Decoupling Authentication from the Usage of Services

Technical Report

TECHNISCHE UNIVERSITÄT DARMSTADT

Department of Computer Science
Theoretical Computer Science
Cryptography and Computer Algebra

Moritz Horsch[1] Johannes Braun[1] Alex Wiesmaier[2]

[1]Technische Universität Darmstadt
Hochschulstraße 10, 64283 Darmstadt, Germany
{jbraun,horsch}@cdc.informatik.tu-darmstadt.de

[2]AGT Group (R&D) GmbH
Hilpertstraße 35, 64295 Darmstadt, Germany
awiesmaier@agtinternational.com

Abstract

Services that require authentication are used via a multitude of different systems. Thereby, the protection of the user's credentials depends on the security of any single system applied once in a while for the usage of services by the user. Thereby, authentication based on user name and password is vulnerable in many ways, for instance, malicious software, eavesdropping, and social engineering. Authentication mechanisms applying smart cards counteract such attacks as the possession of the card is a second security factor besides the knowledge of a personal identification number (PIN). However, in many cases smart cards are not applicable due to the absence of adequate card readers. Furthermore, applying a smart card on potentially insecure systems also bears vulnerabilities like eavesdropping on the PIN entry or remote attacks. By physically decoupling the authentication, we present a new approach to perform smart card based authentication completely independent from the system with which the user accesses a certain service. Thus, ubiquitous applicability of smart card based authentication is enabled as the problem of missing hardware is solved. Additionally, the requirement of system security is reduced to the single authentication device where the user exclusively enters authentication credentials.

1 Introduction

Despite the well known security risks[1], authentication to services and systems is mostly based on user name and password. Thereby, such an authentication only relying on the knowledge of certain credentials bears great risks. Passwords can be eavesdropped by malicious software. Furthermore, many users tend to reuse the same passwords for various services leading to many vulnerabilities.

Authentication mechanisms based on knowledge and possession have many security advantages. However, for a widespread utilization of such schemes, a universal and flexible applicability is fundamental, but is hardly achieved in practice. Using smart cards to realize the requirement of possession requires card readers which are often not available. Furthermore, foreign devices, for instance, in public places are not trustworthy. They might be manipulated by an adversary or just not fulfill the security requirements. An illustrative example are ATMs, which are often manipulated by adversaries to grab the user's credentials to finally be able to withdraw money from the user's account. However, in many cases the manipulation is hardly detectable for ordinary users.

Mobile devices like smart phones are suitable to close this gap, because they are widely spread, always available and have a strong binding to the owner. Thus, such a device can serve as a Personal Security Environment (PSE) and realize a user centric authentication system. Equipped with the adequate technologies, the PSE can serve as a key store, signature generation device or card reader (cf. [3, 4]). Thus, strong authentication is enabled in mobile scenarios when applying the mobile device itself for service usage. However, in many cases, a service might not be consumable on the mobile device. This might be due to the requirement of special hardware as in the case of ATMs or high performance requirements the mobile device cannot fulfill. Furthermore, many examples like multimedia services are imaginable, where a user is not willing to use the mobile device for service consumption just for comfort reasons.

Subsequently, we show how the authentication can be physically decoupled from the usage of services. Namely, the authentication is performed using a mobile and user bound device which we call *authentication device*, while the usage of services is done conveniently on a computer or in general a second device which we call *application device*. The strict separation between authentication and application device provides two main advantages. First, it enables a secure and flexible user authentication independent from the capabilities of the system used for service consumption. Particularly, it enables the ubiquitous applicability of smart card based authentication which is our main focus. Yet in general, arbitrary protocols and authentication mechanisms realizing different security requirements and levels are applicable. Thus, we also provide hints how that can be realized when describing the protocols. Second, it protects the user's credentials from being eavesdropped by malicious software on the application device. Thus, for the protection of the credentials only the security of the authentication device is required, which is much easier to realize, as it is a single device and always controlled by the user.

[1] http://www.h-online.com/security/news/item/Passwords-The-only-constant-in-life-1030529.html

1.1 Outline

In the following, we first present related work give some relevant background on the German identity card. In Section 3 we describe the decoupled authentication in detail. We present general assumptions and detail two variants of our approach, namely the push and the pull procedure. Besides that we explain how they can be implemented in practice based on the German identity card. In Section 4 we compare the push and pull procedure and show the main differences besides the respective advantages and disadvantages. We end with a conclusion and summarize the security gains compared to other authentication methods and identify limitations.

