An Identity Provider using a Secure Element of a Phone: Smart Card Based OpenID

Andreas Leicher[1], Ran Zhou[2], Andreas U. Schmidt[1], Carsten Rust[2], Yogendra Shah[3], and Christoph Sorge[4]

[1] Novalyst IT AG,
Robert-Bosch-Str. 38, 61184 Karben, Germany
{andreas.leicher,andreas.schmidt}@novalyst.de
[2] Morpho Cards GmbH
Riemekestrasse 160, 33106 Paderborn, Germany
{ran.zhou,carsten.rust}@morpho.com
[3] InterDigital Communications, LLC.,
781 Third Avenue, King of Prussia, Pennsylvania, 19406 USA
yogendra.shah@interdigital.com
[4] University of Paderborn
Warburger Strasse 100, 33098 Paderborn, Germany
christoph.sorge@uni-paderborn.de

Abstract. Personal mobile devices with internet connectivity such as smartphones and tablet computers are widespread throughout all levels of society. The most popular device, the smart phone, has become an ubiquitous personal mobile computing platform with personal applications are provided as services to consumers over the internet. This shift to several centrally provided services, with de-centralized access via mobile end-point devices requires solutions for secure Identity and Access Management (IAM). The OpenID protocol enables Single Sign-On (SSO) and Identity Management (IdM) for internet based services. Mobile Network Operators (MNOs) are interested in providing SSO and IdM as a service for their subscribers. In this paper, we discuss and analyze the concept of Smart OpenID, an enhancement to OpenID which moves part of the OpenID authentication server functionality to the smart card of the user's mobile device. This seamless, OpenID-conformant protocol allows for scaling security properties, and generally improves the security of OpenID by avoiding the need to send user credentials over the Internet and thus avoids phishing attacks. We also describe our implementation of the Smart OpenID protocol based on an Android phone, which interacts with OpenID-enabled web services.

Keywords: OpenID, Identity Management, Single Sign-On, Authentication, Smart Cards

1 Introduction

With a growing number of internet services, used both for business and for private purposes, users face the problem of managing their identities as well as their authentication data (in most cases, user names and passwords). They can use the same identity (and associated authentication data) for all services, or try to deal with multiple identities. While the first solution is insecure and a privacy risk, the second one is impractical—creating and remembering many strong passwords is too large a burden on users [?,?]. In addition, attributes belonging to an identity must be kept up to date in all services, and it is difficult for a user to get an overview of where his attributes are stored. Therefore, users need support for managing identities and authentication data. Local management of passwords leaves many open issues. A more comprehensive identity management (IdM) solution, also taking users' privacy into account, is needed. For example, Liberty Alliance [?], CardSpace [?,?], Higgins [?] have been proposed. However, they have not seen a high adoption rate by end users and services in the consumer market. The IdM protocol OpenID has been more successful, but is prone to phishing attacks if combined with password authentication. In this paper, we discuss a novel approach combining the security of smart cards, the authentication systems of mobile network operators and the OpenID protocol, to address all above-mentioned issues. **Smart OpenID** is the present proposal to locate essential, cryptographic functionality of the OpenID Identity Provider (OP) on a secure element, co-operating with an OP protocol endpoint on the user device.

Smart cards are portable and tamper-resistant computing devices which are able to securely store and process information, and which are often used for secure data storage and authentication. The biggest market for smart cards is the mobile communication industry where, in third generation (3G) networks, the Universal Integrated Circuit Card (UICC) is the smart card used to authenticate a subscriber to the mobile network. The UICC is able to perform authentication algorithms and provides cryptographic functions for encryption and decryption [?]. Using technology like Java Card, it is possible to deploy Java applets onto the UICC. Communication between a host application and an applet on the smart card uses Application Protocol Data Units (APDUs) in a command/response protocol [?]. Additional APIs, e.g. OpenMobileAPI [?], enable applications running on a mobile phone to access UICC functions.

