Using a Smartphone to Access Personalized Web Services on a Workstation

Faysal Boukayoua, Jan Vossaert, Bart Decker, Vincent Naessens

Using a Smartphone to Access Personalized Web Services on a Workstation

Faysal Boukayoua[1], Jan Vossaert[1], Bart De Decker[2], and Vincent Naessens[1]

[1] Katholieke Hogeschool Sint-Lieven, Department of Industrial Engineering
Gebroeders Desmetstraat 1, 9000 Ghent, Belgium
firstname.lastname@kahosl.be
[2] Katholieke Universiteit Leuven, Department of Computer Science,
Celestijnenlaan 200A, 3001 Heverlee, Belgium
firstname.lastname@cs.kuleuven.be

Abstract. This paper presents a privacy-friendly mobile authentication solution. It addresses several shortcomings of conventional methods, such as passwords and smartcard solutions. It also meets the needs of an increasingly mobile user. Trust in the client computer is minimal and the authentication is entirely delegated to the smartphone, which makes it portable across different workstations. Our approach involves authentication using securely stored credentials on the smartphone. The client workstation does not need to be modified, whereas only minor changes to the Web server are required.

Key words: mobility, user-centric identity management, privacy, security

1 Introduction

In recent years, there has been an ever increasing growth of personalised Web services. Moreover, users are no longer bound to one workstation for their online activities. With these two trends, the need emerges for a portable and privacy-friendly credential solution. Two frequently used credential types are passwords and PKI solutions embedded in smartcards, both of which have a number of drawbacks. Passwords form a rather weak authentication means and do not allow the service provider to obtain reliable information about the user. Smartcard solutions, on the other hand, require the presence of extra hardware, more specifically card readers. Furthermore, installation of middleware can pose an impeding administrative overhead or is simply not possible due to insufficient permissions on the workstation. The middleware must also typically be trusted, something users are not always able or willing to do.

Contribution. This paper proposes a strategy for mobile authentication to a remote Web server in order to get access to a resource on that server. It involves authentication using securely stored credentials on a mobile device. Our solution does not require installing additional software on the client workstation, whereas

only minimal modifications are needed on the server side. The prototype consists of a pluggable module in a servlet container. This approach also has several advantages over existing solutions. First, authentication is carried out by the mobile device, which substantially increases credential portability across workstations. Moreover, our solution is highly secure because it provides strong authentication and it stores credentials on a tamperproof secure element. Support for a wide range of mobile devices is also ensured by only using the camera for the transfer of the authentication request from the workstation to the smartphone. The user's privacy is preserved as a result of the controlled release of personal attributes (or properties thereof). Finally, the solution is flexible enough to integrate other credential technologies.

The paper is structured as follows. Section 2 lists the requirements of the prototype application. Section 3 presents the attack models we will be covering in the evaluation. Sections 4 and 5 discuss the design and implementation respectively. Section 6 evaluates our prototype. Finally, a number of conclusions are drawn.

2 Requirements

2.1 Functional requirements

F_1 Users want to securely gain access to personalised Web-based services from an arbitrary workstation, in a privacy-preserving manner.
F_2 Service providers want to obtain reliable user information.
F_3 Identity providers have the task of provisioning reliable user information to authenticated trusted modules (claim-based identity management)

2.2 Usability requirements

U_1 Our solution can be applied in conjunction with a broad range of mobile devices.
U_2 No extra software may be installed on the workstation.

2.3 Security and privacy requirements

The architecture from [13] is used because it meets several security and privacy requirements. It allows for selective disclosure of attributes and their properties: these are only released if allowed by the access policy. Besides the fact that credentials never leave the trusted module on which they are stored, attributes and their properties are only transferred to the service provider over a secure, authentic channel. Furthermore, the user must authenticate using a PIN, before the trusted module is activated. The architecture also provides the service provider with reliable user information and allows for identification of users by a trusted third party in case of abuse.