2 Background and related work

First we present related work concerning authentication methods involving a second device to enable comparison with our approach. Then, as we make use of the German identity card as an example to implement the decoupled authentication method, we provide the relevant background to establish a basic understanding about its functionality and application.

2.1 Related work

Some services like Google[2], Dropbox[3], and Facebook[4] already implement a 2-factor authentication to protect the user accounts from hijacking. For instance, to set up Google's *2-step verification* the user must add his mobile phone number to his existing account and install the application *Google Authenticator* on his mobile device. A verification code is sent to the mobile device and must be entered during the setup phase. During any further login the user must enter an additional one-time password, which is generated by the application on the mobile device. Another example is the well established mobile TAN (mTAN) method as used for online banking, where the bank sends an SMS containing a one-time password (the TAN) to the user's mobile device. The TAN must then be entered into the application system so confirm a certain transaction.

All the approaches have in common, that user name and password for login are still entered on the computer. That can lead to security risks. User name and password can be eavesdropped on by malware and can be abused by an adversary. Most severe, if the same passwords are used for multiple accounts – which indeed is often the case – and these other accounts are not secured by an additional authentication step. Furthermore, a possibly complex setup is required repeatedly for each service provider to e.g. register the mobile phone number and to reconfirm the registered devices.

Our proposal provides a much more universal applicability, for instance smart card and certificate based authentication. Thereby, no prior registration of a mobile device at different service providers is required. For example, when using the German identity card, decoupled authentication can even be used for registration. Other authentication methods might require a single registration at an identity provider. During each login user entries like user name and password are protected from eavesdropping. Further

[2] https://support.google.com/accounts/bin/topic.py?hl=en&topic=28786

[3] https://www.dropbox.com/help/363/en

[4] https://www.facebook.com/note.php?note_id=10150172618258920

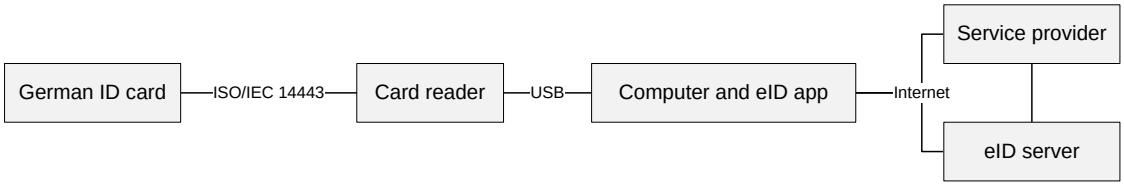

Figure 1: eID-infrastructure

protection as the mTAN procedure in online banking can be kept for the confirmation of transactions to provide security against the manipulation during an active session. However, it is also imaginable that the decoupled authentication is used repeatedly during a session to reconfirm certain steps as transactions or password changes.

2.2 German identity card

The German identity card is equipped with a contactless chip according to ISO/IEC 14443 [1] and compliant to ECC [5] and ICAO Doc 9303 [15]. The card holder's personal data is printed on the card surface and stored on the integrated chip. Common biometric features like the ePassport [15] application and in particular functions for electronic authentication (eID functionality) and the generation of qualified electronic signatures (eSign functionality) are supported. Access to these functionalities is respectively secured with a six digit *Personal Identification Number* (PIN) [8].

Therewith, the German identity card allows its card holder to electronically prove his identity in eBusiness and eGovernment applications. In contrast to a common one-factor authentication by user name and password, the eID functionality enables a two-factor authentication based on the possession of the card and the knowledge of the PIN. Furthermore, the authentication is done mutually. This enables a high level of trust between the service provider who presents a certificate issued by the governmental administration and the user who is equipped with a sovereign ID document.

The authentication process is performed by dedicated eID servers [7]. An eID server manages the certificates issued for service providers, performs the security protocols, and retrieves the personal data stored on the card. The service providers receive the data from the eID server and perform a local authentication process based on their environment. The eID server can be operated by the service providers themselves or by third parties. The corresponding infrastructure is shown in Figure 1. The card holder needs a computer with an eID application installed on it and a card reader to use the eID functionality of the card. The eID application, for instance the Open eCard App [14], implements the required communication and cryptographic protocols as well as the user interaction.