Based on these elements, we describe a prototype implementation of Smart OpenID on an Android phone using a common smart card. In a previous publication [?], some of the authors have presented the protocol itself and the user interface. We now extend this by describing the implementation architecture in detail and discussing security issues. The next section provides a brief introduction to OpenID 2.0 and Smart OpenID, which can already be found in more detail in [?], and is added here for the readers' convenience. Section ?? puts our work into the context of related work. Section ?? provides the details of the technical architecture of the Smart OpenID prototype, in particular regarding the interplay between smart card applet and Android phone app using the Open Mobile API. Section ?? states the most important technical implementation details and performance measurements carried out. Section ?? gives an informal security analysis as well as a semi-formal one based on attack trees. The paper is concluded in Section ??.

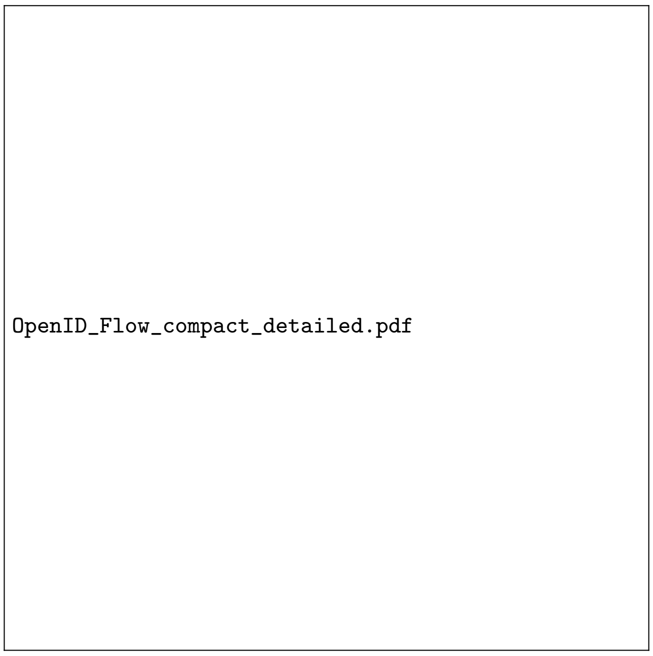

Fig. 1: Overview of the OpenID Protocol Flow

2 Smart OpenID

OpenID 2.0 [?] allows users to log in to different services with a single identifier, where the authentication itself is delegated to the OpenID provider (OP), a generic Identity Provider (IdP). Using OpenID alleviates the need to create separate accounts for services, mitigating some issues with password based authentications, particularly phishing, reuse of passwords on multiple service sites, and insecure means to remember passwords [?,?]. OpenID identifiers are URLs provided by an OP. The OP authenticates the user and validates the user's credentials on behalf of the services, called Relying Parties (RP). Figure **??** gives an overview of the OpenID protocol, which relies solely on HTTP transport, in particularly using the mechanics of HTTP redirects. More detail is found in [?].

To enable an efficient use of the existing and deployed security infrastructure of MNOs, and to provide seamless access to service providers, we introduce a new entity for our smart card based OpenID protocol, called local OP. Figure **??** provides an overview of the Smart OpenID protocol for the following description.

The local OP acts as a delegate of the network OP to authenticate the end user and issue the OpenID assertions to the RP via the user's browser. Using the capabilities provided by the UICC, the local OP can securely store secrets and perform crypto operations, e.g. creating signatures. The local OP can be implemented as a combination of a user level application which provides an interface towards the browser, and an applet on the UICC which handles all cryptographic operations and storage of keys. From the viewpoint of RPs the protocol flow of Smart OpenID just appears as a standard OpenID protocol exchange.

Due to restrictions by the mobile network, the local OP cannot be reached by RPs for the discovery and association steps of the OpenID protocol (steps 2, 3 in Figure**??**), i.e., communication to the local OP is restricted to the device only. Therefore, we introduce the OP Support Function (OPSF), operated by a network entity such as the MNO and reachable via public Internet. Trust association between OPSF and local OP is established by a shared secret S, which is stored on the smart card. Provisioning of the shared secret can be done in various ways, with a practical preference for over-the-air (OTA [**?**,**?**]) methods, or even by binding to a mobile network authentication run, as proposed in [**?**]. The specifics are design choices which are out of scope of this paper.