Other imposed security requirements are:

S_1 It must not be possible to deceive the user about which service provider he is connecting to.
S_2 The user should be able to authenticate to a service provider.
S_3 In case of loss or theft of the smartphone, no unauthorised parties may gain access to the user's credentials or authenticate on his behalf.
S_4 Data authentication and confidentiality should be ensured between:
(a) the workstation and the service provider
(b) the smartphone's trusted module and the service provider
(c) the smartphone's trusted module and the identity provider
S_5 We reasonably assume that the workstation being used, does not possess any trusted code execution capabilities. The device may either belong to the user or to an untrusted third party. Therefore, we aim to reduce trust in the workstation to the smallest possible extent.

3 Attack models

We assume at all times that the attacker does not remotely control the identity management software running on the mobile device. This can be enforced using trusted code execution technologies. Typical attack scenarios may include:

- A malicious network administrator that tries to collect information about the user or even actively manipulates the communication channel in order to impersonate the user or gain potentially valuable information.
- The user's workstation could be infected by one or more pieces of malware. The objectives that lie behind malware, may diverge, depending on their origins. While online organised crime may be out to unlawfully obtain banking credentials, a surveilling government is more likely to focus on profiling the user's online behaviour and possibly on obtaining various authentication credentials and on gaining unsolicited access to the user's services [5].
- The user's smartphone could be stolen. Theft, whether it be for the sake of the phone or the contained credentials and sensitive data, causes the latter to be exposed to the attacker's control, if not properly protected.

Partly arising from the aforementioned requirements and assumptions, the following attack scenarios are analysed and discussed in the evaluation (section 6.1):

A_1 The attacker eavesdrops on network traffic and/or modifies messages.
A_2 The attacker has control over the workstation, apart from the browser.
A_3 The attacker has full control over the workstation, including its browser.
A_4 The attacker acquires physical control over the mobile device.

4 Design

The approach mainly focuses on the smartphone authenticating its user and asserting claims about his personal information. It uses trusted module technology that is readily available in smartphones (i.e. a secure element) or with which

smartphones can be extended (i.e. a SIM card or a secure micro SD card[6]). Trusted modules are tamperproof devices that offer facilities for secure storage of user credentials and key material. They also also provide high-grade cryptographic security.

Our system consists of a software component on the trusted module as well as a minimal middleware part on the smartphone. The trusted module component contains the authentication credentials and logic. Embedding the credentials in a such a device, has several advantages. Since smartphones are vulnerable to theft, a trusted module ensures that the credentials cannot be abused. In addition, some identity management systems (e.g. [13], [1]) rely on the secure computing environment of trusted modules.

The smartphone component allows the user to activate the trusted module, in order to authenticate and gain access to personalised services. Moreover, it enables communication with service providers and can forward communication between the trusted module and the service provider.

