IFIP AICT 396

Simone Fischer-Hübner
Elisabeth de Leeuw
Chris Mitchell
(Eds.)

Policies and Research in Identity Management

Third IFIP WG 11.6 Working Conference, IDMAN 2013
London, UK, April 2013
Proceedings

Springer

IFIP Advances in Information and Communication Technology 396

IFIP – The International Federation for Information Processing

IFIP was founded in 1960 under the auspices of UNESCO, following the First World Computer Congress held in Paris the previous year. An umbrella organization for societies working in information processing, IFIP's aim is two-fold: to support information processing within its member countries and to encourage technology transfer to developing nations. As its mission statement clearly states,

> *IFIP's mission is to be the leading, truly international, apolitical organization which encourages and assists in the development, exploitation and application of information technology for the benefit of all people.*

IFIP is a non-profitmaking organization, run almost solely by 2500 volunteers. It operates through a number of technical committees, which organize events and publications. IFIP's events range from an international congress to local seminars, but the most important are:

- The IFIP World Computer Congress, held every second year;
- Open conferences;
- Working conferences.

The flagship event is the IFIP World Computer Congress, at which both invited and contributed papers are presented. Contributed papers are rigorously refereed and the rejection rate is high.

As with the Congress, participation in the open conferences is open to all and papers may be invited or submitted. Again, submitted papers are stringently refereed.

The working conferences are structured differently. They are usually run by a working group and attendance is small and by invitation only. Their purpose is to create an atmosphere conducive to innovation and development. Refereeing is also rigorous and papers are subjected to extensive group discussion.

Publications arising from IFIP events vary. The papers presented at the IFIP World Computer Congress and at open conferences are published as conference proceedings, while the results of the working conferences are often published as collections of selected and edited papers.

Any national society whose primary activity is about information processing may apply to become a full member of IFIP, although full membership is restricted to one society per country. Full members are entitled to vote at the annual General Assembly, National societies preferring a less committed involvement may apply for associate or corresponding membership. Associate members enjoy the same benefits as full members, but without voting rights. Corresponding members are not represented in IFIP bodies. Affiliated membership is open to non-national societies, and individual and honorary membership schemes are also offered.

Simone Fischer-Hübner
Elisabeth de Leeuw Chris Mitchell (Eds.)

Policies and Research in Identity Management

Third IFIP WG 11.6 Working Conference, IDMAN 2013
London, UK, April 8-9, 2013
Proceedings

Springer

Volume Editors

Simone Fischer-Hübner
Karlstad University, Department of Computer Science
Universitetsgatan 1, 65188 Karlstad, Sweden
E-mail: simone.fischer-huebner@kau.se

Elisabeth de Leeuw
IDTOPIQ - Security & Identity Management
Pracanalaan 80, 1060 RC Amsterdam, The Netherlands
E-mail: elisabeth.de.leeuw@xs4all.nl

Chris Mitchell
University of London, Information Security Group
Royal Holloway, Egham, Surrey TW20 0EX, UK
E-mail: me@chrismitchell.net

ISSN 1868-4238 e-ISSN 1868-422X
ISBN 978-3-642-37281-0 e-ISBN 978-3-642-37282-7
DOI 10.1007/978-3-642-37282-7
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2013933703

CR Subject Classification (1998): K.6.5, C.3, D.4.6, E.3, J.1

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

This volume contains the papers presented at IFIP IDMAN 2013: the Third IFIP WG 11.6 Working Conference on Policies and Research in Identity Management held during April, 8–9, 2013, at Royal Holloway, University of London, UK. Building on the success of IDMAN 2007 and 2010 (which were held in Rotterdam and Oslo, respectively), this conference focused on the theory, technologies, and applications of identity management.

The world of the twenty-first st century is, more than ever, global and impersonal. As a result of increasing cyber fraud and cyber terrorism, the demand for better technical methods of identification is growing, not only in companies and organizations but also in the world at large. Moreover, in our society digital identities increasingly play a role in the provision of eGovernment and eCommerce services. For practical reasons, identity management systems are needed that are usable and interoperable. At the same time, individuals increasingly leave trails of personal data when using the Internet, which allow them to be profiled and which may be stored for many years to come. Technical trends such as cloud computing and pervasive computing make personal data processing non-transparent, and make it increasingly difficult for users to control their personal spheres. As part of this tendency, surveillance and monitoring are increasingly present in society, both in the public and private domains. While the original intention is to contribute to security and safety, surveillance and monitoring might, in some cases, have unintended or even contradictory effects. Moreover, the omnipresence of surveillance and monitoring systems might directly conflict with public and democratic liberties. These developments raise substantial new challenges for privacy and identity management at the technical, social, ethical, regulatory, and legal levels. Identity management challenges the information security research community to focus on interdisciplinary and holistic approaches, while retaining the benefits of previous research efforts.

Papers offering research contributions to the area of identity management were solicited for submission to the Third IFIP WG 11.6 IDMAN conference. There were 26 submissions to IFIP IDMAN 2013. Each submission was reviewed by at least three Program Committee members. The Program Committee decided to accept six full and four short "work in progress" papers. The program also included two invited talks by Angela Sasse, University College London, and Bart Jacobs, Radboud University Nijmegen, as well as a panel on "Risk Analysis Approaches for Identity Management Systems" organized in cooperation with the Norwegian PETweb II project. IFIP IDMAN 2013 was also collocated with a workshop organized by the EPSRC-funded Future of Identity Network of Excellence, which took place immediately after the end of conference. These proceedings include the accepted full and short papers, the keynote paper by Bart Jacobs, as well as short position papers by the panelists.

We would like to thank all authors, especially those who presented their work selected for the program. Moreover, we are very grateful to all PC members and additional reviewers, who contributed with thorough reviews and participated in the PC discussions. We owe special thanks to David Chadwick and Jozef Vyskoc, who volunteered to shepherd some of the accepted papers.

We gratefully acknowledge the contributions of the Organizing Committee Chairs Haitham Al-Sinani and Marcelo Carlomagno Carlos. We also thank the EasyChair conference system provider for granting us free access to their excellent conference management system. Last but not least, we would like to thank all sponsors for their generous support.

January 2013

Simone Fischer-Hübner
Elisabeth de Leeuw
Chris Mitchell

Organization

Conference General Chair

Chris Mitchell	Royal Holloway, University of London, UK

Programme Committee Co-chairs

Simone Fischer-Hübner	Karlstad University, Sweden, IFIP WG 11.6 chair
Elisabeth de Leeuw	The Netherlands

Local Organizing Committee Chair

Haitham Al-Sinani	Royal Holloway, University of London, UK
Marcelo Carlomagno Carlos	Royal Holloway, University of London, UK

Program Committee

Haitham Al-Sinani	Royal Holloway, University of London, UK
Claudio Agostino Ardagna	Università degli Studi di Milano, Italy
Alessandro Armando	DIST - University of Genova, Italy
Bharat Bhargava	Purdue University, USA
Sonja Buchegger	Royal Institute of Technology (KTH), Sweden
David Chadwick	University of Kent, UK
Elisabeth De Leeuw	IdTopIQ, The Netherlands
Simone Fischer-Hübner	Karlstad University, Sweden
Sara Foresti	Università degli Studi di Milano, Italy
Lothar Fritsch	Norwegian Computing Center, Norway
Steven Furnell	University of Plymouth, UK
Stefanos Gritzalis	University of the Aegean, Greece
Thomas Gross	University of Newcastle upon Tyne, UK
Marit Hansen	Unabhängiges Landeszentrum für Datenschutz Schleswig-Holstein, Germany
Hans Hedbom	Karlstad University, Sweden
Alejandro Hevia	University of Chile, Chile
Jaap-Henk Hoepman	Radboud University Nijmegen, The Netherlands
Bart Jacobs	Radboud University Nijmegen, The Netherlands

Lech Janczewski	The University of Auckland, New Zealand
Ronald Leenes	Tilburg University, The Netherlands
Javier Lopez	University of Malaga, Spain
Chris Mitchell	Royal Holloway, University of London, UK
Andreas Pashalidis	K.U. Leuven, Belgium
Aljosa Pasic	Atos Origin, Spain
Siani Pearson	HP Labs, UK
Günther Pernul	Universität Regensburg, Germany
Geraint Price	Royal Holloway, University of London, UK
Muttukrishnan Rajarajan	City University London, UK
Kai Rannenberg	Goethe University Frankfurt, Germany
Einar Snekkenes	Gjövik University College, Norway
Rama Subramaniam	Valiant Technologies, India
Pedro Veiga	FCCN, Portugal
Jozef Vyskoc	VaF, Slovakia
Rose-Mharie Åhlfeldt	University of Skövde, Sweden

Additional Reviewers

Bodriagov, Oleksandr	Kreitz, Gunnar
Broser, Christian	Mitrou, Lilian
Carbone, Roberto	Ranise, Silvio
Deuker, Andre	Reisser, Andreas
Drogkaris, Prokopios	Rodríguez Cano, Guillermo
Greschbach, Benjamin	Sabouri, Ahmad
Hassan, Sabri	Veseli, Fatbardh

Table of Contents

Session 4 - Identity Management with Smart Cards

Session 5 - Federated Identity Management

Towards Practical Attribute-Based Identity Management: The IRMA Trajectory

Gergely Alpár and Bart Jacobs

Institute for Computing and Information Sciences (iCIS),
Radboud University Nijmegen, The Netherlands

Abstract. IRMA is an abbreviation for "I Reveal My Attributes", and at the same time it is the name of a project run by the Digital Security group of the University of Nijmegen and its partners to get attribute-based identity management up and running. This hands-on approach forces us to elaborate many unexplored issues, leading to a better understanding of attributes and their possibilities and challenges.

Cryptographic techniques that enable secure and privacy-friendly attribute-based authentication have been around for more than a decade, see [3,5,6,8]. But what is new is that the latest generation of smart cards is powerful enough to perform the required (non-trivial) cryptographic operations in an adequately efficient manner [9]. Hence only now we see efforts to actually deploy attributes in practice, like the IRMA project[1] at Nijmegen. Two other pilot projects should be mentioned, both of which are carried out by the EU-sponsored ABC4Trust consortium [4]. The Swedish pilot [2] gives anonymous access for elementary school pupils to on-line resources (*e.g.*, chat room), while the Greek pilot [1] enables university students to evaluate lectures anonymously. In both cases eligibility and privacy are of primary importance. Although the IRMA pilot uses the same underlying technology, the objective of our research is more general as we investigate a *broad variety* of attributes and applications. The associated kind of challenges does not appear in these ABC4Trust pilots since each focusses on a single context.

This document gives a brief overview of some of the more salient aspects of the IRMA project.

First of all, attributes are used in a very broad sense as describing some property of a person. This property may be anonymous (non-identifying), such as your gender, or whether or not you are over 18, but in the IRMA context it may also identify you, for example when the attribute is your bank account or social security number. While the underlying technology provides full unlinkability, the attribute values may provide linkability. This usage of identifying attributes may go against the original intention that attributes should be anonymous, but extending their interpretation to (partial) identification greatly extends the application scenarios. For instance, we foresee registration and status attributes for medical personnel (giving access to medical files), for employees (giving access

[1] See `www.irmacard.org` for up-to-date information and developments.

S. Fischer-Hübner et al.(Eds.): IDMAN 2013, IFIP AICT 396, pp. 1–3, 2013.

to premises, networks, and PCs), and for customers (giving benefits, and online access to their purchase/bonus history). Additionally, attributes may be used for a micro medical dossier, with essential (emergency) information.

Second, the IRMA project uses a smart card implementation [9] based on the Idemix technology [7]. However, the essential feature involved is selective disclosure of only a limited number (possibly only one) attributes, while hiding all other attributes. This focus on the core of the credential technology—using zero knowledge proofs—means that the system allows high level interfaces, beneath which other implementations, such as U-Prove, can also be employed. As a simplified view of the technology and the concepts, we may say that "credentials are issued and attributes are shown". Thus, at this stage, only a small part of the power of Idemix is actually used on IRMA cards. Future upgrades may involve more functionality.

Next, the extensive use of all kinds of attributes within IRMA leads to *dependencies* between these attributes: attribute X can only be issued after attribute Y has been verified. As an example, before you can receive an attribute stating what your bank account or mobile phone number is, you need to authenticate properly. This authentication may involve a mixture of already issued attributes on your IRMA card (like your name and date of birth) and out-of-band communication (like a one-time SMS code). These dependencies lead to a tree structure for attributes. An interesting question then arises: what should be the "root" attributes that do not depend on any other attributes on the card. It turns out that this question has deep implications for the "identity fabric" in our society. For instance, one can image that your Facebook identifier is issued as an attribute, so that you can use your IRMA card as Facebook login. But should this Facebook attribute be root, or not? If it is a root, it cannot depend on any other attributes, and must be issued purely via out-of-band authentication. But if it is not a root, it will typically depend on your name and date of birth attributes; in that case one can no longer have a pseudonym Facebook account, enforcing Facebook's real name policy.

Finally, who should decide about such delicate issues? We foresee an independent, non-profit foundation that runs the IRMA scheme and sets such policy issues. Also, this foundation should do the certificate management that regulates access to the card, both for issuing and verification of credentials. Within the IRMA project there is close coordination with both public and private parties to openly discuss such issues. Ongoing work involves a (experimental) connection between existing government identity management infrastructures and IRMA cards, for the issuing of credentials based on government data.

The use of a wide variety of attributes leads to a new activity that we label as "credential design". It involves the organisation of individual attributes in signed containers (credentials), with dependencies between them. In the IRMA set-up a credential contains at most four (related) attributes, such as first name, family name, full names, initials. Our experience so far leads to the following principles of credential design.

1. Attributes in one credential form a coherent set.
2. Each attribute in one credential falls under the responsibiility of a single most authoritative issuer.
3. Attribute duplication (same content, multiple issuers) should be avoided.
4. Verifiers should only be able to read a limited, predefined set of attributes.
5. Credential dependencies should be public.
6. An independent non-profit scheme manager should decide about such dependencies.

