed from a mobile phone through a specific API.

PACS with a microSD-SE

In the PACS we designed and implemented we not only needed to authenticate the smartphone, but also at the same time to transfer 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 as well as to relay data to the smartphone and subsequently to the server as shown in Figure 5.

All connections to the SE have to be initiated by either the smartphone or the access point, and it's not possible for the SE to communicate 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 a 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 6.

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 signal to the smartphone application that the first phase of the process is finished.

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 is left (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 the answer is

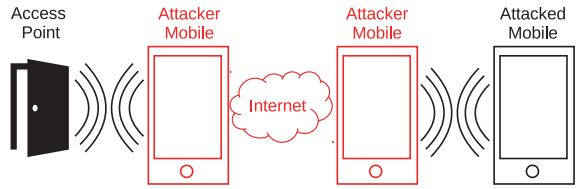

Figure 7: Hardware relay attack: An attacker uses two smartphones to extend the reach of the NFC transaction

sent back to the JavaCard applet which stores it and enters the third state. 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. Since the SE is responsible for creating the authentication response and since 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 the section „Attacking Authentication“ 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 the next section.

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.

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 [7]. 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 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 [7, 10. 2. 11], the attacker places one of the smartphones at the reader, and the other on the victim's smartphone (see Figure 7). 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.

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 PACS, physical attacks and theft are addressed with the following two risk-reducing factors.

First, as the authorization data is 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.

Second, the end-user needs to manually start the NFC transaction. For this, he has to unlock the smartphone and use a button inside of an installed application (but it's 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's necessary that the SEs NFC interface is activated prior to a transaction, but the decision that this has to be manually done 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.

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.

	Card	SE UICC or embedded	HCE software-only	microSD-SE SE & 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 online access points)	(–) not possible	(0) system dependent	(+) possible	(+) possible
Offine access rights (for online smartphones)	(–) terminal	(0) over MNO or terminal	(+) online over smartphone	(+) online over smartphone

Table 1: Comparison between different PACS

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 the section „Protocol Design“) 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.

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 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.
- *SE:* This variant stands for all solutions which depend on third parties like MNOs, TSMs or handset manufacturers. Examples are the solution from Kaba [9] which is hosted by Legic Connect or Tapit, a solution from Swisscom [16]. Tapit will be denounced by the end of 2015.
- *HCE:* This is the solution described in the section „Protocol Design“ which is implemented without using a SE.
- *microSD-SE:* The microSD-SE-approach uses a separate SE to solve some security problems of a HCE-only solution as shown in the section „Separate Secure Element“. 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 1 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 by 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 and does not allow that private key operations are executed from the phone host.

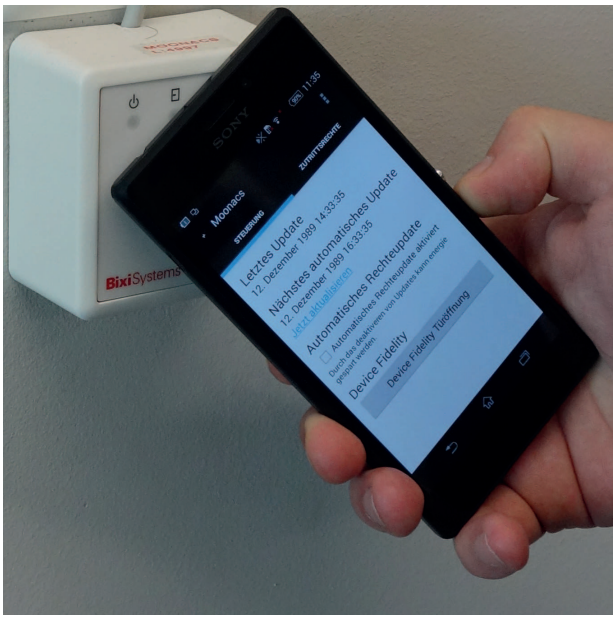

Figure 8: Implementation of our PACS in action

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 don't know any systems where an UICC-SE or an embedded SE is used while also providing online authorization data via NFC, and we don't know if all requirements are met to enable such an implementation.
- *Offline access rights:* Although we focused on online authorization in this paper, our solution can also 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 as soon as the smartphone is connected. 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 in order 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 since 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.