The access control of the ID card is realized by a set of security protocols specified in TR-03110 [10]. These protocols perform a user authentication based on a PIN and a mutual authentication between the ID card and the eID server. The access rights of the eID server are proven by certificates issued by the *German Federal Administration Office*. A security proof of the protocols is given in [2, 6] concerning the security against active adversaries having access to the communication channels between the involved components.

3 Decoupled authentication

In the following we describe our method to decouple the authentication from the system on which a specific service is used. Basically, we can separate the whole process into three phases:

1. Service request
2. (Mutual) authentication
3. Service usage

The proposed technique assigns the phases to two independent devices: The first and the third phase is processed on one device, namely the *application device* through which the user requests the service and uses it afterward. The second phase is processed on the second, namely the *authentication device* through which the user performs the authentication and thereby proves his access rights to the service. Note that both devices are physically separated. Thus, the application device has no access to the user input like user name, passwords or PINs. That means there is a strict separation between authentication and service usage. The only interface between the two systems is the user himself.

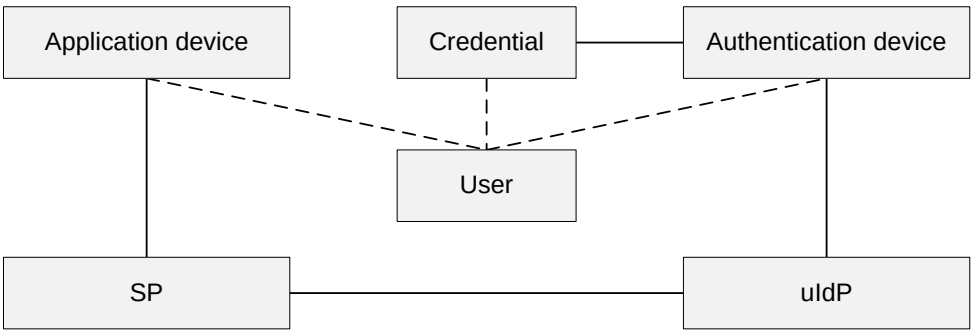

Figure 2: Decoupled Authentication: Participants and Devices

As shown in Figure 2, the system consists of several participants. The services are provided by service providers (SP) and the authentication of the users is performed by universal identity providers (uIdP). The uIdPs may provide different authentication mechanisms to achieve various security and authenticity levels (cf. SkIDentity project[5]). The user has access to an application device to use the service and an authentication device to prove his identity based on some credential.

In an exemplary Web service scenario, the procedure would be as follows: The user browses the web site of the desired service on his application device. To obtain access, the SP requests the authentication of the user. Using the authentication device, the user performs the authentication via the uIdP. After the successful authentication, the uIdP confirms it to the SP and the user is granted access to the service on the application device.

The key challenge is to maintain the logical link between authentication and the usage of a service while physically decoupling them such that the service request comes from the application device and the authentication is performed through the authentication device. We propose two different variants to

[5] http://www.skidentity.com

deal with that challenge: the *push procedure*, where the authentication is triggered externally, and the *pull procedure*, where the authentication process is triggered by the user. We describe the two methods in Sections 3.2 and 3.3. Beforehand, we describe and justify our basic assumptions.

3.1 General assumptions

We assume, that the user possesses a credential, for instance a smart card, which supports a user authentication to verify that the user is the legitimate card holder and a mutual authentication to the uIdP. The mutual authentication allows the smart card to verify that the uIdP is authorized to access the card holder's personal data and the uIdP is able to verify the genuineness of the card.

We assume that the authentication device is trustworthy and not compromised by malicious software. If we are using a smart phone as an authentication device, it can still to be considered as more secure than common computers. In case of business phones, the installation of foreign software is often forbidden or only allowed to a limited extent. By the application of *Mobile Trusted Modules* (MTM) [17] a compromise may be prevented. Indeed, the application of MTMs in mobile devices is limited so far, however compared to standard computer systems a much higher user acceptance is to be expected. Thus, a faster and easier market breakthrough is facilitated.