Step numbers in the following description refer to Figure **??**. When the user logs on to a RP using his identifier (step 1), the RP performs discovery of the OPSF (step 2) and establishes an association based on an *association secret* key S_a, which in turn is derived by OPSF from S and a random nonce `asc_hdl`, called *association handle* to ensure freshness, i.e., uniqueness of the authentication process. The OPSF sends the association response [**?**, sec. 8.2] back to RP, in which S_a is encrypted using a Diffie-Hellman secret (step 3). Using an HTTP redirect (step 4), the user's browser is sent to the OPSF, which then triggers the execution of the local OP application on the user's device (step 5). This is achieved by registering a custom URL scheme `soid.scheme://` on the device which triggers the call to the local OP (step 6). By this, modifications of the browser are avoided, as the scheme registration happens with the installation of the local OP app in the Android operating system. The local OP now authenticates the user (step 7), requesting e.g. a PIN code, effectively introducing local, two-factor authentication, in the simplest case by the PIN request being validated on the smart card. Strong user authentication using, e.g., biometrics could be used without principle changes.

On authentication success, the local OP uses S together with `asc_hdl` received in the redirected request to derive S_a. The local Op then creates an assertion message according to OpenID specifications and signs it using S_a. The browser is then redirected (step 8) by the local OP to the RP with an HTTP header containing the signed assertion (step 9). The RP can verify the assertion using S_a and allow the user to access his service (step 10).

It is possible for a single local OP to provide multiple identifiers for one user, e.g., for personal or business use. Since the security relevant operations of the local OP are in the smart card and cannot be modified by the user, the entity controlling the OPSF and the key provisioning procedure, for instance an MNO, has control over the identifiers. The local OP will only create assertions for the identifiers which are enabled by the MNO. The user could, for example, create or buy identifiers to be installed in the local OP if he needs separate identifiers for different purposes. In [**?**] we have described how to enable advanced use cases

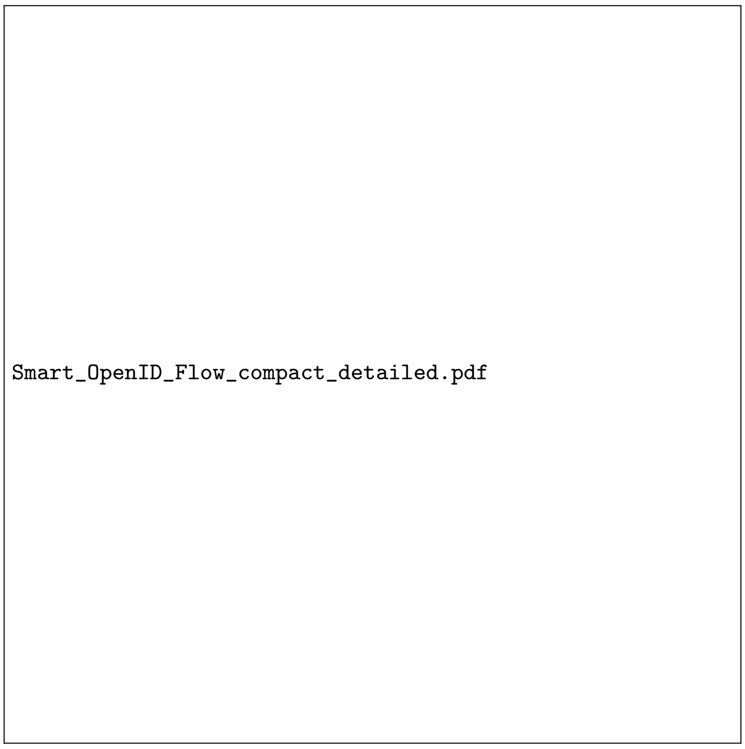

Fig. 2: Protocol Flow for the Smart OpenID protocol

using multiple, context-dependent, identifiers for a user, employing the identifier selection [?, sec. 10] method, e.g. to enable attribute-based access to services.

3 Prior and Related Work

As OpenID receives a lot of attention from industry and research, most research work is centered around the integration of different authentication mechanisms into the OpenID protocol.