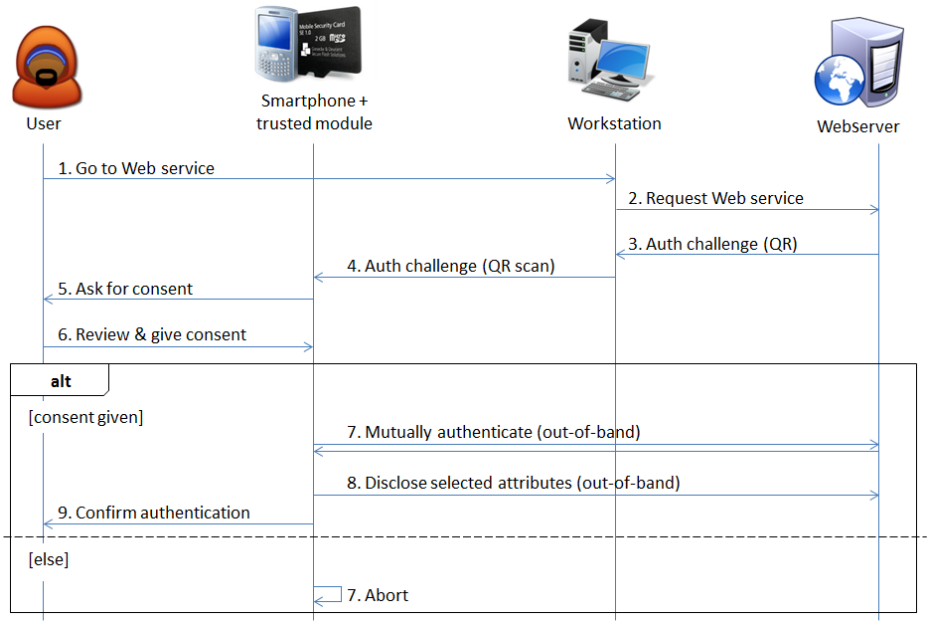

Fig. 1. Overview of the application.[3]

Authenticating HTTP(S) sessions using the smartphone. As the authentication device (i.e. the smartphone) differs from the device on which the Web service is

[3] For the sake of clarity, card revalidation as well as on the fly attribute fetching from identity providers, are not displayed. The latter one is carried out between step 6 and 8 if an attribute (or property thereof) does not reside on the trusted module.

accessed (i.e. the workstation), a strategy has been devised to allow the server to bind an authentication to an already existing `HTTPS` session context. This strategy is further described in this section.

Nowadays many communication technologies are supported by the current generation of smartphones. However, most of them require additional software to be installed in order to allow interaction with the smartphone. Optical communication on the other hand, can easily be used to transfer data from the browser – running on the workstation – to the mobile. The procedure discussed below and depicted in figure 1 on the facing page, exploits the workstation's screen and the smartphone's camera to transfer information between the two:

1. The browser initiates an `HTTPS` session with the Web service.
2. The Web service sends a Web page containing a two-dimensional bar code (or QR-code) representation of the authentication challenge. This challenge consists of the following three parts:
   (a) *the authentication ID*: this is merely a replacement for the session ID, which would otherwise be vulnerable to theft if the QR code were at the hacker's field of view. The link between the session ID and the authentication ID is stored and resolved by the Web service.
   (b) *the service provider's certificate*, which is issued by the *certificate authority*. It contains the provider's name, address, public key and the attributes that this provider requires or optionally requests.
   (c) *the `minimum revalidation time`*. The architecture in [13] provides a revocation strategy where every so often the trusted module contacts a trusted third party – the *revalidation service*, thereby revalidating itself. If the revocation status is OK, the `revalidation time` on the card is set to the current time. However, if the card has been revoked, no changes are made, effectively rendering the card useless in the subsequent steps. The revalidation time is service provider-specific, as it is dependent on the authentication needs that a Web service imposes. For instance, the verification requirements to get a student discount at a cinema are likely to be less stringent than what is needed to authenticate personnel in a nuclear power plant.
3. The smartphone captures the QR-code that is shown in the browser, using its embedded camera.
4. The smartphone displays the connection parameters and certificate information of the service provider to which the user is about to authenticate.
5. The user reviews the information that is shown. If the displayed service provider is indeed the one he is trying to connect to and if he agrees to release the required personal information, the user gives his consent. Subsequently his smartphone initiates a connection to the Web service's authentication module, using its xG connection.
6. The trusted element in the smartphone and the Web service execute a mutual authentication protocol after which the requested attributes (or properties thereof) are released. The authentication ID is included as an additional attribute and allows the Web service to link the authentication/identification to the `HTTPS` session initiated in the first step.

5 Prototype

This section discusses in more detail the four software components that are developed for the prototype, namely the identity management component on the trusted module and the software on the smartphone, the identity provider and the service provider.

5.1 Trusted module application

For authentication and attribute assertion, the prototype uses the architecture presented in [13]. It is a privacy-friendly, user-centric federated identity management approach which relies on a trusted module. Multiple identity providers can endorse the user's personal information to multiple service providers. The trusted module acts as a mediator between the two. More specifically, an identity provider can store personal attributes (or properties thereof) in each owner's trusted module. Information that is endorsed by identity providers can then be disclosed to service providers. The trusted module controls access to identity information. Before the trusted module releases attributes, it first requires the service provider to authenticate and prove that it is authorized to access the requested personal attributes.