3.2 Push procedure

First we describe the *push procedure*. We give a brief overview and continue with a detailed description and an explicit example for the realization with the German identity card and an NFC-enabled device.

3.2.1 Overview

In case of the push procedure the authentication process is externally activated by the uIdP. When the user requests a service with an application device, he provides an additional identifier (ID). This ID may be entered every time or locally stored on the device and transmitted automatically. The ID might be an email address, a phone number, a user name or any kind of an URI, which in particular might be public information. The uIdP uses this ID to determine the authentication device and to trigger the authentication[6]. By receiving a message, for instance an SMS or email, from the uIdP the authentication device starts a local eID application to perform the user authentication. After a successful authentication, the SP grants access to the user on the application device.

3.2.2 Application flow

Figure 3 shows the application flow of the push procedure. Note that we provide the application flow considering the usage of a rather powerful credential like a smart card or another token with the computational power to perform certain cryptographic operations and protocols. In the following, we describe the steps of the procedure:

1. The user tries to access a service, for instance, by browsing a web site or starting a program which initiates the connection to the service.

[6] Note, that either the user must be registered at the uIdP, or the given ID must be a global identifier like a mobile phone number.

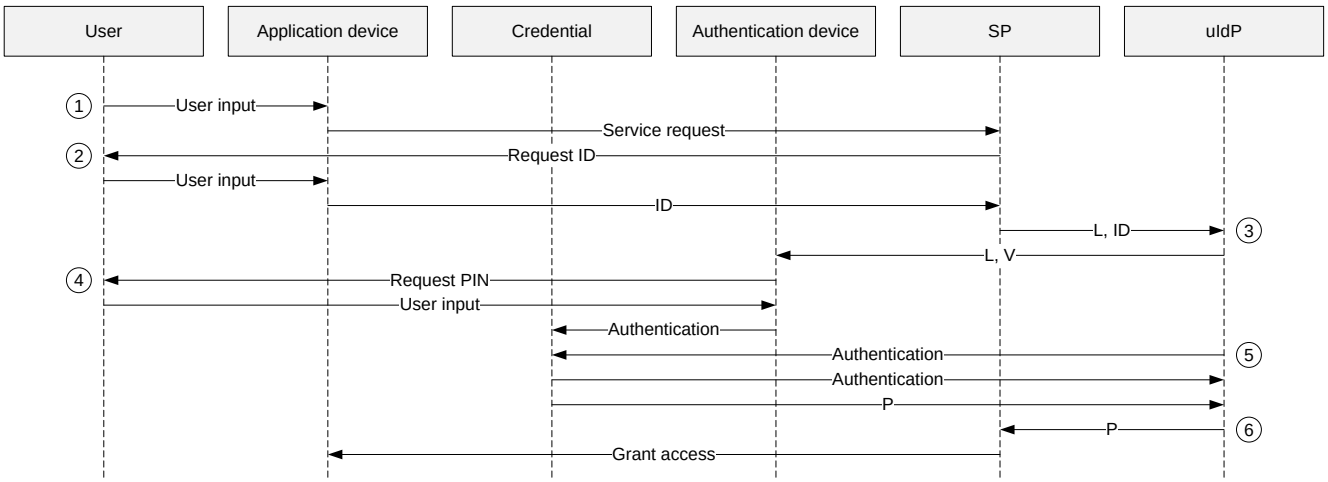

Figure 3: Flow chart of the push procedure

2. The SP requests the ID from the user to determine his authentication device.

3. The SP specifies a set of required personal data L to identify the user. That can be a list of personal characteristics like name, surname, address, age, place of birth, nationality, gender or other electronic identities assigned to the user. The ID and L are transmitted to the uIdP. Using the ID, the uIdP determines the authentication device and sends a message containing L and V to trigger the authentication process. The data V contains information about the uIdP and the SP like certificates, descriptions, and terms of usage. That information is shown to the user to allow him to verify which providers are about to receive his personal data.

4. Afterward, the user applies his credential to the authentication device and proves the legitimate possession of the credential. For that purpose password-based authentication protocols are suitable, for instance, Password Authenticated Connection Establishment (PACE) [10] or Simple Password Exponential Key Exchange (SPEKE) [16].