A study on the security aspects of OpenID [?] highlights the danger of phishing. By tricking an user into giving away their OpenID authentication credentials, e.g. by setting up a fake OP, an attacker can get access to all OpenID enabled services in the name of the legitimate user. In order to do this, the attacker does not have to attack the OP directly but can set up a malicious RP which then redirects the user to the fake OP. As OPs act as IdPs for multiple users an attacker can use this technique to easily collect credentials for a large amount of users. Another attack is Cross-Site-Request-Forgery (CSRF), which silently logs the user on to a service of the attacker's choice once the user has logged in into another RP and then allows the attacker to perform actions in the user's name. The CSRF attack relies on the fact that the OP and not the RP decides on the login security policy and that the OP does not show the login process to the user if a CSRF attack happens.

OpenID does not specify the authentication method between OP and user device, thus different proposals have emerged for the integration of strong authentication methods with OpenID. In [?,?] a smart card based solution for OpenID authentication is presented, where a smart card stores a TLS certificate which may be used for authentication with the OP. TPM based authentication for OpenID is described in [?].

MNOs show increasing interest in leveraging their subscriber base and customer data in novel business models [?], which are tightly integrated with mobile network authentication. By this, MNOs may become what has been called *IdM Enablers* [?,?,?]. One way to expose cellular network authentication to general Internet-based IdM contexts is via the Generic Bootstrapping Architecture (GBA) [?]. 3GPP has investigated the option for interoperability of GBA and OpenID [?,?,?], providing GBA based authentication at the OP. A similar approach for integration of EAP-SIM with OpenID has been taken by [?].

Liberty Alliance (LA) introduced a new entity in [?], called advanced client. At its core is the trusted module (TM), which is an extension of the network IdP. Local applications and service providers can delegate user authentication to the TM instead of the network IdP. The TM is considered to run in a trusted and secure environment of the device. In [?] the deployment of a TM onto a UICC using 3GPP Over The Air (OTA) management operations [?,?] is discussed.

Smart OpenID differs from the previous proposals in introducing a local authentication proxy for OpenID which takes part of the IdP functions into the device, and in particular onto a secure element. This approach is conceptually close to the LA advanced client. Based on the concept of Smart OpenID [?], we have described a binding of Smart OpenID authentication to GBA in [?].

4 Architecture

The local OP, introduced as a delegate of the network OP, consists of a user level application on the mobile phone and an applet on the UICC. This corresponds to the dual application architecture, promoted by the GSMA (GSM Association) to support secured NFC (Near Field Communication) services [?]. The smartphone (as the physical carrier of the local OP) is usually divided into two isolated subsystems connected through a well-defined interface [?]. The subsystems are the so-called application processor running the operating system and applications, and the baseband (radio) implementing the interface to the mobile network. Due to this separation, applications are normally not able to access the UICC, which is connected to the baseband processor, since no proper APIs are provided. The OpenMobileAPI, specified by SIMAlliance [?], is introduced to fill the gap and leverage the UICC for third-party applications. Therefore, the local OP can be implemented with a dual application architecture using OpenMobileAPI as secure element access API.

`graphics/DualAA.pdf`

Fig. 3: Dual Application Architecture

4.1 Dual Application Architecture

In order to utilize the UICC to improve the security level of applications and services on the smartphone, the dual application architecture as depicted in Figure **??** should be applied. The user interface is provided by the mobile application as the consumer facing component, while the actual logic resides in the secure element to guarantee the security of the whole application. The communication channel between the components can be provided by the OpenMobileAPI.

4.2 Access Control Mechanism

In the dual application architecture, the access to the secure element should be controlled carefully to prevent threats from malicious applications. A signature-based access control mechanism is proposed by GSMA in [**?**]. The secure element access API should integrate an access control module implemented based on this mechanism, consisting of three sub-modules: (1) Access control filtering in the secure element access API, (2) Access control rules database & engine in the secure element access API, (3) Access control data in the secure element.

The access control module listens to the access request, which contains the signature of the application's certificate and the identifier (AID) of the target applet, issued by the mobile application. Upon receipt of the request, the engine looks up the applicable rule in the database. The access control rule is built from the access control data stored on the secure element. If no available rule is found, the engine checks the data on the secure element to decide, whether to build a new rule or reject this access request. The format of the access control data depends on the underlying file structure in secure elements, for example, the PKCS#15 structure could be used on the UICC.