References

1. Abendroth, J., Liagkou, V., Pyrgelis, A., Raptopoulos, C., Sabouri, A., Schlehahn, E., Stamatiou, Y., Zwingelberg, H.: D7.1 Application Description for Students. Technical report, ABC4Trust (2012)
2. Bcheri, S., Goetze, N., Orski, M., Zwingelberg, H.: D6.1 Application Description for the School Deployment. Technical report, ABC4Trust (2012)
3. Brands, S.A.: Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy. MIT Press, Cambridge (2000)
4. Camenisch, J., Krontiris, I., Lehmann, A., Neven, G., Paquin, C., Rannenberg, K., Zwingelberg, H.: D2.1 Architecture for Attribute-based Credential Technologies. Technical report, ABC4Trust (2011)
5. Camenisch, J., Lysyanskaya, A.: An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 93–118. Springer, Heidelberg (2001)
6. Camenisch, J., Lysyanskaya, A.: A Signature Scheme with Efficient Protocols. In: Cimato, S., Galdi, C., Persiano, G. (eds.) SCN 2002. LNCS, vol. 2576, pp. 268–289. Springer, Heidelberg (2003)
7. IBM Research Zürich Security Team. Specification of the Identity Mixer cryptographic library, version 2.3.4. Technical report. IBM Research, Zürich (February 2012)
8. Verheul, E.R.: Self-Blindable Credential Certificates from the Weil Pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 533–551. Springer, Heidelberg (2001)
9. Vullers, P., Alpár, G.: Efficient Selective Disclosure on Smart Cards Using Idemix. In: Fischer-Hübner, S., de Leeuw, E., Mitchell, C. (eds.) IDMAN 2013. IFIP AICT, vol. 396, pp. 53–67. Springer, Heidelberg (2013)

Data Protection by Default in Identity-Related Applications

Marit Hansen

Unabhängiges Landeszentrum für Datenschutz Schleswig-Holstein, Kiel, Germany
marit.hansen@privacyresearch.eu

Abstract. "Privacy by default" is being discussed as one important principle for ICT system design. This principle has been taken up as "data protection by default" in the proposal for a European Data Protection Regulation published in 2012. However, it is debated what this principle should mean in practice. In this text, we analyze the relation to "security by default" and "privacy by design" and discuss different possible interpretations of the "data protection by default" principle. After presenting general considerations on how to choose and implement appropriate default settings, we exemplarily describe recommendations for typical identity-related application scenarios such as social network sites, user tracking on the web and user-controlled management of one's identities. Both the general and the scenario-based elaborations provide guidance for developers as well as evaluators.

Keywords: Data Protection by Default, Privacy by Default, Privacy by Design, Security by Default, Identity, Identity Management.

1 Introduction

For several years, data protection authorities and other privacy advocates have been demanding "privacy by default", often in combination with "privacy by design" for conceptualizing, developing and operating ICT systems that are used to process personal data [1]. The general idea of "privacy by design" is to design technologies and business practices right from the beginning according to privacy criteria, thereby implementing important privacy guarantees and features in ICT systems and processes. "Privacy by default" addresses on the one hand that "privacy by design" should be a matter of course in ICT development and operation. On the other hand it defines the goal for a privacy-friendly standard configuration so that the usage of the ICT system does not infringe people's privacy. However, the exact interpretation of "privacy by default" is being debated.

The proposed European Data Protection Regulation [2] has taken up the principles of "privacy by design" and "privacy by default". Since the legal text focuses on "data protection" rather than "privacy" – entirely in the tradition of the European data protection framework in force –, it consequently has incorporated the principles under the names "data protection by design" and "data protection by default". Even without statutory "data protection by design and by default" on the European level, in some

S. Fischer-Hübner et al. (Eds.): IDMAN 2013, IFIP AICT 396, pp. 4–17, 2013.

European Member States national data protection law took up these principles at least partially, e.g. in the German telemedia law and the data protection law. In addition, they became criteria of the Schleswig-Holstein Privacy Seal [3] and the European Privacy Seal [4]. Standard settings are clearly relevant when assessing privacy properties of ICT products and services, because they will decide on the effort for users to apply the appropriate configuration for a privacy-compliant use. This is both relevant for users on behalf of an organization – they have to consider the legal requirements when processing personal data – and for end-users for their personal purposes, e.g., when joining a social network. The importance of defaults in ICT design has been explored by various researchers (e.g. [5] and [6]): They conclude that many users stick with the preconfigured settings which might put them unexpectedly at risk.

Not only European data protection authorities, but also the consumer organization BEUC regarded "privacy and security by design" and "privacy and security by default" as important principles that could be already derived from current law [7], in particular from Art. 17 of the European Data Protection Directive [8] that demands "appropriate technical and organisational measures" to protect personal data.

In this paper, we focus on "data protection/privacy by default". The following section describes how "data protection by default" is regarded in the proposed European Data Protection Regulation and what other definitions are being discussed. Section 3 presents general considerations for applying "data protection by default" in practice. In Section 4, recommendations for default settings are shown for three identity-related application scenarios. Section 5 summarizes the findings and gives an outlook.

2 Towards a Definition of "Data Protection by Default"

Starting point of this paper is the proposal for a European Data Protection Regulation [2]. In Art. 23 para. 2, "data protection by default" is introduced: "The controller shall implement mechanisms for ensuring that, by default, only those personal data are processed which are necessary for each specific purpose of the processing and are especially not collected or retained beyond the minimum necessary for those purposes, both in terms of the amount of the data and the time of their storage. In particular, those mechanisms shall ensure that by default personal data are not made accessible to an indefinite number of individuals." [2]

This should be read together with recital 61: "(61) The protection of the rights and freedoms of data subjects with regard to the processing of personal data require that appropriate technical and organisational measures are taken, both at the time of the design of the processing and at the time of the processing itself, to ensure that the requirements of this Regulation are met. In order to ensure and demonstrate compliance with this Regulation, the controller should adopt internal policies and implement appropriate measures, which meet in particular the principles of data protection by design and data protection by default." [2]

Firstly, it is noteworthy that rather than ICT developers this provision addresses the controller only, i.e., the entity that "determines the purposes, conditions and means of the processing of personal data" (Art. 4 para. 5 of the proposed Regulation [2]). Nevertheless, the preconfigured settings are often determined by the ICT developers. Of course, there might be an indirect effect on ICT developers if controllers demand

"data protection by default" [9]. However, it would be more appropriate to define an obligation for data controllers, data processors and producers of data processing systems as proposed in the Draft Report of the European Parliament [10].

Secondly, the meaning of "data protection by default" is not clear. The last sentence of Art. 23 para. 2 on accessibility "to an indefinite number of individuals" points to social network sites and other Internet services. Apart from that, the meaning seems to be reduced to the necessity principle (see also Art. 5 (c) of the proposed Regulation [2]): processing of personal data is only allowed if and as long the data are necessary for the purpose. Also, the European Data Protection Supervisor (EDPS) has demanded further clarification: "The principle of data protection by default aims at protecting the data subject in situations in which there might be a lack of understanding or control on the processing of their data, especially in a technological context. The idea behind the principle is that privacy intrusive features of a certain product or service are initially limited to what is necessary for the simple use of it. The data subject should in principle be left the choice to allow use of his or her personal data in a broader way. The EDPS recommends including in Article 23(2) a reference to this position of the data subject and providing the necessary clarification in recital 61." [9]

As a contrast, in her work on "privacy by design" Cavoukian does not refer to "least privacy intrusive", but regards "privacy by default" as "privacy as the default setting": "If an individual does nothing, their privacy still remains intact. No action is required on the part of the individual to protect their privacy – it is built into the system, by default." [11] This sounds good, but if, e.g., the purpose of the application is to disclose personal data to others (as for social networks), an absolute privacy protection cannot be achieved while keeping the functionality. In this case, the EDSP reference to a "simple use" with least privacy infringement probably fits better to the users' expectations.

It is widely agreed that a preconfigured setting does not constitute a user's consent – she might not even be aware of the setting. Also it is out of the question that a user should be able to change the configuration. However, it is debated whether "privacy by default" demands a preconfigured setting. Commissioner Reding, who criticized that "[p]rivacy settings often require considerable operational effort in order to be put in place" [12], focuses on lowering the threshold for users to configure the system according to their needs: privacy settings would have to be "designed to be easily found and manipulated by the user" (Reding according to [13] on "privacy by default"). In this interpretation, "privacy by default" would not need a preconfigured setting, but instead users could be forced to set the configuration at the install or first use of the system. The part of Reding's speech on "privacy by default" recognizes that the use of personal data for other purposes than specified could "be allowed with the explicit consent of the user or if another reason for lawful processing exists". This could mean that the default setting already foresees possible legitimate interests of the controller – other than maximizing the user's privacy – and can only be overridden by an explicit choice (opt-out) of the user. Frankly speaking, "opt-out" is not what users understand by "privacy by default".

In another way, the proposed Regulation falls short: the limitation to the necessity and data minimization principle. The Draft Report of the European Parliament extends that dividing "data protection by default" into two categories: the default by the controller when the data subject is given a choice, and the default of applying "data

protection by design" by data processors and producers to ensure that the privacy-compliant use by controllers [10]. For the latter case, that report relates it to all "principles relating to personal data processing" as introduced in a broadened version of Art. 5 of the proposed Regulation by explicitly mentioning transparency, purpose limitation, data minimization, integrity, storage minimization, intervenability, and accountability [10].

This seems to be a good approach to converge the discussion on "privacy by default". After all, the related – and often overlapping – concept "security by default", together with "security by design" or "built-in security", has been discussed for decades. The information security community has not agreed upon one single definition. But indisputably the well-known standard rule "deny by default" for firewalls is a good example: "firewalls should block all inbound and outbound traffic that has not been expressly permitted by the firewall policy" [14] – note that the default "deny all" means that no network traffic can pass, so even a "simple use" would not be possible, and the art of good firewall rules requires skilled engineers. Another possibility of "security by default" is realized by hardening ICT systems, i.e. all "services and features that are not widely needed should be disabled by default or accessible only to a small population of users" and "software should run with the least necessary privilege" [15]. Further, encryption as default, e.g. HTTPS as default for web browsing instead of making it an opt-in feature, is mentioned as an example for "security by default" [16].

Summarizing, "data protection by default" should encompass not only privacy by design as a default, but also least privacy-infringing (or maximally privacy-enhancing) default settings where this is reasonable. The question of defaults should address data controllers, processors and producers.

3 Applying "Data Protection by Default" in Practice

In ICT design, "data protection by default" addresses those cases where different configurations are possible, usually depending on the user's choices. For applications that cannot be configured differently, it is demanded anyway that they are legally compliant, and – if the "privacy by design" approach is followed – that they not only respect, but promote the data subject's privacy.

There is no universal answer to when configurable settings should be used and when wired-in functionality without an option to adapt should be preferred. Wired-in functionality may be regarded over-protective or even invasive for the autonomy and informational self-determination of the individual – both important core values of privacy protection. If configurable settings are chosen, their granularity has to be determined: On the one hand, fine-grained controls may be more appropriate to reflect any situation. On the other hand, they might be too complex for users to understand their meaning and the impact of modification. This would increase the risk of undesired consequences when changing the settings [17]. This usability issue is also raised by companies who fear that "privacy by default" could result "in software design that confuses and annoys customers with repeated notices and warnings." [18] However, this argumentation is inexplicable since "privacy by default" aims at the opposite.

Kesan and Shah [6] describe a framework for setting defaults, specifically for software, and the role of policy makers: In general, they don't recommend policy makers to intervene when developers choose the default settings – as long as they choose a setting that all parties would have agreed upon. However, they do see a need for interference of policy makers in three cases: "when users lack the knowledge and ability to change an important default setting" (this category may also comprise settings unexpected by the user), if the default settings "cause harm to third parties" or if it "does not comport with existing law and policy" [6]. In severe cases, they regard it as "necessary to change the setting from a default value to a wired-in setting" [6]. Especially in the area of privacy, one or more of the listed conditions for policy makers' intervention will often apply concerning today's standard software configuration and the lack of expertise and risk awareness of many users.

When discussing the right defaults, two different types of configuration should be distinguished:

1. The configuration of an additional process that is not strictly needed for the original functionality of the application or its simple use. This is usually associated with a new purpose (e.g. an additional subscription of a newsletter or additional data transfer to other parties that analyze the data). This additional process may be perceived as useful by the data subject, or not.
2. The configuration of a process necessary for the purpose within the application – here it has to be determined how the default setting should be. For instance, in a case where some data have to be transferred, it could be encrypted by default or not. Another example would be how the indubitable payment process for some goods or services is handled, e.g. via prepayment, credit card or direct debit.

The default for the first configuration type is quite clear: For each additional purpose, process or party getting access to personal data the default setting should be "No". Also for some original purposes like disclosing data to friends in social networks, meaningful privacy defaults could limit the risks.

This second configuration type needs more thorough elaboration: It often depends on the functionality that is supported at the user's side; it frequently needs the user's awareness or even consent. For example, the user requires transparency on the payment provider for organizing the money transfer, and usually wants to choose according to her own needs. Here a default would not be demanded. Also, it is not clear which payment method would suit the user best: prepayment, anonymous e-cash, or separated providers that keep financial data confidential and won't gain information on the purchased goods. Further, different methods may result in different costs for the user. Here transparency is needed, but not a one-size-fits-all default.

Concerning encrypted data transfer, thanks to the SSL support in all browsers this can and should be the default for web requests. But for e-mail encryption, this could only be a default if sender and receiver use the same crypto system. Note that some security or privacy functionality, like encryption, may cause performance losses. With today's machines, it is mostly negligible for SSL, so it does not constitute an obstacle against the default setting. Further, legal restrictions for specific countries have to be considered: Even if the privacy default should be a clear "Yes" for well implemented encryption, this may put the user – or the producer – at some legal risk.