5. Now a mutual authentication between the credential and the uIdP is performed. For example, the authentication can be based on certificates and a public key infrastructure or on shared secret information. The authentication device herein acts as an intermediary. After all participants have been successfully authenticated, the credential transmits the required personal data P specified by L to the uIdP.

6. The uIdP forwards the data P to the SP. The SP checks the obtained data and grants access to the service to be used through the application device.

As mentioned before, also weaker authentication mechanisms as e.g. based on user name and password or certificates are possible. In that case mainly Steps 4-5 are different from the above. First, the user is not required to prove the possession of a credential but shows the knowledge of user name and password or of the private-key belonging to the public-key specified in the certificate by entering it into the authentication device. Thereby, the authentication device performs the cryptographic protocols and

necessary computations as e.g signature generation. Concerning Step 5, instead of the mutual authentication a single sided authentication of the user can be realized. Furthermore, the personal data needs not be read out from a credential 8like smart card or certificate) but might be entered by the user and transmitted to the uIdP by the authentication device. Note that all these potentially security sensitive user entries are done on the authentication device and are not visible for the application device.

3.2.3 Application example

In this section we describe how the general procedure from above can be realized with the infrastructure of the German identity card. An NFC-enabled, mobile device is used as the authentication device and card reader, as described in [13]. The service is used through an arbitrary (potentially insecure) computer, which is independent from the mobile device. Consider that the user is also registered at the uIdP with his mobile phone number.

- The uIdP acts as an eID server [9] and implements the *eCard-API-Framework* according to TR-03112 [11].
- The message sent from the uIdP to the authentication device in Step 3 (cf. Figure 3) must contain a TCTOKEN according to TR-03112 [12]. The TCToken contains amongst other things the URL of the eID server, a session identifier, negotiation information, and a pre-shared key to establish a TLS-based connection between the eID application and the eID server.
- The user authentication in Step 4 is performed using the PACE protocol. PACE provides a password-based authentication using a PIN and establishes a secure channel to prevent the communication from being eavesdropped.
- The mutual authentication in Step 5 is performed using the Extended Access Control (EAC) [10] protocol. The eID server proves its authorization using Card Verifiable Certificates issued by the *German Federal Administration Office*. The card shows its authenticity by proving the possession of a private key.

3.3 Pull procedure

The ID used in the push procedure is typically constant and may represent some security-sensitive information, for instance, a mobile phone number.

In this section we describe the *pull procedure*, which provides the possibility of pseudonymous authentication for anonymous usage of services, because of using random IDs. We give a brief overview of the procedure and continue with a detailed description.

3.3.1 Overview

Using the pull procedure, the authentication procedure is initialized by the user instead of the uIdP. That means, the authentication device contacts the uIdP and requests the user authentication. To achieve a link between authentication and service request, the SP has to display an ID on the application device,