4.3 UICC Access through RIL

The radio interface layer (RIL) is an abstraction layer between the smartphone operating system and the baseband. For example, Android provides a RIL between telephony services, used to monitor the basic phone information, and the baseband processor. This RIL consists of two primary components [?]:

- **RIL daemon**: initializes the vendor RIL, processes all communication from Android telephony services, and dispatches calls to the vendor RIL as solicited commands.
- **Vendor RIL**: is baseband-specific, processes all communication with the baseband processor, and dispatches calls to the RIL daemon through unsolicited commands.

The communication between the vendor RIL and the baseband is in the form of Hayes AT commands [?]. If a customer application wishes to communicate with the UICC through the RIL, the required AT commands must be supported by the RIL. However, the Android telephony system does not provide the needed APIs to access the UICC, and the RIL does not support all the needed AT commands, which prevents access to the UICC from user applications.

4.4 OpenMobileAPI

The SIMAlliance OpenMobileAPI is a software interface between applications on mobile phones and applets on secure elements. It is designed to meet the need of accessing the secure element through the RIL layer on a smartphone. A three-layer architecture of the API is defined in [?] as follows:

- **Transport layer**: provides general access to secure elements using APDUs. This layer forms the foundation for the other layers.
- **Service layer**: abstracts the available functions, such as cryptography and authentication, in secure elements. The application developer can utilize the services provided by the secure element without concerning about the underlying transport API.
- **Application layer**: represents the various applications that use OpenMobileAPI.

Using OpenMobileAPI as the secure element access interface, the dual application architecture can be realized as depicted in Figure ??. The mobile application part of the local OP lies in the application layer. By calling APIs from the service layer, the application can securely store the secret on the UICC, verify graphics/OpenMobileAPIArchitecture.pdf

Fig. 4: Open Mobile API in Dual Application Architecture

PIN to locally authenticate the end user, sign the authentication assertion using the HMAC (Hash-based message authentication code) function from the crypto API, transmit data with the generic transport API, and so on. All these service requirements are converted into command APDUs in the transport layer and sent to the applet on the UICC.

5 Implementation

We have implemented the local OP according to the dual application architecture as an Android application on a Samsung Galaxy S2 i9100P and a UICC applet on a SIMply ON product from Morpho e-Documents [?]. For the OPSF and local OP we used the Java based OpenID libraries [?].

The communication channel in the local OP is provided by OpenMobileAPI integrated in the Android phone[5]. The access control data on the UICC is formatted and stored in a PKCS#15 file structure to grant the access from the Android application to the applet. The Android Intent mechanism is used to enable the redirection from the browser to the local OP application.

[5] Currently only several NFC-enabled Android phones contain the SIMAlliance OpenMobileAPI

Figure ?? shows screenshots of a login process. The user enters his identifier at the RP, as shown in (a). We chose the popular developer community site Stack Overflow for demonstration. After pressing the "Log in" button on this page, this RP enters an association exchange with our OPSF at `op.demo.novalyst.de`. The OPSF creates an association for authentication of the user with identifier "janesmith". Note that both the local OP as well as the OPSF counterpart are enabled to handle multiple identifiers. After that, the RP redirects the user browser to the OPSF address and the OPSF in turn issues the trigger for the local OP application, as described in Section ??. All this happens in the background and the user is now presented with the first prompt of the SmartOpenID app. In the case shown in (b), the local OP applet on the smart card is still locked and the user is asked in (c) to enter his PIN to unlock it. Note that the user may not have to type in his/her PIN for each authentication to an RP site, but only once when the device is powered up, or only once every m seconds, or as required by a locally configured policy. In this case, i.e., when the local OP applet on the smart card is in the unlocked state, steps (b) and (c) disappear and the log in to the RP site becomes rather seamless. After successful local user authentication via the PIN, the user authorizes the creation and submission of the signed assertion message to the rightful (i.e., requested by the user) RP in the confirmation dialog (d). After the user pressed on "Authenticate", the SmartOpenID app creates an assertion message and the local OP applet on the smart card creates a signature over it, during which the user sees the progress screen (e). The signed assertion is sent to the RP via the user's browser and the user is logged in at (f) and may for instance see his personal profile (g).