In general, the desired privacy default might not be the same all over the world. Think of an application that allows the user to store data in a cloud. According to European data protection law, personal data from Europe shall not be transferred to locations where no adequate level of protection for the rights of the individual is guaranteed. Indeed, undesired access to data demanded by the local government is discussed as one of the major risks when using foreign cloud services. So, a software tool that provides different options for storage locations may have the default setting "Use the European cloud". But should this also apply to non-European users? Probably not – users and policy makers from other nations may prefer a cloud within their territory. In this case, the software could first – in a legally compliant way – analyze the geolocation of the user's IP address to find out whether she comes from Europe and then determine the default setting. However, the exact location of data processing and the place of jurisdiction can be very relevant for a user who wants to assert her data subject's rights, to know about additional risk (e.g. whether the location falls under a data retention regime) or to file a lawsuit.

The solution would be to foresee defaults like that, but to make sure that users notice them and can get more information on the setting, the reasoning behind and possibilities to change the default value. Of course, this would be necessary, too, if costs for the users differ depending on the chosen option. Here it might be better to do without a default, but be explicit on costs and conditions, especially if more security and better privacy protection cost more.

The information on the defaults should always be accessible, even if the user doesn't feel a need to change them. This is not only demanded by transparency requirements, but is also a good means to help users in understanding the system and be able to debate whether the preconfigured defaults are the right ones or should be adapted – not only for the individual, but for a bigger user group.

Not only location can constitute different desired defaults. A special case is the legal requirement for stricter protection of children data. Also, there might be user groups with individual needs, e.g. based on cultural differences or because of disabilities. However, determining the appropriate default is not a compelling legal reason for a comprehensive analysis of personal data at the controller's side. For client configurations at the user's side, it is possible that the user can choose at install or first use between various categories to get the most appropriate defaults, but this must not result in a transfer of potentially sensitive data (e.g. on disabilities).

In addition, there might be "configuration providers" offering the most appropriate default for their respective community, e.g. as a downloadable file or a newsfeed with updates. This may be a data protection authority, a consumer organization, an association of people with disabilities, a church, a club or other interested users. These configuration providers could support users if a reaction on a security breach is necessary ("from now on no further data transfer to XY", "abstain from broken crypto algorithm XZ"). By this, more flexibility in shorter reaction time could be achieved. Note that, by providing this default setting, the responsibility and liability for at least that part of the configuration is shifted from the producer/controller/processor to this configuration provider.

From the previous discussion and the legal data protection framework, criteria for assessing potential default settings are sketched in form of a flowchart in Fig. 1.

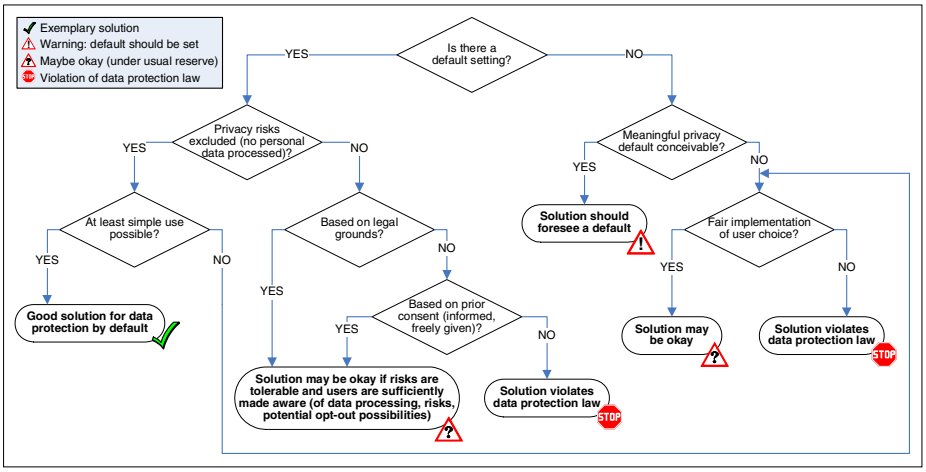


Fig. 1. Assessing potential default settings for user choices in applications

Clearly, a default setting that excludes any risks and enables an at least simple use of the system is a good solution for data protection by default. However, exclusion of any privacy risk may lead to a severe restriction of functionality to that even a simple use is not possible. In some cases, this can be the best choice to guarantee that users become aware of configuration options – it is comparable to the "deny all" firewall rule that forces the users to think about what they really need.

However, there may be another solution that does not rule out any privacy risk, but is based on legal grounds. Hopefully, these legal grounds are not overly privacy-infringing themselves (think of the data retention directive). An example is the legal provision on pseudonymous profiling of users under strict conditions as foreseen in the German Telemedia Law and in the Draft Report of the Parliament [10]. In this case, the law has defined that users have to be notified and can opt-out, but the default setting may be "Yes" for pseudonymous profiling, although this would not be the least privacy-infringing solution. So it is debatable whether this legally compliant solution still counts as "data protection by default". The reasoning behind privileging pseudonymous data processing is that the purpose can be achieved by accepting a reduced risk compared with a detailed analysis under real names, no matter whether the otherwise legally demanded consent is collected or not. In general, a risk may be tolerable if it is known to the data subject and no overriding legitimate interests of the data subject would be adversely affected. The same line of thought can be applied if the default relies on a consent the user has given earlier.

As stated above, there are many situations where a meaningful privacy default cannot be established. In these cases, a fair implementation of user information and choice is necessary. Otherwise it is questionable whether a solution can be lawful.

4 Identity-Related Application Scenarios and "Data Protection by Default"

In this section, three examples for identity-related applications and potential defaults are discussed: social network sites, user tracking on the web and user-controlled identity management systems.

4.1 Social Network Sites

Although the discussion on "data protection by default" is much older, with social network sites it became apparent to huge parts of the society how important some privacy features are and what may happen if private information is widely distributed. One example is the problem with Facebook parties where users have accidentally invited thousands of members of the network instead of their small group of real friends. As already written in Section 2, the proposed European Data Protection Regulation explicitly demands mechanisms that prevent by default that personal data are made accessible to an indefinite number of individuals. [2]

Various research teams have investigated the value of privacy settings in social networks such as Facebook, e.g. [19], [20] and [21]. They found that, in large part, the privacy settings didn't match the users' expectations who typically expected a higher degree of privacy and less visibility of their personal data to others.

The understanding of privacy settings could be improved by an appropriate choice of words. For instance, a Facebook "friend" is not always a friend, and when it comes to friends of friends, this may sound nice and still quite intimate, but there is no reliable estimation of users who belongs to this category. For research purposes in [20], the Facebook category "everyone" for configuring the accessibility was renamed to "stranger". By this, the scope of potential access can be better expressed.

The question of "data protection by default" in practice for social network sites has been tackled by several researchers. In the work of [22], "privacy by design" and "privacy by default" have been analyzed separately. Whereas, among others, the aspect of allowing anonymous or pseudonymous use in social networks is considered a "privacy by design" requirement, the following collection of recommendations and considerations are categorized as "privacy by default" [22]:

- In the default setting, only the basic functionalities should be provided, i.e. only those personal data should be collected and used that are necessary for employing a basic service as expected by the users. This also enables users to become familiar with the system.
- Additional privacy-related functionality needs explicit consent by users. For instance, external search engines should not be able to find a user or to access her data unless she opts in. The biometric analysis of photos would need explicit information of the data subject and her consent.
- The default setting for the visibility of profile or status data should be limited to the user's circle of close friends. There might be an exception for social networks that are dedicated to exchanging business contacts.

- If minors are member of the social network, the default settings have to be especially strict.
- Even with strict default settings a usage of the service should be possible. Otherwise there is a risk that users activate everything they can to overcome the limitations of the default settings.
- The provider has to inform the users about defaults and options in a comprehensible way in the registration process. They should be able to conveniently access and check their setting. The information should be well-arranged and easy to understand. Settings should be changeable at any time with effect for the future.

It should go without saying that personal data of non-members of a social network should not be collected unless they have given consent.

4.2 User Tracking on the Web

As soon as commercial sites became part of the World Wide Web, the tracking of users started – based on analysis of IP addresses, the referrer information or other browser chatter and cookies of different kinds. This information is being used especially for marketing research and for adjusting online advertisement according to the usage profile of the web surfers (behavioral targeting). Not always the data are matched to individual users, but at least the collected raw data has to be considered personal data. Most web browsers offer some possibilities for configuration, and several users install plug-ins that help them filtering tracking functions or advertisements. Still, the data collection by many web sites, directly or via third parties, doesn't comply with the requirements set out in the European e-Privacy Directive [23] or Member State's law.

Since 2007, consumer organizations in the U.S. have uttered the need for some "Do Not Track" solutions. In this situation, the World Wide Web Consortium took up a proposal of a "Do Not Track" (DNT) header that could be integrated in web browsers. Three different values of DNT are possible: "1" for "opt-out", meaning that the user does not want to be tracked; "0" for "opt-in", meaning that user consents to being tracked; and "null" (no header sent) if the user has not expressed a preference [24].

When Microsoft announced that their Internet Explorer 10 would be rolled out with default setting "1" (i.e. no tracking), several stakeholders in the U.S. regarded that as an affront since Microsoft, and not the user, would be exercising choice on the browser signal. They threatened to ignore all "no tracking" values sent from Internet Explorers because these settings would represent a choice made by a company instead of the user herself.

However, the European Commission has clarified that "it is not the Commission's understanding that user agents' factory or default setting necessarily determine or distort owner choice. The specification need not therefore seek to determine the factory setting and should not do so, because to intervene on this point could distort the market. [...] the standard should foresee that at the install or first use of the browser the owner should be informed of the importance of their DNT choice, told of the default setting and prompted or allowed to change that setting." [25] Note that the Commission's statement does not judge on which default the factory setting should be preferred – this is left to the discretion of the browser producers.

This fight over the default setting for behavioral advertizing has an economic background. Even when surveys have shown that the majority of U.S. users as well as European users don't want to be tracked, the huge advertising industry strongly opposes the idea of a non-tracking default. It challenges the business model of offering services "for free" in exchange for collecting and analyzing personal data for individually targeted advertisements. So it may boil down to the question who will pay for web services and what privacy-respecting business models may work.

4.3 User-Controlled Managing of One's Identities

User-controlled identity management systems assist and empower users to manage their own partial identities in their digital lives as well as their privacy [26]. They act as gateways and guardians between the user and her communication partners. It is not easy to define the right defaults in such concepts, because active privacy management by the users themselves require their understanding and control while the "data protection by default" principle mainly addresses those users who are not aware of the risks and are not willing or able to configure their system according to their needs. Of course, user-controlled identity management systems would have to inform and train users, but should not generally follow a too paternalistic approach which may be the case when deciding automatically on behalf of the users.

Still, a few guidelines can be given here, too:

- When user-controlled identity management systems are employed for achieving unlinkability between different partial identities, this has to be supported throughout all network layers: For instance, if linkable device IDs, IP addresses or browser chatter are communicated, the protection against linkage that private credentials can offer is void. Also, additional data hidden in files may jeopardize the protection. So metadata in documents or photos should not be disclosed to others unless explicitly chosen by the user. Note that this setting relies on additional privacy-by-design functionality.
- Naturally, all communication should be encrypted by default. Further, the storage of personal data under control of the user herself should be done in a highly secure and trustworthy area to protect unauthorized access.
- As design feature, warnings should be displayed by default if the personal data are to leave the area where the legal data protection framework relevant to the user (e.g. the European Economic Area for European citizens) can be enforced. This may be extended in cases of security or data protection breaches.
- By default, each new contact would mean to create and use a new pseudonym (representing another partial identity) if not stated otherwise. For existing contacts, a re-use of the already established pseudonym should be the default unless the user demands a new one. Note that this violates the rule of maximizing unlinkability for best privacy protection due to presumed functionality requirements: In user-controlled identity management systems, it should be the (changeable) default to enable longer lasting relationships with a communication partner instead of being limited to an anonymous use or one-time contacts.

- Only those attribute values that are necessary for the purpose should be released. The identity management system cannot decide on its own which attributes are needed. So here a fully automatic transfer of personal data without explicit and informed user consent is not advised. However, this process may be supported by certificates that communication partners (the "relying parties") can get from a trusted party after having assessed the necessity for the attributes for the specified purpose. For the German eID card, this is done to allow selective disclosure of attributes such as information about the age or the domicile, and to distinguish between usage under pseudonym or by giving the real name [27].

For attribute-based credentials that provide a more general solution on possible attributes that may be selectively disclosed meaningful and privacy-friendly defaults still have to be elaborated [28]. Also, the integration of additional parties that take part in the issuing or revoking process or may be relevant if an investigation is needed has to be considered when discussing privacy by default.

5 Conclusion

The principle "data protection by default" is part of the "privacy by design" approach. Although both "data protection by default" and "by design" have been incorporated as a legal requirement in the proposed European Data Protection Regulation [2], they are still not well defined. The Draft Report of the European Parliament on the Regulation [10] provides some clarification. But even this improved version of a legal proposal does not answer all questions for an application in practice.

Looking at various interpretations of the "data protection by default" principle, we conclude that two major types of configuration should be distinguished: the introduction of functionality with additional purposes on top of the basic system, and the setting of different options for components within the core application context. Whereas all additional functionality, that is not needed for the original purpose but can involve privacy risks, should be denied by default, the form of the preconfigured setting for mandatory functionality is not always clear when different options exist. However, such situations often require the user's awareness of the data processing and related risks so that a seamless interaction is not advised anyway. It is recommended that users get the necessary information to compare settings regarding privacy risks and other relevant properties such as costs or required functionality. In many cases, a fully automatic decision on which defaults are the best for privacy cannot work since there is no overall accepted privacy metrics. Also, even privacy-aware users may have different preferences. In some cases localized defaults should be preferred over global settings, e.g. if the jurisdiction of processors plays a role. For those who don't want to delve into tailoring their settings, configuration providers can step in and offer meaningful defaults fitting various user groups.