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 [9,16] or they are HCE-based [17].

Before HCE was available in the Android framework, an alternative was to use the inverse reader mode [15]. Systems that adopted this approach are, for example, AirKey[2] from EVVA and NFC Porter[3] from IMA. These systems both store their credentials on the mobile phone for offline use [13].

Most UICC-SE solutions follow the *online access point* model described in the section „PACS Models", i.e. 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[4]. 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 [4], 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 [5]. 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.

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.

2 http://www.evva.at
3 http://www.nfcporter.com
4 http://www.hidglobal.com

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 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 Figure 8.

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 as well as 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.

Acknowledgements

We 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.

References

[1] Android open source project. Android keystore system. https://developer.android.com/training/articles/keystore.html. Accessed: 2015-08-21.

[2] C. Arnosti and D. Gruntz. Man-in-the-Middle: Analyse des Datenverkehrs bei NFC-Zahlungen. IMVS Fokus Report, 8(1):24-31, 2014.

[3] DeviceFidelity Inc. CredenSE 2.10j classic is NFC card-emulation and certied JavaCard SE in a MicroSD. 2013. http://devifi.netfirms.com/devifi.com/assets/DeviceFidelity_CredenSE.pdf

[4] A. Dmitrienko, A.-R. Sadeghi, S. Tamrakar, and C. Wachsmann. SmartTokens: Delegable access control with NFC-enabled smartphones. In International Conference on Trust & Trustworthy Computing (TRUST), volume 7344 of Lecture Notes in Computer Science (LNCS), pages 219-238. Springer, June 2012.

[5] S. Drimer and S. J. Murdoch. Keep your enemies close: Distance bounding against smartcard relay attacks. In Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium, SS'07, pages 7:1-7:16, Berkeley, CA, USA, 2007.

[6] N. Elenkov. Android Security Internals: An In-Depth Guide to Android's Security Architecture. No Starch Press, San Francisco, CA, USA, 1st edition, 2014.

[7] L. Francis, G. Hancke, K. Mayes, and K. Markantonakis. Practical relay attack on contactless transactions by using NFC mobile phones. IACR Cryptology ePrint Archive, Report 2011/618, 2011. http://eprint.iacr.org/2011/618.

[8] T. Janssen and M. Zandstra. HCE security implications. Technical report, UL Transaction Security, 2014.

[9] Kaba. Mobile access solutions. http://www.kaba.com/en/kaba/innovation/654636/mobile-access-solutions.html.

[10] E. Lee. NFC Hacking: The Easy Way, 2011.

[11] M. Maass, U. Müller, T. Schons, D. Wegemer, and M. Schulz. NFCGate: An NFC Relay Application for Android. In Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks, WiSec '15, New York, NY, USA, 2015.

[12] NFC World. NFC phones: The denitive list. http://www.nfcworld.com/nfc-phones-list. Last updated on 21 August 2015.

[13] M. Roland and J. Langer. Comparison of the usability and security of NFC's dierent operating modes in mobile devices. e&i Elektrotechnik und Informationstechnik, 130(7):201-206, 2013.

[14] M. Roland, J. Langer, and J. Scharinger. Applying relay attacks to google wallet. In 5th International Workshop on Near Field Communication (NFC), pages 1-6, Feb 2013.

[15] C. Saminger, S. Grunberger, and J. Langer. An NFC ticketing system with a new approach of an inverse reader mode. In 5th International Workshop on Near Field Communication (NFC), Feb 2013.

[16] Swisscom. The swiss wallet of tomorrow. http://www.tapit.ch/en.

[17] Telcred AB. A new approach to access control. http://telcred.com. Accessed: 2015-08-20.