In the demo, the HMAC-SHA1 algorithm is chosen as the algorithm to sign the authentication assertion, and the PBKDF2-HMAC-SHA1 is implemented in the applet according to the specification [?] to provide the key derivation function. A performance measurement is performed to show the feasibility of using the UICC as the secure element in Smart OpenID. For the PBKDF2 function, the master key is our shared secret S having a size of 64 bytes, and the salt is the association handle `asc_hdl` with 240 bytes. An iteration count of the underlying function (HMAC-SHA1) is included to increase the difficulty of attacks using exhaustive search. A set of experiments are defined with the iteration count from 1 to 1024 rounds. According to the result depicted in Figure ??, the iteration count should be less than 64 rounds in order to reduce the waiting time of the user. The consumed time for transmitting APDUs from the Android application to the UICC applet via OpenMobileAPI, and the performance of the HMAC-SHA1 algorithm for signing the received message are measured and depicted in Figure ??.

6 Analysis of Smart OpenID

Smart OpenID does not require changes to the OpenID protocol at RPs, which means that it can be directly deployed. Some vulnerabilities of the normal OpenID protocol are addressed by the OpenID Security Best Practices [?]. Phishing of user credentials for an OpenID identifier should be of special concern, as OpenID enables access to all RP services and replaces multiple passwords with one credential. Smart OpenID increases the security by introducing two-factor authentication bound to the device and secure element, providing security from misuse in the case of a stolen or lost phone. The user only has to remember his PIN code for the Smart OpenID application, which is only communicated to the smart card. It is possible to use varying grades of multi-factor local authentication, e.g. biometrics, to further increase the security. In addition, using OTA mechanisms, an MNO may provide a service to remotely delete the smart card applet and thus prevent usage of stolen cards, and further securely manage the life-cycle of a local OP. In Smart OpenID, the user is notified by a uniform, local notification GUI element if a login process is taking place and to which RP, preventing the issue of silent logon by CSRF attacks.

The local OP provides not only a secure storage of credentials, but due to its function as an assertion issuing entity, also enables attribute based access control with some level of privacy for the user. The concept allows MNOs to easily provide IdM as a service to RPs, monetizing user data that they already possess. Thus, Smart OpenID can create privacy towards the network and still issue trustworthy assertions on a user's identity or attributes to RPs.

In our experiments, we successfully logged in to various RPs on the Internet. Since communication to the remote server in normal OpenID is replaced by faster local communication, the user even perceives less delay using the local OP. The smart cards currently used in mobile networks are equipped with anti-tamper devices and using cryptographic protection which makes cloning of the card very difficult. Given the fact that an attacker needs physical access in order to attack the card's contents, the user has enough time to react and report the loss to his MNO to block further usage of the card. As OpenID addresses the issues of the password dilemma, the presented combination with a strong, smart card based, local authentication provides an efficient protection from attacks, and creates a secure and seamless solution for SSO. Our current research is focused on mobile scenarios, and hence uses the UICC for the local OP. Smart OpenID can be extended to other smart cards or security tokens, such as secure micro SD cards, which are all based on the Javacard platform. The split-terminal scenario described in [?] can serve as a blueprint for the extension of the concept to other devices, such as laptops.

We have performed a semi-formal security analysis of Smart OpenID method using attack trees [?], with a high-level overview of the result shown in Figure **??**.

The assumptions underlying the analysis are as follows: the attacker knows the Smart OpenID identifier of the victim and has knowledge of the protocol, thus intends to either uncover the credential (the shared secret) or violate the identity of the victim indirectly. Instead of transmitting the credential directly, a derived key is computed and used. The length of the credential is 64 bytes, which has a obviously larger key space than the normal password, thus a cryptanalysis of the derived key is computational impractical. The retry counter of PIN prevents the lost UICC from being misused. Similarly to the countermeasure in OpenID, the replay and the session swapping attack can be avoided by using the nonce value in the protocol. The evaluation of the attack tree leads to the conclusion that the main practical attack path would be the man-in-the-middle attack, which is caused if the RP cannot properly authenticate the network OP. This shows that successful attacks on Smart OpenID are unlikely for an attacker with skills such as phishing and sniffing, and computing resources for brute-force attacks.