An open question is whether "data protection by default" should try to maximize the privacy level when other kinds of data processing are privileged in data protection law. This is especially important for pseudonymous data processing that can massively reduce the risks to privacy in comparison to working with personal data on the

basis of consent. Here "data protection by default" may mean "legally compliant by default" instead of "maximum privacy". In these cases, transparency on the defaults and ways for users to change them are essential.

Note that it is not sufficient to have good default settings if users are later on being tricked into releasing data or activating services. Experiments have shown that already the way in which individuals are asked for a decision concerning their privacy might influence their choice [29] – such psychological effects have to be further investigated. Supervisory authorities should demand neutral and fair information of users – otherwise a given consent might not be regarded as valid.

Whereas the default settings in the presented scenarios are important, but usually can be changed with some effort by the users, the choice of "data protection by default" becomes even more crucial when the personal computer as interface for looking at settings and changing them vanishes. With tablet computers and mobile phones, the convenience for reconfiguring the system already decreases. Similarly, Smart TVs that are based on web protocols appear to have similar privacy risks as browsing the Internet, but less configuration possibilities. Finally, for ubiquitous computing only few proposals exist on how "privacy by default" could be implemented [30].

All in all, the discussion on defaults should not promote that users lack necessary information and simply rely on the assumption that the best choice for them has been made already. This would not reduce, but rather increase the vulnerability of individuals' privacy.

Acknowledgments. I am grateful for the support of Jozef Vyskoč and Simone Fischer-Hübner throughout various versions of this text.

References

1. 32nd International Conference of Data Protection and Privacy Commissioners: Privacy by Design Resolution. Proposed by Cavoukian, A., approved in October 2010, Jerusalem, Israel (2010), http://www.ipc.on.ca/site_documents/pbd-resolution.pdf
2. European Commission: Proposal for a Regulation of the European Parliament and of the Council on the protection of individuals with regard to the processing of personal data and on the free movement of such data (General Data Protection Regulation). COM (2012) 11 final, Brussels, January 25(2012), http://ec.europa.eu/justice/data-protection/document/review2012/com_2012_11_en.pdf
3. Hansen, M., Probst, T.: Datenschutzgütesiegel aus technischer Sicht: Bewertungskriterien des schleswig-holsteinischen Datenschutzgütesiegels. In: Bäumler, H., von Mutius, A. (eds.) Datenschutz als Wettbewerbsvorteil – Privacy sells: Mit modernen Datenschutzkomponenten Erfolg beim Kunden, pp. 163–179. Vieweg, Wiesbaden (2002)
4. European Privacy Seal, https://www.european-privacy-seal.eu/
5. Nielsen, J.: The Power of Defaults. Jakob Nielsen's Alertbox (September 26, 2005), http://www.useit.com/alertbox/defaults.html
6. Kesan, J.P., Shah, R.C.: Setting Software Defaults: Perspectives from Law, Computer Science and Behavioral Economics. U Illinois Law & Economics Research Paper No. LE06-012. Notre Dame Law Review 82, 583–634 (2006)

7. Bureau Européen des Unions de Consommateurs (BEUC): EU General Data Protection Framework – BEUC answer to the consultation (December 31, 2009), http://www.beuc.org/custom/2010-00021-01-E.pdf
8. Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data. Official Journal L 281, 0031–0050 (November 23, 1995)
9. European Data Protection Supervisor: Opinion of the European Data Protection Supervisor on the data protection reform package (March 7, 2012), http://www.edps.europa.eu/EDPSWEB/webdav/site/mySite/shared/Documents/Consultation/Opinions/2012/12-03-07_EDPS_Reform_package_EN.pdf
10. Albrecht, J.P.: Draft Report on the proposal for a regulation of the European Parliament and of the Council on the protection of individual with regard to the processing of personal data and on the free movement of such data (General Data Protection Regulation) (COM (2012)0011 – C7-0025/2012 – 2012/0011(COD)), Committee on Civil Liberties, Justice and Home Affairs (Decmber 17, 2012), http://www.europarl.europa.eu/sides/getDoc.do?language=EN&reference=PE501.927
11. Cavoukian, A.: Privacy by Design: The 7 Foundational Principles (August 2009), http://www.privacybydesign.ca/content/uploads/2009/08/7foundationalprinciples.pdf (revised January 2011)
12. Reding, V.: Your data, your rights: Safeguarding your privacy in a connected world. Privacy Platform The Review of the EU Data Protection Framework, Brussels (March 16, 2011), Reference: SPEECH/11/183, http://europa.eu/rapid/press-release_SPEECH-11-183_en.htm
13. Altheim, M.: The Review of the EU Data Protection Framework v. The State of Online Consumer Privacy in the US. Blog entry (March 17, 2011), http://ediscoverymap.com/2011/03/the-review-of-the-eu-data-protection-framework-v-the-state-of-online-consumer-privacy-in-the-us/
14. Scarfone, K., Hoffman, P.: Guidelines on Firewalls and Firewall Policy. Recommendations of the National Institute of Standards and Technology. Special Publication 800-41, Revision 1 (September 2009), http://csrc.nist.gov/publications/nistpubs/800-41-Rev1/sp800-41-rev1.pdf
15. Lipner, S., Howard, M.: The Trustworthy Computing Security Development Lifecycle. MSDN, Security Engineering and Communications, Security Business and Technology Unit, Microsoft Corporation (March 2005), http://msdn.microsoft.com/en-us/library/ms995349.aspx
16. Soghoian, C.: Not an option: time for companies to embrace security by default. Ars Technica. (August 9, 2011), http://arstechnica.com/tech-policy/2011/08/not-an-option-time-for-companies-to-embrace-security-by-default/
17. Iachello, G., Hong, J.: End-User Privacy in Human-Computer Interaction. Found. Trends Hum.-Comput. Interact. 1(1), 1–137 (2007)
18. Microsoft: Privacy by Default (March 2012), http://download.microsoft.com/download/B/8/2/B8282D75-433C-4B7E-B0A0-FFA413E20060/privacy_by_default.pdf

19. Liu, Y., Gummadi, K.P., Krishnamurthy, B., Mislove, A.: Analyzing Facebook Privacy Settings: User Expectations vs. Reality. In: Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference, IMC 2011, pp. 61–70. ACM, New York (2011)
20. Madejski, M., Johnson, M., Bellovin, S.M.: The Failure of Online Social Network Privacy Settings. Tech Report CUCS-010-11, Columbia University (2011)
21. Rubinstein, I.S., Good, N.: Privacy by Design: A Counterfactual Analysis of Google and Facebook Privacy Incidents. New York University Public Law and Legal Theory Working Papers, Paper 347 (2012), http://lsr.nellco.org/nyu_plltwp/347
22. Niemann, F., Scholz, P.: Privacy by Design und Privacy by Default – Wege zu einem funktionierenden Datenschutz in Sozialen Netzwerken. In: Peters, F., Kersten, H., Wolfenstetter, K.-D. (eds.) Innovativer Datenschutz, pp. 109–145. Duncker & Humblot, Berlin (2012)
23. Directive 2002/58/EC of the European Parliament and of the Council of 12 July 2002 concerning the processing of personal data and the protection of privacy in the electronic communications sector (Directive on privacy and electronic communications), Official Journal L 201, 0037–0047 (July 31, 2009); amended in 2009 by the Directive 2009/136/EC (November 25, 2009)
24. Fielding, R.T., Singer, D. (eds.): Tracking Preference Expression (DNT). W3C Working Draft 02 October 2012 (2012), http://www.w3.org/TR/tracking-dnt/
25. European Commission, Director-General (Robert Madelin): Letter to World Wide Web Consortium Tracking Protection Working Group. Ref. Ares (2012)743354 (June 21, 2012), http://lists.w3.org/Archives/Public/public-tracking/2012Jun/att-0604/Letter_to_W3C_Tracking_Protection_Working_Group.210612.pdf
26. Hansen, M.: User-controlled identity management: the key to the future of privacy? International Journal of Intellectual Property Management (IJIPM) 2(4), 325–344 (2008)
27. Zwingelberg, H., Hansen, M.: Privacy Protection Goals and Their Implications for eID Systems. In: Camenisch, J., Crispo, B., Fischer-Hübner, S., Leenes, R., Russello, G. (eds.) Privacy and Identity 2011. IFIP AICT, vol. 375, pp. 245–260. Springer, Heidelberg (2012)
28. ABC4Trust – Attribute-based Credentials for Trust, FP7 ICT Integrated Project, https://abc4trust.eu/
29. Acquisti, A., John, L., Loewenstein, G.: What is privacy worth? In: Workshop on Information Systems and Economics, WISE (2009), http://www.heinz.cmu.edu/~acquisti/papers/acquisti-ISR-worth.pdf
30. Schmitt, L.: Privacy as default. Privacy by default! Konzept für Privatsphäre im Ubiquitous Computing. Diploma Thesis, Köln International School of Design (June 2006), http://lutzschmitt.com/pub/Lutz_Schmitt-Privacy_as_default_Privacy_by_default.pdf

Privacy-Friendly Checking of Remote Token Blacklists

Roel Peeters and Andreas Pashalidis

KU LEUVEN, ESAT/SCD - COSIC & iMinds
Kasteelpark Arenberg 10 bus 2446, 3001 HEVERLEE, BELGIUM
firstname.lastname@esat.kuleuven.be

Abstract. Consulting a remote blacklist as part of verifying a token should not come at the cost of privacy. In particular, the blacklist provider should be unable to identify which tokens are being verified. The contents of the blacklist should also be protected; that is, it should not be possible to learn the contents of the blacklist, for example by querying the blacklist provider a large number of times. This paper defines a range of desirable properties for privacy preserving blacklist checking protocols, and surveys existing technical solutions to this problem. We propose adaptations where appropriate, and provide concrete performance estimates for the use case of checking whether or not a passport has been reported lost or stolen.

1 Introduction

As part of verifying a token it is sometimes necessary to check with a remote authority, in an online fashion, whether or not the token has been blacklisted. 'Transport layer security' (TLS) clients such as browsers, for example, can be configured to query a remote 'online certificate status protocol' (OCSP) server as part of verifying a server certificate. This query contains the serial number of the encountered certificate, and the OCSP server's response indicates whether or not the corresponding certificate has been revoked. Similarly, as part of verifying an official document such as an identity card or a passport, inspection systems sometimes issue a query to remote authorities that maintain document blacklists. This query, too, contains the serial number of the document and the response indicates whether or not the document has been blacklisted.

Currently deployed systems for blacklist checking reveal the identity of the token to the remote authority. This situation is unfortunate, because it undermines the privacy of the token owner. In the OSCP setting, for example, the server gets to know with whom the verifier is about to communicate (namely the owner of the token), and in the official document verification scenario, the remote database authority gets to know whose documents are being verified. Since inspection systems are typically located at well-known locations such as border crossings and airports, this reveals citizens' travel patterns.

A more privacy-friendly approach to revocation checking involves pushing the entire blacklist to every verifier. In this way, verifiers can perform the blacklist check locally, without consulting any remote authority. In fact, 'certificate

S. Fischer-Hübner et al.(Eds.): IDMAN 2013, IFIP AICT 396, pp. 18–33, 2013.

revocation lists' (CRLs) follow exactly this approach. However, CRL-like solutions are often unacceptable because the blacklist is itself sensitive and must not be disclosed to verifiers. Otherwise, criminals with illegitimate access to an inspection system, for example, will be able to check, without risking detection, whether or not any stolen passports have already been blacklisted. Currently, Interpol offers itself an online query interface to its 'stolen or lost travel document' (SLTD) database, or may arrange for a copy of the database to be pushed to a national server. The database, is, however, never pushed to individual inspection systems[1].

The remainder of this paper is structured as follows. The next section defines our model of remote blacklist systems and desirable properties. Section 3 describes a simple solution that achieves some of the properties, and Sect. 4 surveys the literature in search of schemes that outperform the simple solution. Section 5 describes our proposal, which is an adaptation of an existing techniques. Finally, Sect. 6 concludes.

2 Model and Desirable Properties

This section describes our abstract model for blacklist systems. First, we describe our system model. Next, the definitions of the desirable properties are given. Finally, we discuss the adversary model.

2.1 System Model

A remote blacklist system consists of three types of players, namely tokens, verifiers, and a blacklist provider. It is assumed that each token is assigned a unique identifier from the universe of token identifiers $\mathcal{T}$, and is issued to a user. In our system description, we assume that a single verifier exists in the system, and denote it by V.[2] The remote blacklist provider is denoted by BP and has a collection of blacklists $B_1, B_2, \ldots$, with each blacklist $B_v = \{\tau_1, \tau_2, \ldots, \tau_{|B_v|}\} \subseteq \mathcal{T}$ containing $|B_v|$ distinct token identifiers, where v denotes a blacklist's version number. The way in which BP constructs the blacklist collection is outside the scope of this paper. It is assumed that BP and V communicate over a secure channel.

A remote blacklist system also defines two protocols, namely Init and Query. The Init protocol is executed between BP and V, where BP's input is B_v. It is executed at least once, and may be executed regularly in fixed intervals or on demand. Without loss of generality, we assume that, when Init is executed for the vth time, BP's input is B_v (for all $v \in \{1, 2, \ldots\}$).

The Query protocol is executed whenever V wishes to verify a token, and involves all three parties. In some systems, the token has a bidirectional communication interface (e.g. eID documents, smartphones), while in others this

[1] `https://www.interpol.int/Public/FindAndMind/default.asp`

[2] Sections 4.2 and 5 consider the setting of multiple verifiers.

interface is unidirectional (e.g. the machine readable zone of a document, a public key certificate). Tokens with a bidirectional interface may play an active role in the protocol, i.e. react to incoming messages, while tokens with a unidirectional channel are typically passive. That is, V simply reads information from the token (e.g. its identifier τ) and uses it during the protocol. BP's input to the protocol is B_v for a given version v, and V's input is τ, the identifier of the token. At the end of the protocol execution, V learns whether or not $\tau \in B_v$.