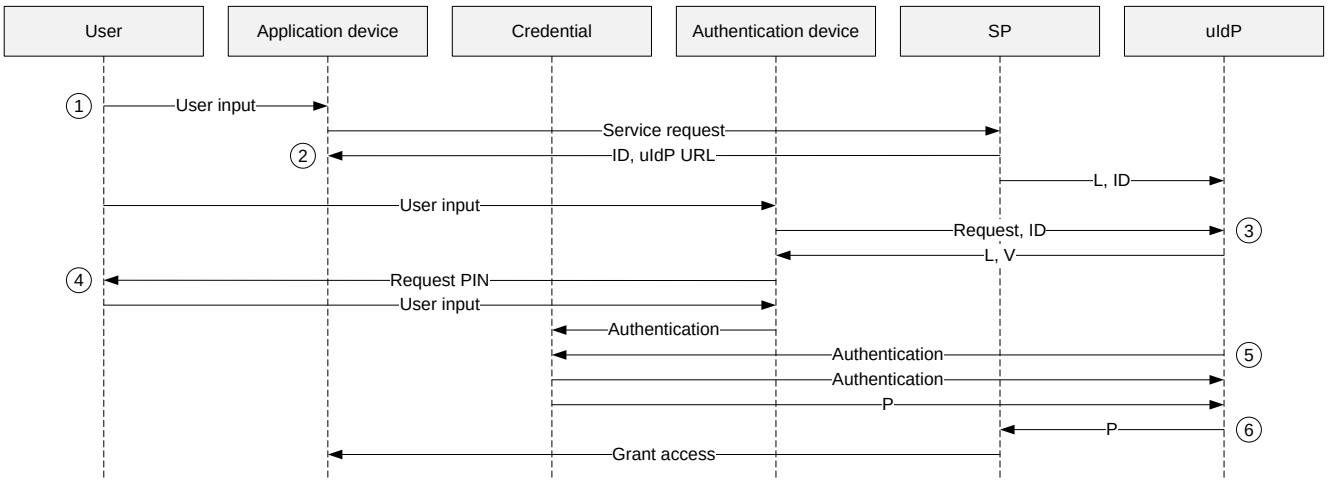

Figure 4: Flow chart pull procedure

which in turn is entered into the authentication device. Furthermore, the SP must provide the URL of the uIdP that needs to be contacted for the authentication purpose.

The ID and the URL can e.g. be character strings, a bar-code or QR-code which might be captured using the camera of the mobile (authentication) device. Based on the URL, the eID application can establish a connection to the uIdP and perform the user authentication for the specified ID.

3.3.2 Application flow

The pull procedure is depicted in Figure 4. Again we focus on the usage of a powerful credential like a smart card. The steps are the following:

1. The user requests the service.
2. The SP generates a random ID and sends it to the user. The ID is also announced to the uIdP to link the authentication process to the service request. Additionally the SP specify a set of required personal data L.
3. The user enters the ID and the uIdP's URL on the authentication device. The device connects to the uIdP and requests the user authentication for the ID. The uIdP delivers L and some additional uIdP and SP related verification data V.
4. The user authentication is performed.
5. The mutual authentication is performed and the personal data P is transmitted to the uIdP.
6. The uIdP forwards the data P to the SP and the user is granted access to the service.

The Steps 4-5 are the same as in the push procedure, and again the kind of user authentication as well as if it is performed mutually might depend on the security requirements of a specific application (cf. Section 3.2.2).

3.3.3 Application example

The process of using the German identity card and an NFC-enabled device is almost the same as described in the push procedure. We just need to adapt the activation of the eID application running on the authentication device. The ID and the URL of the uIdP, both chosen by the SP, must be entered into the eID application. Under the URL, which directs to the uIdP, the eID application should retrieve a TCTOKEN (cf. [12, Kapitel 3.2]).

4 Comparison of approaches

In this section we describe the main differences of the push and the pull approach and show the respective advantages and disadvantages. The comparison is summarized in Table 1.

4.1 Initialization

One of the main differences concerns the initialization of the authentication procedure. In the push procedure the authentication process starts automatically. Thus, the user is not required to perform additional steps to initialize the authentication on his authentication device, which can be seen as a clear usability advantage compared to the pull procedure, where the user is required to manually initialize the authentication.

4.2 ID and registration requirements

The push method requires that the user, or rather his authentication device can be unambiguously identified with the ID. This can either be achieved by prior registering with a specific uIdP or by using a global identifier as e.g. the user's mobile phone number[7]. In contrast, the pull method allows, that for each authentication a new and random ID can be used to establish the link between authentication and service usage. Thereby, an registration at the uIdP is not required, as the user contacts the uIdP.

4.3 Authentication device

Due to the registration when using the push method, authentication can be bound to a specific device, which provides the security gain, that the authentication is only possible when being able to access that registered device. The pull method on the other hand provides much more flexibility as the user can apply any device for authentication that provides the necessary functionality without any prior registration at a uIdP.

4.4 Choice of the uIdP

In the push method, the flexibility concerning the use of different uIdPs is limited due to the registration requirements. Either the user must stick to one single uIdP or must register with several uIdPs. The former has the advantage that the user may choose one which he considers trustworthy, but has the disadvantage, that SPs must interact with different uIdPs for different users. In contrast, the latter

[7] Which in turn requires the registration of the global identifier at some point.

	Push	Pull
Automatized initialization	yes	no
Prior registration required	yes	no
Arbitrary uIdPs possible	no	yes
Determined authentication device	yes	no
Unique user ID	yes	no
Anonymous service usage	no	yes

Table 1: Comparison of Push & Pull Procedure

requires the user to perform a possibly time-consuming registration with many uIdPs and the user is required to trust in many different uIdPs.