Concerning the trusted module, three types are commonly available on smartphones, namely SIM cards, secure elements and secure micro SD cards. Software components for these modules are typically developed using the Java-Card subset of the Java programming language. This facilitates portability of the the proof-of-concept from [14] to the mobile phone, as it is also developed using the same technology.

Every smartphone is typically equipped with a SIM, which is usually issued by telecom operators and hence installing additional software requires their permission. This hinders deployment of our application. The main difference between the two other trusted modules lies in the fact that secure elements are embedded in the phone and therefore can not be easily transferred to another mobile device. Also, similarly to SIM cards, the manufacturer's permission is required to deploy software on a secure element. The secure micro SD card on the other hand, is easily pluggable into a regular micro SD slot. This facilitates integration for the large majority of smartphones. Moreover, because this type of trusted module doesn't require the assistance of a third party to access it, it also proves to be the better choice for research purposes.

Our prototype is developed using the *Mobile Security Card SE 1.0*, which is manufactured by Giesecke & Devrient[6]. It has a built-in tamperproof module, which is used for the storage of key material and user attributes and credentials. The card can also carry out cryptographic operations without the keys ever having to leave the tamperproof area. It runs the JavaCard 2.2.2 platform and has 2GB of regular non-tamperproof storage, which can, for instance, be used to store encrypted, authenticated data.

5.2 Smartphone application

To enable communication between the trusted module on one hand and service and identity providers on the other hand, a mobile application is developed. This application uses the communication capabilities of the smartphone (e.g. Bluetooth, NFC, xG or camera) to interact with external devices and forward the necessary communication to the trusted module. In the setting of a Web application, xG is used to interact with remote servers. Connections to identity providers may be required in order to obtain requested attributes that do not reside on the trusted module. A communication request is passed to the smartphone application, which then initiates connections to the required identity providers.

Apart from the data transfer, the mobile application also offers multiple management functions to the user. This functionality is related to the identity management architecture presented in [13]. For instance, the application shows the user to which service provider the trusted module is about to authenticate. It does this by interpreting information that is obtained from the connection parameters and the service provider's certificate. Furthermore, the application also displays the user attributes (or properties thereof) that are requested. Attributes may be *mandatory* or *optional*. If any optional attributes are requested, the user can select whether or not to disclose them. The application also allows the user to specify policies, that are then enforced by the trusted module.

5.3 Server extensions

The service provider The server extension allows service and identity providers to support the authentication mechanism. Service providers can specify mandatory and optional attributes for authentication. Mandatory ones are required for the authentication to succeed, while optional attributes could be used, for instance, to enhance the service that is being delivered to the user. The extension handles the authentication and makes the received attributes available to the Web application.

The Web server authentication module was implemented for Apache Tomcat by means of an authentication valve. This valve behaves like a filter for the Web application and can, hence, intercept and handle authentication and authorisation requests. The required server credentials are obtained from a MySQL server. The module can be easily plugged into existing Tomcat applications by carrying out minor changes in the configuration files and adding our solution's `jar file` in Tomcat's `lib` folder.

As mentioned before in section 4 on page 3, the login page that the user sees, displays a QR code that contains the *authentication ID*, the *service provider's certificate* and the *minimum revalidation time* that is imposed by the service provider.

The service provider utilises a TLS layer to ensure authenticity and confidentiality during its communication with the workstation. However, the same approach is hard to apply to communication to and from the trusted module,

due to resource constraints that the latter poses. Therefore, instead of relying on TLS, an authenticated Diffie-Hellman key agreement protocol is used to authenticate both parties and to establish a session key for an end-to-end secure and authentic channel. The protocol is implemented using the cryptographic facilities that are provided by the secure micro SD card. Both key agreement as well as signatures are carried out entirely on the card, thereby never disclosing the private key material. More information about the concrete protocols can be found in [13].

The identity provider The attribute provisioning by identity providers is largely based on the architecture in [13]. It provides support for multiple identity providers and has an *audit authority* that assigns the provisioning of a certain attribute to the appropriate provider(s). The same authority also determines the set of attributes that a certain service provider may request. The architecture thereby adheres to the principle of *justifiable parties*, from Kim Cameron's 7 Laws of Identity[3].