2.2 Desirable Properties

A remote blacklist system must enable V to obtain the required information. That is, it must satisfy *correctness*: at the end of a Query protocol execution where BP's input is B_v and V's aim is to check the status of the token with identifier τ, V learns whether or not $\tau \in B_v$. In the following, we list further desirable properties for a remote blacklist system.

- **T-User Privacy.** Given a well-defined subset of token identifiers $T \subseteq \mathcal{T}$, for each Query protocol execution BP learns nothing beyond whether or not $\tau \in T$. Note that T must be fixed before the Query protocol execution starts. Also, if $T = \mathcal{T}$, then the strongest possible privacy notion is achieved as BP essentially learns nothing from Query protocol executions.
- **Weak T-User Privacy.** The definition of T-User Privacy is relaxed such that BP may also learn τ if $\tau \in T$.
- **Marginal Blacklist Hiding** For each Init protocol execution, V learns nothing about $B_1, \ldots, B_v$ beyond the fact that the blacklist is of length at most $|B_v| + \mu_v$ for some well-defined margin $\mu_v \in \mathbb{Z}^*$, and, for each Query protocol execution, V learns nothing about $B_1, \ldots, B_v$ other than (a) whether or not $\tau \in B_v$, and (b) the upper bound for the blacklist size, as described above.
- **Efficiency.** The Init and Query protocols must have computation and communication complexity that scales well in $|B_v|$; in particular, for the Query protocol, a constant complexity is desirable.
- **Token Binding.** We define multiple variants of this property, as follows.
  - **Reactive -** V must be able to convince an independent auditor that it executed the Query protocol only for tokens that were actually involved in these protocol executions.
  - **Weak Proactive -** V must not be able to execute the Query protocol with BP for a token that was not involved in a Query protocol execution.
  - **Strong Proactive -** V must not be able to execute the Query protocol with BP for a token that is not involved in the *current* protocol execution. This property requires the token to be active, i.e. to respond to incoming messages.
- **Online operation.** BP can authorise each incoming query to a specific version of the blacklist. This is achieved if BP's participation is required in the Query protocol. This property prevents a compromised verifier from executing the Query protocol a large number of times without risking detection.

Strong proactive token binding implies weak proactive token binding, but proactive token binding does not imply reactive token binding (and vice versa). Moreover, in some applications, e.g. certificate revocation checking for websites, a zero margin may be acceptable if disclosing the exact value of the size of the B_v is not a problem. In other applications, such as the list of lost or stolen passports, a positive margin is required as the exact value of $|B_v|$ must remain hidden. Otherwise, an adversary that can execute the system's protocols with BP and that recently stole a moderate number of passports can easily deduce the point in time at which these passports appear in the blacklist. ‘

In the use case of passport revocation checks, we expect a realistic blacklist size to contain several million entries. Despite such a lengthy blacklist, and in order to avoid accumulating queues in front of border control guards at a busy airport, a Query protocol execution must complete within a few seconds. Moreover, token binding is particularly desirable in the context of passport inspection, because it contributes to fraud detection and auditing towards data protection compliance. A solution that supports strong proactive token binding cannot, however, fully replace solutions without this property. This is because there will always be passports without a chip or with a broken chip that nevertheless will need to be checked against the blacklist. Interestingly, a scheme that provides strong token binding, or a scheme with $\mathcal{T}$-user privacy where the passport chip plays a critical role, would provide an incentive for citizens *not* to destroy their passport chips for privacy reasons.

2.3 Adversarial Model

The blacklist provider can always refuse to run the query against the blacklist. We assume that BP has an interest to prevent the acceptance of blacklisted tokens and that it will therefore not deny its service, unless it suspects that the verifier is compromised. However, while BP is assumed to be honest in this respect, it may attempt to determine the identities of tokens that are not truly blacklisted, for example by running incoming queries against hidden ‘shadow’ blacklists in parallel to the real blacklist. Such parallel and invisible query evaluations should be prevented. In this sense, we consider an honest-but-curious BP. Note that systems that provide $\mathcal{T}$-user privacy or (weak) B_v-user privacy automatically also provide protection against verifiers that maintain shadow blacklists.

V may also misbehave, for example by providing a token identifier that does not correspond to a token that is currently under inspection. Since V's goal is to verify the legitimacy of tokens, we consider such behaviour to be outside V's interest. V may, however, become compromised and then serve as a vehicle for the adversary to check the status of stolen tokens, or to otherwise disrupt the system by issuing superfluous queries. The different variants of token binding limit the damage caused by compromised verifiers: while reactive token binding enable detection of compromised verifiers, systems that provide proactive token binding ensure that compromised verifiers cannot query BP about tokens that are not present. Moreover, the property of online operation ensures that BP can

keep a record of how many queries are issued by each verifier, and can hence trigger an alarm if some verifier issues an abnormal amount of queries.

3 A Simple Solution

This section describes a simple solution to the above problem. This solution makes use of a technique called 'oblivious polynomial evaluation', and essentially adapts the protocol proposed by Freedman *et al.* [8] to our setting.

The Init protocol proceeds as follows. First, BP constructs a random polynomial $f(\cdot)$ of degree $|B_v|$, the roots of which correspond to the elements on the blacklist. Using a homomorphic encryption scheme, each coefficient of the polynomial is then encrypted using BP's public key, and the resulting ciphertexts are sent to V. V stores these ciphertexts.

In order to execute the Query protocol for token identifier τ, V obliviously evaluates the encrypted polynomial $f(\cdot)$ at τ. This is done in ciphertext space and is possible due to the homomorphic properties of the encryption scheme. The result is then multiplied with a random number r and is sent to BP. By decrypting the received value, BP obtains the value $rf(\tau)$. If τ is a root of the polynomial, then $rf(\tau) = 0$ and BP learns that $\tau \in B_v$. Otherwise, the decryption yields a random number and hence does not reveal anything about τ beyond the fact that $\tau \notin B_v$. In both cases, BP returns a bit to V that indicates whether or not $\tau \in B_v$.

Table 1 lists the security and efficiency properties achieved by the simple scheme described above. For the efficiency evaluation, it is assumed that, during a Query protocol run, V uses the hashing-to-bins method as described in [8]. We stress that, in this scheme, the complexity of the Init protocol is amortized over a potentially large number of Query protocol executions.

Table 1. Properties fulfilled by simple scheme

Property	Comment			
T-User privacy	Yes, for $T = B_v$			
Blacklist hiding	Yes (zero-margin), if BP uses a fresh key for every version v.			
Token binding	No			
Online operation	Yes			
Efficiency	Init		Query	
	BP	V	BP	V
(computation)	$\mathcal{O}(\lvert B_v\rvert)$	-	$\mathcal{O}(1)$	$\mathcal{O}(\ln\ln\lvert B_v\rvert)$
(communication)	$\mathcal{O}(\lvert B_v\rvert)$		$\mathcal{O}(\lvert B_v\rvert)$	

Note that the original protocol as described in [8] requires V to return an encryption of $rf(\tau) + \tau$ (rather than an encryption of $rf(\tau)$) to BP; since this enables BP to pinpoint the identity of the blacklisted token if it appears on the blacklist, this protocol variant only provides weak B_v-user privacy. Moreover, while the original protocol uses Pallier encryption, in the modified version

described above it is possible to further improve computation efficiency using exponential ElGamal. This variant of ElGamal does allow decryption of ciphertexts, but one can test if a ciphertext corresponds to a given plaintext, and this suffices in our setting. However, this is only an improvement in the constant factor, the asymptotic complexity remains the same.

This simple scheme suffers from two major shortcomings: (a) $\mathcal{O}(|B_v|)$ elements need to be tranferred for each Query protocol, and (b) the scheme does not provide any form of token binding.

4 Survey

In the literature, systems that simultaneously address user privacy and blacklist hiding are, depending on their exact properties, said to solve the problem of 'private disjointness testing' (PDT), 'private set intersection' (PSI), 'private set intersection cardinality' (PSI-CA), or 'authorised private set intersection' (APSI). Note these problems are more general than ours: they focus on the case where V queries multiple identifiers in a single protocol execution, whereas the Query protocol as defined in Sect. 2.1 requires V's input to be a single identifier.

In most of the works we discuss below, the 'client', not the 'server', obtains the result of the protocol. While at first glance it may appear more natural for V to assume the role of the client in our setting, we sometimes reverse the roles of the two parties such that BP obtains the result instead. This role reversal yields a significant efficiency advantage because multiple Query protocol executions can be performed after a single Init protocol execution. That is, the effort of Init is amortized over a potentially large number of queries. Note that systems that provide (weak) B_v-user privacy, i.e. systems where BP learns the outcome of the Query protocol (and informs V in a subsequent message), are likely to be considered sufficiently privacy-friendly with respect to users, because only a small number of tokens will be blacklisted. It is important, however, to ensure that V is unable to run queries against shadow lists.

4.1 Schemes without Proactive Token Binding

This section surveys related work on protocols that do not support token binding.

Oblivious Polynomial Evaluation. We now briefly review certain schemes that build on [8], and show that, while they offer advantages in the generic PSI and PSI-CA setting, in our case where V 's input to the Query protocol is a single element, they essentially degenerate to the simple solution described above.

- Kiayias and Mitrofanova [13] proposed a variant that uses superposed encryption which offers advantages when V's set is large. However, the usage of superposed encryption offers no advantages in our setting and hence can be omitted.

- Hohenberger and Weis [9] propose a variant that is secure against a dishonest V being able to illegitimately convince BP that the intersection is not empty[3]. However, in our setting it is in V's interest to provide the correct input to the protocol, and hence this type of protection is not required. Note that, solutions that provide some form of token binding must provide security against dishonest verifiers; however, token binding is outside the scope of [9].
- The solution of Ye *et al.* [20, 21] is based on Sylvester matrices. For both datasets, the data serves as the roots of a polynomial. From these two polynomials the Sylvester matrix is constructed. The determinant of this matrix indicates whether or not the two sets intersect. The privacy is protected by encrypting the polynomials with an additive homomorphic encryption scheme, such as Paillier's[4]. To avoid that one could also learn the cardinality of the set intersection, the determinant is evaluated in a secure two-party computation. However, since the set of the verifier only contains a single element, this is not an issue and we can use the protocol as being sketched in the intuition section of [20, 21].

In our setting, the asymptotic communication and computation complexity of the above schemes are identical to the complexity of the simple scheme.

OT and PIR-Based Techniques. Schemes that do not use the idea of polynomial evaluation, such as 'private information retrieval' (PIR), 'symmetric PIR' (SPIR), and 'oblivious transfer' (OT) schemes, can also be used for privacy-friendly checking of remote blacklists. PIR schemes enable a client to retrieve some data items from a database server, without the server learning which items are retrieved. SPIR schemes also protect the privacy of the database, in that the client can only learn a single data item whose index is fixed in advance. The difference between SPIR and OT schemes is that, while the former require a communication complexity that is sublinear in the size of the database, the latter do not.

Naor and Pinkas proposed OT protocols with adaptive queries [15]. In these protocols, the client is allowed to learn at most k out of n data items, and while it can adaptively decide which ones to receive, the database server does not learn anything about the client's choices. The authors showed that these protocols can also be used for the client to learn whether or not an element exists in the database. In terms of efficiency, their protocol requires $\mathcal{O}(\log n)$ invocations of an 'one-out-of-two' oblivious transfer protocol [7].

Chor, Gilboa and Naor introduced Private Information Retrieval (PIR) by keywords [2]. Later, Ogata and Kurosawa introduced the more efficient notion of Oblivious Keyword Search (OKS), where the client learns the data items associated with his keyword privately [18]. This is a form of PSI with additional data transfer. One of their protocols, namely the one based on RSA blind signatures, is very efficient and proceeds as follows. Initially, the server generates an

[3] In [8] this is possible, for example, by returning an encryption of zero to BP.

[4] Since this solution requires that the ciphertext needs to be decrypted, one cannot use the more efficient exponential Elgamal.

RSA signature key pair and signs the keywords $w_1, w_2, \ldots$ yielding signatures $K_1, K_2, \ldots$. The server then commits to the each item of content c_i by computing the commitment $E_i = G(w_i||K_i||i) \oplus (0^l||c_i)$, where $G()$ is a pseudo-random function, w_i is the keyword associated with c_i, and l is a security parameter. The server then sends these commitments to the client, which subsequently asks the server for a blind signature on the keyword w of interest. After unbliding the received signature, denoted K, the client computes $G(w||K||i) \oplus E_i$ for every commitment E_i and, if the result has an l-bit prefix of zeroes, then the remaining bits constitute c_i.

De Cristofaro and Tsudik proposed a PSI protocol [4, Fig. 4], essentially removing the data transfer from [18]. As a result this protocol is more efficient than the one proposed by Ogata and Kurosawa, although the asymptotic complexity remains the same. If this scheme is used in our setting, then the communication complexity of the Init and the Query protocols are $\mathcal{O}(|B_v|)$ and $\mathcal{O}(1)$, respectively. The computation complexity for the Query protocol are $\mathcal{O}(|B_v|)$ for the client and $\mathcal{O}(1)$ for the server.

Huang, Evans, and Katz [10] proposed an approach based on garbled circuits, and showed that their approach is typically more efficient faster than De Cristofaro and Tsudik's protocol. De Cristofaro and Tsudik [5, 6], however, optimized their protocol in terms of efficiency which outperform garbled circuits. The identified drawbacks are that the scheme does not provide blacklist hiding across blacklist versions and it is not clear how to achieve proactive token binding, i.e. how to convert it to an APSI system.

Privacy-Preserving Revocation Checking. Solis and Tsudik argue that PIR techniques are too heavyweight for the purposes of blacklist checking, and they proposed a simpler alternative that uses the idea of certificate revocation trees [19]. Narisimha, Solis and Tsudik extended this idea to also include certificate revocation lists [16]. Both proposals require the client to query a range of k elements instead of a single element. An advantage of this approach is that there is no initialization phase, so for a small number of queries the communication overhead outperforms traditional CRLs. However, the degree of privacy depends on the size of the range interval; there is a trade-off between privacy and the number of elements to be transmitted. Furthermore, the systems do not provide blacklist hiding in our setting since the client also learns whether or not elements that have *not* been queried are on the blacklist.