Using global identifiers can relax these drawbacks, but comes with strong privacy concerns we discuss in the next paragraph. In comparison, when considering the pull method, the choice of the uIdP can be left to the SPs providing flexibility on the SP's side. As the SP tells the user with which uIdP he needs to interact, the only requirement is to use standardized interfaces for the communication. However, the user is required to trust in any uIdP a SP might choose to cooperate with.

4.5 Privacy

With the above, it can be seen that the push method leads to several privacy concerns and its applicability is limited for the scenario of anonymous service usage (as e.g. in the case of the pseudonym functionality of the German identity card). In the push method, the user reuses one and the same ID for many different services and he must enter his ID on a (potentially insecure) application device. Thus, the unique ID could be leaked from the authentication device and might enable the eavesdropper to identify the user. Furthermore, SPs are collaboratively able to link the service usage of different services to a single person. Even worse, in the case of using a global identifier the person might completely be identified. Additionally, the uIdP, who executes the authentication for many different services, can trace and profile users – and as mentioned above, using different uIdPs to counteract tracing and profiling is difficult to realize.

The pull method removes the privacy threats of the push method. For each session a new and random ID can be used to establish the link between authentication and service usage, which removes the linkability of different service requests to one person. Besides that, many different uIdPs can conveniently be used to counteract user profiling.

5 Conclusion

The main security gain of the push and pull procedures for authentication is the separation of service consumption and authentication. Security-sensitive data like passwords is exclusively entered on the user bound authentication device. The security of that system hereby is under the control of the user. The user does not have to rely on the security measures of foreign or public devices. By the application of a mobile device as authentication system, different strong authentication methods are universally and ubiquitously available for application. For example, the security strengths of smart card based authentication can be used independent from the availability of the required hardware and software on the application device. Besides that, the uIdP might implement many different authentication methods, while a SP can integrate any of these into his service by just implementing a single interface to the uIdP.

Malicious software installed on the application device can only get knowledge of the ID during the login process. In case of using random IDs as in the pull procedure, they contain no security-sensitive information. Considering constant IDs, the severity of the security threats depends on the type of ID and mainly concerns privacy issues. Hereby, a global identifier like a phone number, which directly leads to a person's identity is more threatening than an ID like *my_pseudonym@my_uidp.com*. The latter in many cases provides sufficient anonymity. Of course, constant IDs still involve the risk that an adversary might try to abuse them. But, if the user exclusively applies the respective ID in the context of the push method he is at least informed about such trials and can react appropriately.

However, our authentication method cannot protect the service usage itself. Malware that spies on and manipulates the user interaction or man-in-the-browser attacks on the application device, where the adversary might take over the current session are not prevented. But still, even in the case of such a powerful adversary, the authentication credentials are secure and the attack is restricted to the current session. Furthermore, decoupled authentication could be repeatedly used during a session to authorize sensitive transactions. The push procedure for example would allow to transfer certain transaction data within the initialization message to the authentication device comparable to the mTAN approach. In the pull procedure, the transaction data can become a part of the ID specified by the SP.

Considering the security of the application device, we assume it not to be compromised, which is manageable compared to requiring the security of the application device. However it is worth to mention that when using smart card based authentication with our approach, then the second security layer of card possession still holds when the authentication device is compromised. Furthermore, guaranteeing security of the application device comprises an interesting research question for future work. For instance, when using MTM-enabled smart phones a promising approach is to combine the decoupled authentication with an additional *Remote Attestation* [17, Section 10.3] performed by the uIdP to identify unauthorized changes of the device.

References

[1] ISO/IEC: Identification cards - Contactless integrated circuit(s) cards - Proximity cards - Part 1-4. International Standard, ISO/IEC 14443, 2008 - 2011.

[2] Jens Bender, Marc Fischlin, and Dennis Kügler. Security Analysis of the PACE Key-Agreement Protocol. In *Information Security Conference (ISC) 2009*, volume 5735 of *Lecture Notes in Computer Science*, pages 33–48. Springer, Sep 2009.