7 Conclusions

In this paper, we have presented an implementation of Smart OpenID, an approach that allows using the lightweight identity management protocol OpenID while leveraging the security properties of smart cards. While the Smart OpenID approach has been presented in a previous publication, we have shown its practical applicability by showing that existing UICCs have sufficient computational power for all algorithms required. In addition, we have provided an analysis of its security using an an attack tree. Due to the widespread availability of UICCs, and since the approach is based on the widely accepted OpenID protocol, we believe that Smart OpenID can be a valuable contribution to provide a practical security solution that deals with major security issues in internet-based services.

References

1. Smith, R.E.: The strong password dilemma. In: Computer Security Journal, vol. 18 (Computer Security Institute) (2002) 31–38
2. Brown, A. S., Bracken, E. ,Zoccoli, S., Douglas, K.: Generating and remembering passwords Applied Cognitive Psychology, vol. 18 (2004) 641–651
3. Liberty Alliance Project. Web page at http://www.projectliberty.org (2002)
4. Chappell, D., et al.: Introducing windows cardspace. MSDN. April (2006)
5. Bertocci, V., Serack, G., Baker, C.: Understanding windows cardspace Addison-Wesley Professional (2007)
6. Higgins Personal Data Service. http://www.eclipse.org/higgins/
7. Leicher, A., Schmidt, A.U., Shah, Y.: Smart OpenID: A Smart Card Based OpenID Protocol. In: D. Gritzalis, S. Furnell, and M. Theoharidou (Eds.): SEC 2012, IFIP AICT 376, pp. 7586 (2012)
8. Chen, Z.: Java Card Technology for Smart Cards. Prentice Hall (2000)
9. ISO : ISO 7816-4: Identification cards - Integrated circuit cards - Organisation, security and commands for interchange (2005)
10. SIM Alliance: OpenMobile API Specification v2.0.2. http://www.simalliance.org (2011)
11. OpenID.net: OpenID Specifications. http://openid.net/developers/specs/
12. Uruena, M., Busquiel, C.: Analysis of a Privacy Vulnerability in the OpenID Authentication Protocol. IEEE Multimedia Communications, Services and Security (MCSS2010)(Krakow, Poland, 2010)