The identity providers are implemented as Web services that each use their respective MySQL database to retrieve and subsequently disclose requested attributes. Provisioning requests are obviously only met upon successful mutual authentication between the identity provider and the user's trusted module. Similarly to the service providers, further communication - e.g. the provisioning of user attributes - takes place through and end-to-end secure and authentic channel.

6 Evaluation

6.1 Validation against attack models

This section evaluates the prototype against the attack scenarios that have been mentioned in section 3. We refer to these scenarios using their numbering "A_x".

M_1 A_4: **The attacker acquires physical control of the mobile device.** User credentials are securely stored on a trusted module, which is only activated upon successful PIN authentication by the user. As a result, the attacker will not learn anything about the credentials, their attributes or properties thereof. Nor will he be able to use them. In addition, cards with a compromised PIN can be revoked.

M_2 $A_1 + A_2 + A_3$: **The attacker has full control of the workstation, including the browser. In addition, he can also eavesdrop on network traffic and/or modify messages.** As credentials are neither stored on nor transferred to the workstation, the attacker will never learn anything about them. In this respect, our system has better security properties than some existing two-factor authentication systems, like Google's, since authentication takes place entirely out-of-band. Furthermore, the mobile application feedback empowers the user to choose whether or not to connect to the given

service provider. As a result, the user is better protected against malware directing him to unwanted providers.

M_3 $A_1 + A_2$: **The attacker has control of the workstation, but not of the browser.** Besides the measures discussed in M_2, the user is also protected against man-in-the-middle attacks that would cause him to authenticate a forged session to the same service provider.

M_4 A_1: **The attacker can eavesdrop on network traffic and/or modify messages.** The remarks from M_3 also apply here. In addition to that, the attacker will not be able to read, nor inconspicuously modify communication traffic between the smartphone and the service provider. The same applies to traffic going from the workstation to the service provider, assuming the user properly communicates using HTTPS.

6.2 Comparison with other solutions

The solution proposed in this article, devises a protocol as well as a token mechanism. It is useful to divide this section according to these two categories in order for the comparison to make sense.

Comparison of tokens Passwords are very cheap in terms of financial cost. But they only offer low-grade security and no possibilities for personalisation. Moreover, they pose usability problems due to the typical limitations of the human memory. Passwords also require trust in the workstation and hence they are vulnerable to theft.

Smartcards are superior to passwords when it comes to security properties, as nearly every modern smartcard has cryptographic facilities on board. They are often issued for a single application: e.g. banking, social security, student cards,... This can potentially lead to a proliferation of cards. eID cards, however, can be reused throughout different applications, which mitigates the manageability problem to a certain extent. We further evaluate the case of the German eID. It is one of the most privacy-friendly designs [1]. Their drawback, however, lies in the requirement of certified card readers, which decreases usability. This can pose a problem in some countries, as these devices are not commonly available. In addition, users might not always have the required privileges to install them. Finally, most governmental eID architectures also require the user to install additional software on the workstation.

Software tokens on the other hand, impose no additional hardware cost, and similarly to smartcards, they have good cryptographic properties. However, they typically require changes to, as well as trust in the workstation: software tokens on an infected workstation are prone to theft and, subsequently, to offline attacks and misuse. Tokens like X.509 certificates also do not offer any mechanisms for selective attribute disclosure. In addition, the issuance of these certificates is rarely free of charge.

Related solutions in literature, such as [10], rely on a smartphone and transfer a proxy credential to the browser on the workstation. The browser can then use

this credential to set up mutually authenticated `HTTPS` sessions with servers. This has the advantage that no additional software needs to be installed on Web servers. However, it requires the user to install a software component to transfer the proxy credential from the smartphone to the browser. Moreover, since traditional X.509 certificates are used, the user's actions can be linked. Furthermore, the credentials are transfered to the workstation, which implies that significant trust is required in the workstation (i.e. these technologies are not usable on untrusted workstations). Even though the proxy credentials are limited in lifetime, abuse is still possible.