4.2 Schemes Supporting Proactive Token Binding

Oblivious Signature-Based Envelope (OSBE) schemes enable a server to send a message to a client such that (a) the client obtains the message if and only if it is authorized to do so by a trusted third party, and (b) the server does not learn whether or not the client was given such authorization [14,17]. The trusted third party authorizes clients by issuing a special digital signature, namely an 'OSBE signature' σ, over another message μ that is known to both the client and

the server. OSBE schemes typically require the client to first send a 'blinded' version of σ to the server. The server then combines the received value with μ and adds extra randomness to it such that the server ends up with two values: a response for the client, and a symmetric encryption key that it uses to encrypt the message. The response together with the encrypted message is returned to the client. Due to the construction of the scheme, the client can only recover the symmetric key, and hence decrypt the ciphertext, if σ is a signature over μ.

OSBE schemes can be used to construct a remote blacklist system that, apart from $\mathcal{T}$-user privacy and zero-margin blacklist hiding, supports weak proactive token binding. To this end, it is required that the token issuing authority embeds an OSBE signature into the tokens it issues and that V extracts it at inspection; we assume that the signed message is the identifier of each token. In order to execute the Query protocol, V then simply proceeds according to the OSBE system: first it blinds the signature and sends it to BP. The blacklist provider then derives a response and a key for each entry on B_v, and encrypts a well-known message, e.g. '`lost or stolen`' under each key. The resulting responses and ciphertexts are then sent to V (in a random order) which then uses its knowledge of the OSBE signature in order to derive a key for each received response. If the ciphertext corresponding to any derived key decrypts to '`lost or stolen`', then the client concludes that the token has been blacklisted. Note that, using this approach, the client essentially derives $|B_v|$ symmetric keys in order to identify at most a single entry from the (randomized) blacklist.

It is possible to achieve a constant-factor reduction of the communication and computation complexity by replacing the encryptions of '`lost or stolen`' with a one-way hash value of each key. This has been suggested in the context of 'privacy-preserving policy-based information transfer' (PPIT) [3] and is also used in other contexts (e.g. for authentication based on a commitment that is hidden behind a hash function [12]). The asymptotic communication and computation complexity of the Query protocol, however, remains $\mathcal{O}(|B_v|)$. We stress that this cost is likely to be prohibitive in the scenario of passport inspection at busy airports. However, adopting this approach would not introduce incompatibility with current standards, since the signature σ can occupy one of the unused fields of the standard ePassport application specified in [11].

Based on a variant of PPIT that uses RSA signatures, De Cristofaro and Tsudik proposed a more efficient APSI protocol [4, Fig. 2]. Figure 1 shows our adaptation of this protocol to the remote blacklist setting. In this figure, the modulus N, the exponent e and the full-domain hash function H_1 constitute the public RSA signature key of the token issuer, σ denotes the token issuer's signature on the token identifier τ, H_2 is a one-way hash function, and g is a generator of the subgroup of quadratic residues modulo N. It is assumed that the parameters N, e, H_1, H_2, g are known to both V and BP.

While the original system in [4, Fig. 2] consists of a single protocol, our adaptation divides it into the Init and the Query protocols, as shown in the figure. The Init protocol starts with V choosing a random number r_{V} and sending $X = g^{r_{\mathsf{V}}}$ to BP. The blacklist provider then chooses its own random number

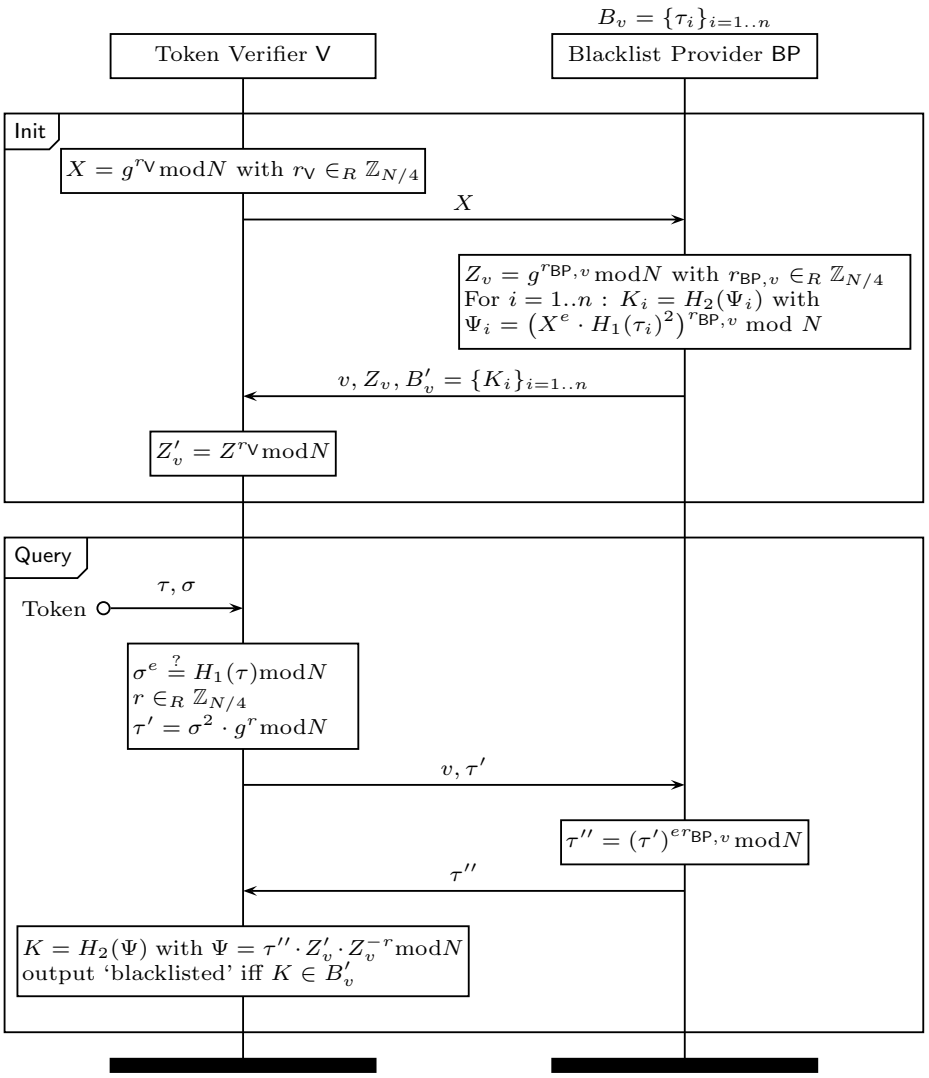


Fig. 1. Scheme from [4, Fig. 2] adapted to the remote blacklist setting

$r_{\mathsf{BP},v}$ and, for each blacklisted token identifier $\tau_i \in B_v$, computes the value $K_i = H_2((X^e \cdot H_1(\tau)^2)^{r_{\mathsf{BP},v}})$ and sends these values along with the version identifier v and the value $Z_v = g^{r_{\mathsf{BP},v}}$ to the verifier. V then computes the value $Z'_v = Z^{r_\mathsf{V}}$ and stores it together with the 'blinded' blacklist $B' = \{K_1, K_2, \dots\}$ and its version number. Note that the complexity of the Init protocol is $\mathcal{O}(|B_v|)$.

The Query proceeds as follows. First, V obtains the values τ and σ from the token. Then it verifies the signature, chooses a random number r from $\mathbb{Z}_{N/4}$, and computes the value $\tau' = \sigma^2 \cdot g^r \mod N$ and sends τ' to the blacklist provider.

In order to avoid any ambiguity in the presence of multiple blacklist versions, V also includes v in this message. BP then computes $\tau'' = (\tau')^{er_{\mathsf{BP},v}} \mod N$ and returns τ'' to V. Finally, the verifier computes $K = H_2(\tau'' \cdot Z'_v \cdot Z_v^r) \mod N$ and checks whether or not $K \in B'$. If it is, then it concludes that τ has been blacklisted.

As the original system [4, Fig. 2], our modified scheme above provides zero-margin blacklist hiding, online operation, and weak proactive token binding. As a result of dividing the scheme into two protocols, the complexity of the Query protocol is decoupled from the size of the blacklist; it is merely $\mathcal{O}(1)$. Moreover, the complexity of the Init protocol is amortized over a potentially large number of queries. However, in contrast to the original scheme, the same 'blinded version' of the blacklist B' that BP sends to V during the Init protocol is reused over multiple Query protocol executions. It remains to be shown that this does not introduce security issues.

A shortcoming of the scheme in Fig. 1 is that does not support blacklists that contain tokens issued by multiple issuers. That is, it provides only $\mathcal{T}_{\mathcal{I}}$-user privacy, where $\mathcal{T}_{\mathcal{I}}$ is the set of token identifiers that are signed by a given issuer using a particular signature key. This is because, for each blacklisted token, the blacklist provider must use, in its computations, the parameters of the corresponding issuer signature key; hence, the blacklist provider must be made aware to which 'group' the token that is currently being queried belongs. In the context of passport inspection, this means that it is not possible to hide the nationality of travellers from the blacklist provider. Our proposal in the next section does not suffer from this drawback.

5 Our Proposal

While the De Cristofaro and Tsudik's PSI scheme [4, Fig. 4], based on RSA blind signatures, is among the most efficient schemes proposed in the literature, it does not provide any form of token binding. Reactive token binding can nevertheless be achieved simply by requiring (a) the blacklist provider to keep a log of all blind signatures received) and (b) the token verifier to keep a log of all tokens identifiers, their signatures, and the randomness used to blind the signatures. Using these logs, an auditor can identify any superfluous queries that V issued to the blacklist provider, and initiate further investigation, as appropriate.

This section presents our proposal for a privacy-preserving remote blacklist checking. Like the scheme in [4, Fig. 4], it is based on RSA blind signatures. Section 5.1 presents our protocol and Sect. 5.2 compares its complexity to certain other schemes.

5.1 Protocol Description

We assume that the token issuer embeds the token identifier τ and its signature σ on τ into the token. In contrast to the requirements of the scheme shown in Fig. 1, we do not require the issuer to use any particular signature scheme. We further

assume that the blacklist provider knows σ for every blacklisted token. While this may seem to be an additional burden, it ensures that strict procedures are followed when adding tokens to the blacklist. That is, a token can be added only in cooperation with the issuer or someone that has or had physical access to the token.

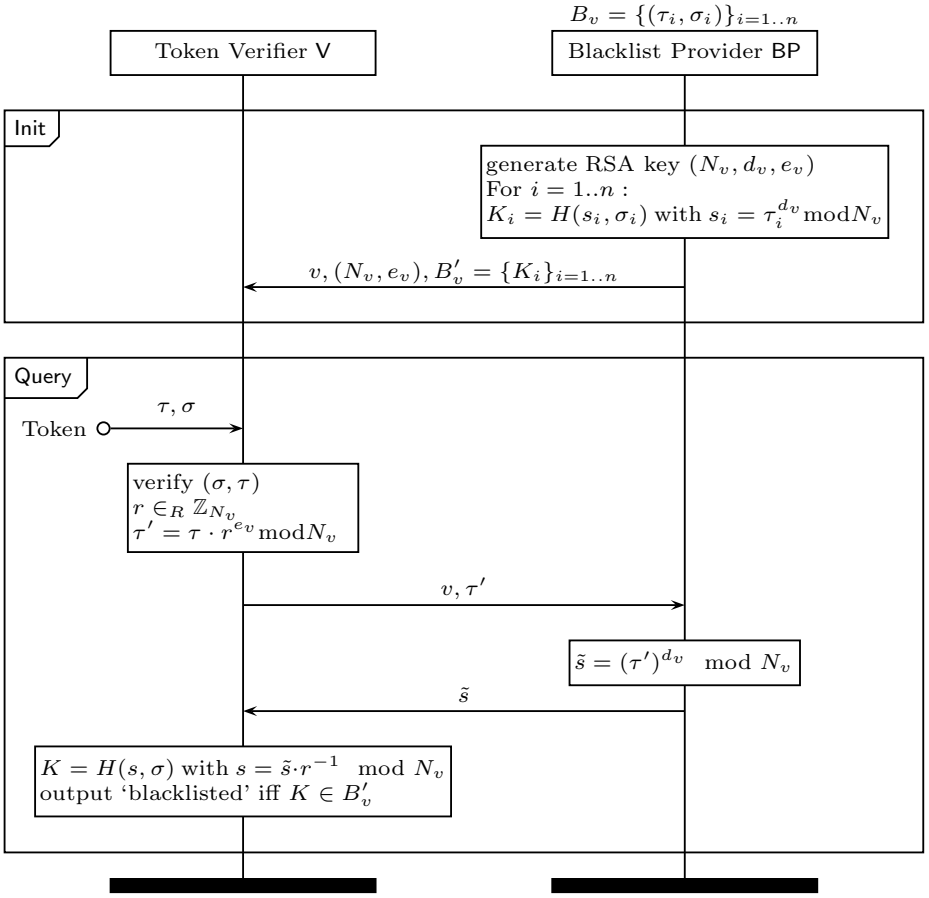


Fig. 2. Privacy-Friendly Checking of Remote Token Blacklists

Figure 2 shows our proposed scheme. In this figure, H denotes a one-way hash function that V and BP agree on. The Init protocol proceeds as follows. First, BP generates a fresh RSA signature key pair where (N_v, e_v) is the public verification key and d_v the secret key. Note that this key is used only for the current blacklist version. The blacklist provider then signs every blacklisted token using this key; s_i denotes the signature over the ith entry in the blacklist. Finally, it computes $K_i = H(s_i, \sigma_i)$ where σ_i denotes the token issuer's signature for the

corresponding entry, and sends these values, along with its public key (N_v, e_v) and the version number v to V. The verifier stores the received data.