13. van Thanh, D., Jonvik, T., Feng, B., Van Thuan, D., Jorstad, I.: Simple strong authentication for internet applications using mobile phones. In: IEEE GLOBECOM Global Telecommunications Conference, 2008. (2008)
14. 3GPP: Remote APDU Structure for (U)SIM Toolkit applications. TS 31.115, 3GPP (December 2009)
15. 3GPP: Remote APDU Structure for (Universal) Subscriber Identity Module (U)SIM Toolkit applications. TS 31.116, 3GPP (December 2009)
16. Schmidt, A.U., Leicher, A., Shah, Y., Cha, I.: Efficient Application SSO for Evolved Mobile Networks. In: Proceedings of the Wireless World Research Forum Meeting 25 (WWRF 25), London, UK, 2010
17. Tsyrklevich, E., Tsyrklevich, V.: Single Sign-On for the Internet: A Security Story. BlackHat Conference Las Vegas 2007 (2007)
18. Urien, P.: Convergent identity: Seamless OpenID services for 3G dongles using SSL enabled USIM smart cards. In: Consumer Communications and Networking Conference (CCNC), 2011 IEEE, IEEE (2011) 830–831
19. Urien, P.: An OpenID provider based on SSL smart cards. In: 7th IEEE Consumer Communications and Networking Conference (CCNC). (2010)
20. Leicher, A., Schmidt, A.U., Shah, Y., Cha, I.: Trusted Computing enhanced OpenID. In: 2010 International Conference for Internet Technology and Secured Transactions (ICITST). (2010) 1–8
21. Telco 2.0: Telco 2.0 Manifesto - Business Model Innovation for the Digital Economy. `http://www.stlpartners.com/manifesto.php`
22. Camenisch, J., Fischer-Huebner, S., Rannenberg, K.: Privacy and Identity Management for Life. Springer Verlag (2011)
23. Koschinat, S., Bal, G., Rannenberg, K.: Economic Valuation of Identity Management Enablers. PrimeLife Deliverable D6.1.2 (May 2011)
24. Koschinat, S., Bal, G., Weber, C., Rannenberg, K.: Privacy by sustainable identity management enablers. Privacy and Identity Management for Life (2011) 431–452
25. 3GPP: 3G Security; Generic Authentication Architecture (GAA); System description. TR 33.919, 3GPP (June 2010)
26. 3GPP: Identity management and 3GPP security interworking; Identity management and Generic Authentication Architecture (GAA) interworking. TR 33.924, 3GPP (June 2011)
27. Weik, P., Wahle, S.: Towards a generic identity enabler for telco networks. In: Proc. 12th Internat. Conf. on Intelligence in Networks (ICIN 2008), Bordeaux. (2008) 20–23
28. Ahmed, A.S.: A User Friendly and Secure OpenID Solution for Smart Phone Platforms. Master's thesis, Aalto University, School of Science and Technology, Faculty of Information and Natural Sciences (2010)
29. Jorstad, I., Johansen, T., Bakken, E., Eliasson, C., Fiedler, M., et al.: Releasing the potential of openid & sim. In: Intelligence in Next Generation Networks, 2009. ICIN 2009. 13th International Conference on, IEEE (2009) 1–6
30. Liberty Alliance: ID-WSF Advanced Client 1.0 Specifications. Technical report, (2007)
31. Liberty Alliance: ID-WSF Advanced Client Implementation and Deployment guidelines for SIM/UICC Card environment. Technical report, (2007)
32. GSM Association. NFC Handset APIs & Requirements Version 2.0. `http://www.gsma.com/mobilenfc/wp-content/uploads/2012/03/gsmanfcuiccrequirementsspecificationversion20.pdf`, 11 2011.
33. S. Verclas and C. Linnhoff-Popien. *Smart Mobile Apps: Mit Business-Apps ins Zeitalter mobiler Geschäftsprozesse.* Springer, 2011.

34. Android Platform Developer's Guide. Radio layer interface. http://www.kandroid.org/online-pdk/guide/telephony.html.
35. 3GPP. 3GPP TS 27.007 AT command set for User Equipment (UE). http://www.3gpp.org/ftp/Specs/html-info/27007.htm, 3 2012.
36. Morpho e Documents. SIMply ON Java UICC for GSM and 3G networks. http://www.morpho.com/IMG/pdf/morpho_telco_simply_on_2p_gb-3.pdf.
37. OpenID4Java Project: OpenID 2.0 Java libraries. http://code.google.com/p/openid4java/
38. B. Kaliski. PKCS #5: Password-Based Cryptography Specification Version 2.0. RFC 2898 (Informational), September 2000.
39. OpenID Foundation: OpenID security best practices. http://wiki.openid.net/OpenID-Security-Best-Practices
40. S. Mauw and M. Oostdijk. Foundations of Attack Trees. In Proceedings of ICISC. 2005, 186-198.

screenshots/01_UserEnterIdentifier.pdf

(a)

screenshots/02_PleaseUnlockDialog.pdf

(b)

screenshots/03_UserEntersPinCode.pdf

(c)

screenshots/04_LoginConfirmationDialog

(d)

screenshots/05_AuthenticationOngoing.jpg

(e)

screenshots/06_UserLoggedIn.pdf

(f)

screenshots/07_UserProfilePage.pdf

(g)

Fig. 5: Smart openID Login in Practice

`graphics/PerformancePBKDF2Iteration.pdf`

Fig. 6: Performance Measurement for PBKDF2-HMAC-SHA1 with the Iteration Count from 1 to 1024

`graphics/PerformanceSignature.pdf`

Fig. 7: Performance Measurement for APDU Transmission and HMAC-SHA1

`graphics/AttackTree.pdf`

Fig. 8: Attack Tree Analysis of Smart OpenID