Another popular solution are hardware tokens, such as RSA SecurID devices. They offer high-grade security and typically do not require any changes to the user's workstation. However, hardware tokens are often issued for a single application, which can potentially cause a proliferation of these devices and thereby give rise to a usability issue. In addition, similarly to many smartcards, issuing these tokens brings about an extra hardware cost per new service provider. Lastly, hardware tokens typically do not provide any user-controlled attribute disclosure functionality.

The application proposed by this paper also provides high-grade security. It allows service providers to personalise their services and users to protect their privacy, by virtue of the selective attribute disclosure mechanism. Only one smartphone with a trusted module is needed. This is beneficial in terms of usability as well as marginal hardware and software cost. The most substantial investment to be done, is the initial one: adding extra service providers only comes at an additional administration cost. In addition, since our solution is mobile, credentials are highly portable across workstations. Moreover, trust in the workstation is minimised, as credentials never leave the trusted module on which they are stored and because the authentication procedure takes place entirely out of band - using the phone's xG connection. The inconvenience that the need for smartphone connectivity brings about, does not outweigh the security advantages. Mobile internet is becoming increasingly common, a trend that is only expected to continue in the following years.

Comparison of protocols Traditional (user name and password) authentication mechanisms do not have any notion of federated identity management: there is no clear division into identity and service providers. The user's data is usually managed and stored by the service provider. The latter is generally not assured about the authenticity of the provided user data, while the user is confronted with duplicate registrations and potential privacy breaches. Phishing attacks are also likely to happen on high-profile websites.

OpenID and Shibboleth are examples of network-based identity management. They both rely on a single identity provider for authentication and attribute provisioning. A disproportional amount of trust lies with a single identity provider, concerning the confidentiality of user information. Moreover, direct communication between the two types of providers, is inherent to OpenID and Shibboleth, which allows identity and service providers to collude and thereby uncover ad-

ditional information about the user. Systems such as these are also prone to phishing attacks[4][11]. User-controlled attribute disclosure is not provided in OpenID, whereas s Shibboleth is capable of such functionality using an additional plugin.

Though it has been discarded in Februari 2011, Windows CardSpace is another interesting application to look at. Similarly to our solution, CardSpace is a claims-based identity management architecture with support for multiple identity providers. The latter cannot collude against the user with any service provider, since there is a strict separation between the two. The system has a high resistance against phishing attacks, as discovery information about identity providers is securely stored on the user's workstation[4]. Windows CardSpace also provides facilities for user-controlled attribute disclosure. The major drawback of CardSpace, however, is the lack of portability to other platforms: it requires at least Windows XP and Internet Explorer 7. In addition, it relies on NTFS encryption for its secure storage[9]. Another disadvantage is that the use of CardSpace is typically limited to the workstation it is installed on. So-called InfoCards can be exported and imported back again, but no *native* way is provided to use credentials in a mobile manner.

A similar approach to our solution is presented in [8]. However, there are two important differences. First, our solution allows the use of multiple identity providers. Moreover, in [8] the user has to go through an unverifiable registration process for each service. This approach also implies long-term storage of the user's data by the service provider, which increases the risk of information loss and privacy breaches.

Our solution has the same privacy, security and architectural advantages as those mentioned in the case of CardSpace. But in addition, the application is designed from the ground up to be used in a mobile context. Therefore, credentials are easily portable from one computer to another. Application portability is also high, since only the smartphone app needs to be adjusted to a different platform. The trusted module and all code that resides on it, can be reused without any modifications. The proposed application also minimises trust in the workstation, but introduces a new trusted component: the smartphone. Nevertheless, in many cases the user may reasonably want to trust his smartphone rather than f.i. a computer at an Internet café or a library.