The Query protocol proceeds as follows. First, V obtains τ and σ from the token. It then verifies σ, chooses a blinding factor r from $\mathbb{Z}_{N_v}$ and, using r together with the verification exponent e_v, it randomizes τ to obtain $\tau' = \tau \cdot r^{e_v}$. It then sends τ' together with the version number v (in order to avoid ambiguity) to BP. The blacklist provider then blindly signs the received value and obtains the blind signature $\tilde{s} = (\tau')^{d_v}$ which it sends back to V. Note that $\tilde{s}$ is a blind RSA signature on τ. By removing the blinding factor the verifier obtains $s = \tilde{s} \cdot r^{-1}$ and computes $K = H(s, \sigma)$. If this is identical to any of the values K_i on the blinded blacklist, then V concludes that the token is blacklisted.

Since the issuer's signature σ must be known to V in order to execute the protocol correctly, and since we assume that this signature can be obtained only from the token itself, our proposed scheme provides weak proactive token binding. It also provides $\mathcal{T}$-user privacy, zero-margin blacklist hiding, and online operation. Furthermore our proposed scheme has two advantages over the one discussed in Fig. 1: firstly, the blacklist can contain tokens issued by multiple issuers and, secondly, the 'blinded blacklist', i.e. the values K_i, does not depend on any value contributed by the verifier. This means that BP may precompute these values at any time, and may even be able to reuse the same values for multiple verifiers. In terms of computation and communication complexity, our proposed protocol is essentially identical to the PSI scheme in [4, Fig. 4];

It is worth mentioning that, in the context of passports inspection, the above scheme does not impose any change to already issued electronic passports compliant to the relevant standard [11]. This is because these passports already contain a signature that covers the passport number, and this signature is already read and verified by (compliant) inspection systems.

5.2 Comparison of Security Properties and Asymptotic Efficiency

Table 2 and 3 provide an overview of the asymptotic complexities of, and the privacy properties achieved by: the simple scheme discussed in Sect. 3, the 'Privacy-Preserving Revocation Checking' (PPRC) scheme proposed by Narisimha *et al.* [16], the APSI scheme of De Cristofaro and Tsudik [4, Fig. 2] with the discussed modifications in Sect. 4.2, and our proposal discussed in Sect. 5. In Table 3, $\mathcal{T}_{\mathcal{I}}$ denotes the set of token identifiers that are signed by a particular token issuer using the same signature key.

Table 4 provides a comparison in concrete computational and communicational costs specifically for the use case of checking a blacklist of passports. In this table, 'x exp' denotes that x mudular exponentiations must be computed. It is also assumed that ten million passports are blacklisted (i.e. that $|B_v| = 10^7$). This may seem to be a large number given that, according to some estimates, there are approximately five hundred million passports in circulation worldwide; we nevertheless believe that a real system must operate efficiently for blacklists of this order of magnitude. In order to provide a fair comparison between different schemes, we selected cryptographic keys sizes according to the 2011 ECRYPT II

Table 2. Asymptotic efficiency of existing schemes

Scheme	Init			Query		
	comp_{V}	$\text{comp}_{\mathsf{BP}}$	comm	comp_{V}	$\text{comp}_{\mathsf{BP}}$	comm
Simple scheme	-	$\mathcal{O}(\lvert B_v\rvert)$	$\mathcal{O}(\lvert B_v\rvert)$	$\mathcal{O}(\ln\ln\lvert B_v\rvert)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$
PPRC	-	-	-	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(\lvert T\rvert)$
APSI-1	$\mathcal{O}(1)$	$\mathcal{O}(\lvert B_v\rvert)$	$\mathcal{O}(\lvert B_v\rvert)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$
Our proposal	-	$\mathcal{O}(\lvert B_v\rvert)$	$\mathcal{O}(\lvert B_v\rvert)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$

Table 3. Privacy properties of existing schemes

Scheme	T-User Privacy	Blacklist Hiding	Online operation	Token binding
Simple scheme	Yes, for $T = B_v$	Yes	Yes	No
PPRC	Yes, with tunable T	No	Partially	No
APSI	Yes, for $T = \mathcal{T}_{\mathcal{I}}$	Yes	Yes	Weak Proactive
Our proposal	Yes, for $T = \mathcal{T}$	Yes	Yes	Weak Proactive

Table 4. Comparision of existing schemes, applied to the use case of passports

Scheme	Init			Query		
	comp_{V}	$\text{comp}_{\mathsf{BP}}$	comm	comp_{V}	$\text{comp}_{\mathsf{BP}}$	comm
Simple scheme	-	$9 \cdot 10^7$ exp	2.1 GB	6 exp	2 exp	56 B
PPRC	-	-	-	1 exp	1 exp	2.8 kB
APSI	2 exp	10^7 exp	267 MB	2 exp	1 exp	608 B
Our proposal	-	10^7 exp	267 MB	1 exp	1 exp	608 B

(European Network of Excellence in Cryptology) keysize report [1] for a security level that provides medium-term protection. More precisely, we assumed an RSA modulus of 2432 bits, and a bitlength of 224 for both elliptic curve elements and token identifiers.

6 Conclusion and Outlook

In this paper, we considered the problem of checking a remote blacklist in order to establish the legitimacy of a token in a privacy-preserving way, i.e. in a way that does not leak more information than strictly necessary. We compared existing schemes from the literature and described slight adaptations that tweak these schemes for optimal efficiency in the remote blacklist setting. Finally, we described how to adapt the most efficient scheme we found in the literature such that it achieves the property of 'weak proactive token binding'. This property guarantees that the token verifier cannot query the blacklist provider about tokens that it never saw. We believe that it is possible to use our proposal with electronic passports that conform to [11]. Moreover, due to its high efficiency and moderate storage requirements, we believe that it can be integrated into all online inspection devices, both fixed and mobile.

The work in this paper is limited in several respects, as follows. Firstly, while we believe that our survey covers most recent work in the area, is far from

complete. Secondly, the arguments in this paper are informal. Important future research includes the formalisation of the different security notions and adversary models, with the aim to provide security proofs. Thirdly, the construction of a scheme that simultaneously achieves strong proactive token binding, $\mathcal{T}$-user privacy and blacklist hiding, as well as the construction of efficient schemes with unconditional rather than computational privacy guarantees, is also an interesting direction of future research.

Acknowledgements. We would like to thank Julien Bringer for his insightful comments on an earlier version of this paper. This work was supported by the Flemish Government, IWT SBO SPION, FWO G.0360.11N Location Privacy, and by the Research Council KU Leuven: GOA TENSE; and by the European Commission through the FIDELITY project (contract number 284862).

References

1. Yearly Report on Algorithms and Keysizes (2011), D.SPA.17 Rev. 1.0. Technical report, ICT-2007-216676 ECRYPT II (June 2011)
2. Chor, B., Gilboa, N., Naor, M.: Private information retrieval by keywords. Cryptology ePrint Archive, Report 1998/003 (1998), `http://eprint.iacr.org/`
3. De Cristofaro, E., Jarecki, S., Kim, J., Tsudik, G.: Privacy-Preserving Policy-Based Information Transfer. In: Goldberg, I., Atallah, M.J. (eds.) PETS 2009. LNCS, vol. 5672, pp. 164–184. Springer, Heidelberg (2009)
4. De Cristofaro, E., Tsudik, G.: Practical Private Set Intersection Protocols with Linear Complexity. In: Sion, R. (ed.) FC 2010. LNCS, vol. 6052, pp. 143–159. Springer, Heidelberg (2010)
5. De Cristofaro, E., Tsudik, G.: Experimenting with Fast Private Set Intersection. In: Katzenbeisser, S., Weippl, E., Camp, L.J., Volkamer, M., Reiter, M., Zhang, X. (eds.) Trust 2012. LNCS, vol. 7344, pp. 55–73. Springer, Heidelberg (2012)
6. De Cristofaro, E., Tsudik, G.: On the performance of certain private set intersection protocols. Cryptology ePrint Report 2012/054 (2012), `http://eprint.iacr.org/`
7. Even, S., Goldreich, O., Lempel, A.: A randomized protocol for signing contracts. Commun. ACM 28(6), 637–647 (1985)
8. Freedman, M.J., Nissim, K., Pinkas, B.: Efficient Private Matching and Set Intersection. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 1–19. Springer, Heidelberg (2004)
9. Hohenberger, S., Weis, S.A.: Honest-Verifier Private Disjointness Testing Without Random Oracles. In: Danezis, G., Golle, P. (eds.) PET 2006. LNCS, vol. 4258, pp. 277–294. Springer, Heidelberg (2006)
10. Huang, Y., Evans, D., Katz, J.: Private set intersection: Are garbled circuits better than custom protocols? In: Proceedings of the NDSS 2012. IEEE (2012)
11. International Civil Aviation Organization. Document 9303, vol. 2, pt. 1 (2006)
12. Juels, A., Wattenberg, M.: A fuzzy commitment scheme. In: Motiwalla, J., Tsudik, G. (eds.) CCS 1999, Proceedings of the 6th ACM Conference on Computer and Communications Security, Singapore, November 1-4, pp. 28–36 (1999)
13. Kiayias, A., Mitrofanova, A.: Testing Disjointness of Private Datasets. In: S. Patrick, A., Yung, M. (eds.) FC 2005. LNCS, vol. 3570, pp. 109–124. Springer, Heidelberg (2005)

14. Li, N., Du, W., Boneh, D.: Oblivious signature-based envelope. Distributed Computing 17(4), 293–302 (2005)
15. Naor, M., Pinkas, B.: Oblivious Transfer with Adaptive Queries. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 573–590. Springer, Heidelberg (1999)
16. Narasimha, M., Solis, J., Tsudik, G.: Privacy-Preserving Revocation Checking. Int. J. Inf. Secur. 8(1), 61–75 (2009)
17. Nasserian, S., Tsudik, G.: Revisiting Oblivious Signature-Based Envelopes. In: Di Crescenzo, G., Rubin, A. (eds.) FC 2006. LNCS, vol. 4107, pp. 221–235. Springer, Heidelberg (2006)
18. Ogata, W., Kurosawa, K.: Oblivious keyword search. Journal of Complexity - Special issue on coding and cryptography 20(2-3), 356–371 (2004)
19. Solis, J., Tsudik, G.: Simple and Flexible Revocation Checking with Privacy. In: Danezis, G., Golle, P. (eds.) PET 2006. LNCS, vol. 4258, pp. 351–367. Springer, Heidelberg (2006)
20. Ye, Q., Wang, H., Pieprzyk, J., Zhang, X.-M.: Efficient Disjointness Tests for Private Datasets. In: Mu, Y., Susilo, W., Seberry, J. (eds.) ACISP 2008. LNCS, vol. 5107, pp. 155–169. Springer, Heidelberg (2008)
21. Ye, Q., Wang, H., Pieprzyk, J., Zhang, X.-M.: Unconditionally secure disjointness tests for private datasets. IJACT 1(3), 225–235 (2009)

Concepts and Languages for Privacy-Preserving Attribute-Based Authentication

Jan Camenisch[1], Maria Dubovitskaya[1], Anja Lehmann[1],
Gregory Neven[1], Christian Paquin[2], and Franz-Stefan Preiss[1]

[1] IBM Research – Zurich
[2] Microsoft Research

Abstract. Existing cryptographic realizations of privacy-friendly authentication mechanisms such as anonymous credentials, minimal disclosure tokens, self-blindable credentials, and group signatures vary largely in the features they offer and in how these features are realized. Some features such as revocation or de-anonymization even require the combination of several cryptographic protocols. These differences and the complexity of the cryptographic protocols hinder the deployment of these mechanisms for practical applications and also make it almost impossible to switch the underlying cryptographic algorithms once the application has been designed. In this paper, we aim to overcome this issue and simplify both the design and deployment of privacy-friendly authentication mechanisms. We define and unify the concepts and features of privacy-preserving attribute-based credentials (Privacy-ABCs) and provide a language framework in XML schema. Our language framework enables application developers to use Privacy-ABCs with all their features without having to consider the specifics of the underlying cryptographic algorithms—similar to as they do today for digital signatures, where they do not need to worry about the particulars of the RSA and DSA algorithms either.

Keywords: Authentication, privacy, data-minimization, anonymous credentials, digital credentials.

1 Introduction

More and more transactions in our daily life are performed electronically and the security of these transactions is an important concern. Strong authentication and according authorization based on certified attributes of the requester is paramount for protecting critical information and infrastructures online.

Most existing techniques for transferring trusted user attributes cause privacy issues. In systems where an online identity provider creates access tokens on demand, such as SAML, OpenID, or WS-Federation, the identity provider can impersonate its users and can track a user's moves online. Systems with offline token creation, such as X.509 certificates and some WS-Trust profiles, force the user to reveal more attributes than strictly needed (as otherwise the issuer's signature cannot be verified) and make her online transactions linkable across different websites.

These drawbacks can be overcome with privacy-preserving authentication mechanisms based on advanced cryptographic primitives such as anonymous credentials,

S. Fischer-Hübner et al.(Eds.): IDMAN 2013, IFIP AICT 396, pp. 34–52, 2013.

minimal disclosure tokens, self-blindable credentials, or group signatures [16,11,21,25,6,53]. In these schemes, users obtain certified credentials for their attributes from trusted issuers and later derive, without further assistance from any issuer, unlinkable tokens that reveal only the required attribute information yet remain verifiable under the issuer's public key. Well-known examples being Brands' scheme [11] and Camenisch-Lysyanskaya's scheme [21], which have been implemented in Microsoft's U-Prove [52] and IBM's Identity Mixer [36], respectively. Both implementations are freely available and efficient enough for practical use, yet the real-world adoption is slower than one may hope. One possible reason for the slow adoption of privacy-preserving authentication technologies might be that the various schemes described in the literature have a large set of features where similar features are often called differently or are realized with different cryptographic mechanisms. Many of the features such as credential revocation, efficient attribute encoding, or anonymity lifting even require a combination of separate cryptographic protocols. This makes these technologies hard to understand and compare and, most importantly, very difficult to use.