[3] Johannes Braun, Moritz Horsch, Alex Wiesmaier, and Detlef Hühnlein. Mobile Authentication and Signature (Mobile Authentisierung und Signatur). In *D-A-CH Security 2011*, September 2011.

[4] Johannes Braun, Moritz Horsch, and Alexander Wiesmaier. iPIN and mTAN for Secure eID Applications. In Mark Ryan, Ben Smyth, and Guilin Wang, editors, *Information Security Practice and Experience*, volume 7232 of *Lecture Notes in Computer Science*, pages 259–276. Springer Berlin / Heidelberg, 2012.

[5] Comité Européen de Normalisation (CEN). Identification card systems - European Citizen Card - Part 1-4, CEN/TS 15480. Technical Specification, 2008.

[6] Özgür Dagdelen and Marc Fischlin. Security Analysis of the Extended Access Control Protocol for Machine Readable Travel Documents. In *13th Information Security Conference*, Lecture Notes in Computer Science. Springer, Okt 2010.

[7] Federal Office for Information Security. Technical Guideline eID server (Technische Richtlinie eID-Server). Technical Guideline BSI-TR-03130, Version 1.4.1, 2010. https://www.bsi.bund.de/ContentBSI/Publikationen/TechnischeRichtlinien/tr03130/tr-03130.html.

[8] Federal Office for Information Security. Architecture of the German ID card and residence permit (Architektur elektronischer Personalausweis und elektronischer Aufenthaltstitel). Technical Guideline BSI-TR-03127, Version 1.14, 2011. https://www.bsi.bund.de/ContentBSI/Publikationen/TechnischeRichtlinien/tr03127/tr-03127.html.

[9] Federal Office for Information Security. Requirements for card readers supporting the German ID card (Anforderungen an Chipkartenleser mit nPA Unterstützung). Technical Guideline BSI-TR-03119, Version 1.2, 2011. https://www.bsi.bund.de/ContentBSI/Publikationen/TechnischeRichtlinien/tr03119/index_htm.html.

[10] Federal Office for Information Security. Advanced Security Mechanism for Machine Readable Travel Documents - Extended Access Control (EAC), Password Authenticated Connection Establishment (PACE), and Restricted Identification (RI). Technical Guideline BSI-TR-03110, Version 2.10, 2012. https://www.bsi.bund.de/ContentBSI/Publikationen/TechnischeRichtlinien/tr03110/index_htm.html.

[11] Federal Office for Information Security. eCard-API-Framework. Technical Guideline BSI-TR-03112, Version 1.1.2, Part 1-7, 2012. https://www.bsi.bund.de/ContentBSI/Publikationen/TechnischeRichtlinien/tr03112/index_htm.html.

[12] Federal Office for Information Security. eCard-API-Framework – Protocols. Technical Guideline BSI-TR-03112-7, Version 1.1.2, 2012. https://www.bsi.bund.de/ContentBSI/Publikationen/TechnischeRichtlinien/tr03112/index_htm.html.

[13] Moritz Horsch. Mobile Authentication using the German identity card. Master Thesis, TU Darmstadt, Jul 2011. http://www-old.cdc.informatik.tu-darmstadt.de/reports/reports/Moritz_Horsch_MONA.master.pdf.

[14] Detlef Hühnlein, Dirk Petrautzki, Johannes Schmölz, Tobias Wich, Moritz Horsch, Thomas Wieland, Jan Eichholz, Alexander Wiesmaier, Johannes Braun, Florian Feldmann, Simon Potzernheim, Jörg Schwenk, Christian Kahlo, Andreas Kühne, and Heiko Veit. On the design and implementation of the open ecard app. In *Sicherheit 2012*, March 2012.

[15] International Civil Aviation Organization (ICAO). Machine Readable Travel Documents. ICAO Doc 9303, Part 1-3, 2006 - 2008. http://www2.icao.int/en/MRTD/Pages/Doc9393.aspx.

[16] David P. Jablon. Strong password-only authenticated key exchange. *SIGCOMM Comput. Commun. Rev.*, 26(5):5–26, October 1996.

[17] TCG. TCG Mobile Trusted Module Specification. Version 1.0, 2012. http://www.trustedcomputinggroup.org/resources/mobile_phone_work_group_mobile_trusted_module_specification.