7 Discussion

The current prototype focuses on authentication to Web-based services that are used via the workstation. However, there are several other scenarios in which a mobile device can be used to access personalised services. For instance, users might need to prove that they are old enough to buy alcoholic beverages at a vending machine. Other scenarios include controlled access to buildings, loyalty discounts, etc. In these scenarios, the smartphone application may use NFC or Bluetooth to connect to the service provider. Apart from the communication layer, no other modifications are required.

The smartphone application may also incorporate several other authentication technologies, such as anonymous credentials [2]. This would allow the user to select the most feasible technology. The application could as well maintain a global view of the user's anonymity [12] and offer automatic decision support when attribute values are requested.

Users might unintentionally release additional information. For instance, Bluetooth users are uniquely identifiable. Therefore it would be useful to specify an *anonymity level*. Depending on the defined level, the application can decide to disable certain communication technologies or use anonymous networks [7] to achieve the required anonymity level. In *anonymous mode*, the application may also deny authentication requests in which unique information is requested.

Another useful topic to consider, are the variations in backup, registration and revocation strategy that are brought about when the proposed application is used in a different context. A governmental authentication infrastructure will likely have other properties than a public transport company or a commercial Web service provider. These changes will also manifest when switching to a different underlying credential technology.

Yet another interesting research track to look at, is how trusted code execution can be enforced in smartphones. Enforcing trust on mobile devices would benefit the user, the identity providers, as well as the service providers.

8 Conclusion

This paper has presented a prototype application that includes server and client side support to enable personalised access to Web services. Authentication is delegated from the workstation to the user's smartphone. The solution focuses on minimizing trust in the workstation while still allowing privacy-friendly personalized Web services. The proposed approach was compared to existing solutions in literature and several interesting research tracks were proposed.

References

1. Jens Bender, Dennis Kügler, Marian Margraf, and Ingo Naumann. Sicherheitsmechanismen für kontaktlose chips im deutschen elektronischen personalausweis. *Datenschutz und Datensicherheit - DuD*, 2008.
2. Jan Camenisch and Els Van Herreweghen. Design and implementation of the *idemix* anonymous credential system. In Vijayalakshmi Atluri, editor, *ACM Conference on Computer and Communications Security*, pages 21–30. ACM, 2002.
3. Kim Cameron. The 7 laws of identity. The Identity Weblog, January 2006.
4. David W. Chadwick. Foundations of security analysis and design v. chapter Federated Identity Management, pages 96–120. Springer-Verlag, Berlin, Heidelberg, 2009.
5. Chaos Computer Club. Chaos computer club analyzes government malware. Technical report, Chaos Computer Club, 2011.

6. Giesecke & Devrient. Mobile security card se 1.0: A secure flash solution. http://www.gi-de.com/gd_media/media/en/documents/brochures/mobile_security_2/Mobile-Security-Card-SE-1-0_EN.pdf, 2010.
7. Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *In Proceedings of the 13th USENIX Security Symposium*, pages 303–320, 2004.
8. B. Dodson, D. Sengupta, D. Boneh, and M. S Lam. Secure, consumer-friendly web authentication and payments with a phone. In *Conference on Mobile Computing, Applications, and Services (MobiCASE'10), Santa Clara, CA, USA*, 2010.
9. B. Dorrans. An introduction to cardspace.
10. Massimiliano Pala, Sara Sinclair, and Sean Smith. Porki: Portable credentials via proxy certificates in web environments. In *Public Key Infrastructure*. Springer, 2010. Accepted.
11. E. Tsyrklevich and V. Tsyrklevich. Single sign-on for the internet: a security story. Technical report, 2007.
12. Kristof Verslype and Bart De Decker. Measuring the user′s anonymity when disclosing personal properties. In *MetriSec*. IEEE, 2010.
13. Jan Vossaert, Jorn Lapon, Bart De Decker, and Vincent Naessens. User-centric identity management using trusted modules. In *Public Key Infrastructure*. Springer, 2010. Accepted.
14. Jan Vossaert, Pieter Verhaeghe, Bart De Decker, and Vincent Naessens. A smart card based solution for user-centric identity management. In *PrimeLife International Summer School, Helsingborg, Sweden, 2-6 Aug 2010*. Springer, 2011. Accepted.