To overcome this, we provide unified definitions of the concepts and features of the different privacy-preserving authentication mechanisms. We will refer to this unification as *privacy-preserving attribute-based credentials* or *Privacy-ABCs*. Our definitions abstract away from the concrete cryptographic realizations but are carefully crafted so that they can be instantiated with the different cryptographic protocols—or a combination of them. To enable the use and integration of Privacy-ABCs in authentication and authorization systems, we further present cryptography-agnostic definitions of all concepts as well as a language framework with data formats for, e.g., policies and claims. All languages are specified in XML schema and separate the abstract functionality expected from the underlying cryptographic mechanisms from the opaque containers for the cryptographic data itself. Thus, these languages allow application developers to employ Privacy-ABCs without having to think about their cryptographic realization, similarly to how digital signatures or encryption can be used today. The language described in this paper has been implemented in the ABC4Trust project (www.abc4trust.eu) and will be made available as part of a reference implementation of a Privacy-ABC system which will include a number of cryptographic solutions. The full language description and schema are available as a project deliverable [15].

2 Related Work

Our work builds on the credential-based authentication requirements language (CARL) recently proposed by Camenisch et al. [26]. CARL allows a service provider (verifier) to specify which attributes certified by whom a user needs to present in order to get access. Compared to our work, CARL defines only a small part of a Privacy-ABC system, namely the presentation policy, but does not consider how these attributes are transmitted nor how credentials are issued or revoked. Bichsel et al. [7] have extended CARL to cover the transmission of certified attributes. Version 1 of the U-Prove protocols [52] covers credential issuance and presentation but only supports selective attribute disclosure. It does not consider other features such as attribute predicates, inspection, key binding, (cryptographic) pseudonyms, or revocation.

Privacy-ABCs can be used to realize a privacy-respecting form of attribute-based access control. Traditional attribute-based access control [8,56,54], however, does not see attributes as grouped together in a credential or token. Thus our framework allows one to realize more specific and more precise access control policies. Also, role-based access control [32,50] can be seen as a special case of our attribute-based setting by encoding the user's roles as attributes. Recent work [38] extended RBAC with privacy-preserving authentication for the particular case of role and location attributes.

Bonatti and Samarati [8] also propose a language for specifying access control rules based on "credentials". The language focuses on credential ownership and does not allow for more advanced requirements such as for example revealing of attributes, signing statements, or inspection. The same is true for the languages proposed by Ardagna et al. [2] and by Winsborough et al. [55]. However, the latter allows one to impose attribute properties on credentials and its extension by Li et al. [41] supports revealing of attributes. The Auth-SL language [48] focuses on multi-factor authentication and enables the policy author to specify restrictions on the properties of the authentication mechanisms themselves, but not on attributes of individual users.

The language by Ardagna et al. [1] can also be considered as a predecessor to our language in the sense that it focuses on anonymous credential systems and some of the advanced features. However, it considers only the presentation phase and is less expressive than ours, for instance, it cannot express statements involving attributes from different credentials.

VeryIDX [45] is a system to prevent identity theft by permitting the use of certain identity attributes only in combination with other identity attributes. So-called verification policies specify which attributes have to be presented together. However, these policies are introduced only conceptually without any details on exact expressivity, syntax, or semantics.

Several logic-based and technology-neutral approaches to distributed access control have been proposed [3,5,34,40]. However, none of these have been designed with Privacy-ABCs in mind. In particular, they do not support selective disclosure of attributes, proving predicates over attributes, or attribute inspection.

Summarized, our language framework is the first that covers the whole life-cycle of Privacy-ABCs and also the first one unifying the full spectrum of their features.

3 Concepts and Features

Figure 1 gives an overview of the entities involved in Privacy-ABC systems and the interactions between them. The interactions are named according to their purpose. Depending on the technical realizations, these interactions will be realized differently and might occur multiple times using different protocols (we consider sending a single message also a protocol). These entities are *users*, *issuers*, *verifiers*, *inspectors* and *revocation authorities*. Each issuer generates a secret issuance key and publishes the *issuer parameters* that include the corresponding public verification key. Similarly, each inspector generates a private decryption key and a corresponding public encryption key, and each revocation authority generates and publishes its revocation parameters. We

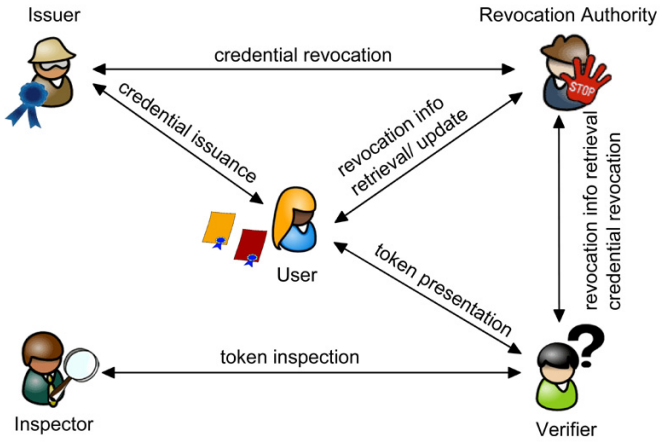


Fig. 1. Entities and the interactions between them

assume that all entities have means to retrieve the public keys of the issuers, revocation authorities, inspectors, and verifiers. Users get issued credentials by issuers via the *credential issuance* protocol. A credential contains attributes that its issuer vouches for w.r.t. the user. A credential can also specify one or more revocation authorities who are able to revoke the credential if necessary for some reason. To issue a credential that is revocable, the user and/or the issuer might need to interact with the revocation authorize prior or during the issuance protocol. Using her credentials, a user can form a presentation token that contains a subset of the certified attributes, provided that the corresponding credentials have not been revoked. This process might require the user to retrieve information from the revocation authority. Additionally, some of the attributes can be encoded in the presentation token so that they can only be retrieved by an inspector. The user can attach inspection grounds specifying under this condition the inspector should reveal these attributes. Receiving a presentation token from a user, a verifier checks whether the presentation token is valid w.r.t. the relevant issuers' public keys and inspector public keys and the latest revocation information (thus, the verifier will interact with the revocation authority). If the verification succeeds, the verifier will be convinced that the attributes contained in the presentation token are vouched for by the corresponding issuers. Finally, if a presentation token contains attributes that can only be retrieved by an inspector and the inspection grounds are met, the verifier can interact with the inspector to learn these attributes.

Informally, a secure realization of a Privacy-ABC system guarantees that (1) users can only generate a valid presentation token if they were indeed issued the corresponding credentials that have not been revoked, (2) that attributes encoded in the presentation token for an inspector can indeed be retrieved by that inspector, and (3) that the presentation tokens do not reveal any further information about the users other than the attributes contained in them.

We now provide a brief explanation of the main features supported by Privacy-ABCs, with a focus on the ones that were not modeled so far in existing identity frameworks.

3.1 Pseudonyms

Each user can generate a secret key. However, unlike traditional public-key authentication schemes, there is no single public key corresponding to the secret key. Rather, the user can generate as many public keys as she wishes. These public keys are called *pseudonyms* in Privacy-ABCs. Pseudonyms [16,42] are cryptographically unlinkable, meaning that given two different pseudonyms, one cannot tell whether they were generated from the same or from different secret keys. By generating a different pseudonym for every verifier, users can thus be known under different unlinkable pseudonyms to different sites, yet use the same secret key to authenticate to all of them.

While it is sufficient for users to generate a single secret key, they can also have multiple secret keys. A secret key can be generated by a piece of trusted hardware (e.g., a smart card) that stores and uses the key in computations (e.g., to generate pseudonyms), but that never reveals the key. The key is thereby *bound* to the hardware, in the sense that it can only be used in combination with the hardware.

There are situations, however, where the possibility to generate an unlimited number of unlinkable pseudonyms is undesirable. For example, in an online opinion poll, users should not be able to bias the result by entering multiple votes under different pseudonyms. In such situations, the verifier can request a special pseudonym called a *scope-exclusive pseudonym*, which is unique for the user's secret key and a given *scope string* [35]. Scope-exclusive pseudonyms for different scope strings remain unlinkable. By using the URL of the opinion poll as the scope string, for example, the verifier can ensure that each user can only register a single pseudonym to vote.

3.2 Credentials and Key Binding

A *credential* is a certified container of attributes issued by an issuer to a user. Formally, an *attribute* is described by the *attribute type* that determines the semantics of the attribute (e.g., first name) and the *attribute value* that determines its contents (e.g., "John"). By issuing a credential, the issuer vouches for the correctness of the contained attributes with respect to the user. The *credential specification* lists the attribute types that are encoded in a credential. A credential specification can be created by the issuer, or by an external authority so that multiple issuers can issue credentials according to the same specification. The credential specification must be published and distributed over a trusted channel. How exactly this is done goes beyond the scope of our language framework; the specification could for example be digitally signed by its creator.

Optionally, a credential can be *bound* to a user's secret key, i.e., it cannot be used without knowing the secret key [42]. We call this option *key binding*. It is somewhat analogous to traditional public-key certificates, where the certificate contains the CA's signature on the user's public key, but unlike traditional public-key certificates, a Privacy-ABC is not bound to a unique public key: it is only bound to a unique secret key. A user can derive as many pseudonyms as she wishes from this secret key and (optionally) show that they were derived from the same secret key that underlies the credential.

3.3 Presentation

To authenticate to a verifier, the user first obtains the *presentation policy* that describes which credentials the user must present and which information from these credentials

she must reveal. If the user possesses the necessary credentials, she can derive from these credentials a *presentation token* that satisfies the presentation policy. The presentation token can be verified using the issuer parameters of all credentials underlying the presentation token.

Presentation tokens derived from Privacy-ABCs only reveal the attributes that were explicitly requested by the presentation policy – all the other attributes contained in the credentials remain hidden. Moreover, presentation tokens are cryptographically unlinkable (meaning no collusion of issuers and verifiers can tell whether two presentation tokens were generated by the same user or by different users) and untraceable (meaning that no such collusion can correlate a presentation token to the issuance of the underlying credentials). Of course, presentation tokens are only as unlinkable as the information they intentionally reveal.

Rather than requesting and revealing full attribute values, presentation policies and tokens can also request and reveal *predicates* over one or more issued attributes. For example, a token could reveal that the name on the user's credit card matches that on her driver's license, without revealing the name. As another example, a token could reveal that the user's birthdate is before January 1st, 1994, without revealing her exact birthdate.

3.4 Issuance

In the simplest setting, an issuer knows all attribute values to be issued and simply embeds them into a credential. Privacy-ABCs also support advanced issuance features where attributes are blindly "carried over" from existing credentials, without the issuer becoming privy to their values. Similarly, the issuer can blindly issue self-claimed attribute values (i.e., not certified by an existing credential), carry over the secret key to which a credential is bound, or assign a uniformly random value to an attribute such that the issuer cannot see it and the user cannot bias it [11,24].

Advanced issuance is an interactive protocol between the user and the issuer. In the first move, the issuer provides the user with an *issuance policy* that consists of a presentation policy specifying which pseudonyms and/or existing credentials the user must present, and of a *credential template* specifying which attributes or secret keys of the newly issued credential will be generated at random or carried over from credentials or pseudonyms in the presentation policy. In response, the user sends an *issuance token* containing a presentation token that satisfies the issuance policy. Then the (possibly multi-round) cryptographic issuance protocol ensues, at the end of which the user obtains the new credential.

3.5 Inspection

Absolute user anonymity in online services easily leads to abuses such as spam, harassment, or fraud. Privacy-ABCs provide the option to add accountability for misbehaving users through a feature called *inspection* [12,27]. Here, a presentation token contains one or more credential attributes that are encrypted under the public key of a trusted *inspector*. The verifier can check that the correct attribute values were encrypted, but cannot see their actual values. The *inspection grounds* describe the circumstances under which the verifier may call upon the inspector to recover the actual attribute values.

The inspector is trusted to collaborate only when the inspection grounds have been met; verifiers cannot change the inspection grounds after receiving a presentation token, as the grounds are cryptographically tied to the token.

The presentation policy specifies which attributes from which credentials have to be encrypted, together with the inspector public keys and inspection grounds under which they have to be encrypted.

3.6 Revocation

Credentials may need to be revoked for several reasons: the credential and the related user secrets may have been compromised, the user may have lost her right to carry a credential, or some of her attribute values may have changed. In such cases, credentials need to be revoked globally and we call this *issuer-driven revocation*. Sometimes credentials may be revoked only for specific contexts. For example, a hooligan may see his digital identity card revoked for accessing sport stadiums, but may still use it for all other purposes. We call this *verifier-driven revocation*.

Revocation for Privacy-ABCs is cryptographically more complicated than for classical certificates, but many mechanisms with varying efficiency [39] exist [23,44,10,18,43]. Bar a few exceptions, all of them can be used for both issuer-driven and verifier-driven revocation.

We describe revocation in a generic mechanism-agnostic way and consider credentials to be revoked by dedicated *revocation authorities*. They are separate entities in general, but may be under the control of the issuer or verifier in particular settings. The revocation authority publishes static *revocation authority parameters* and periodically publishes the most recent *revocation information*. When creating presentation tokens, users prove that their credentials have not been revoked, possibly using *non-revocation evidence* that they fetch and update from the revocation authority. The revocation authority to be used is specified in the issuer parameters for issuer-driven revocation and in the presentation policy for verifier-driven revocation. When a credential is subject to issuer-driven revocation, a presentation token related to this credential must always contain a proof that the presented credential has not been revoked. Issuer-driven revocation is performed based on the *revocation handle*, which is a dedicated unique attribute embedded in a credential. Verifier-driven revocation can be performed based on any combination of attribute values, possibly even from different credentials. This allows the revocation authority for example to exclude certain combinations of first names and last names to be used in a presentation token.

3.7 Cryptographic Realization

Among the most prominent instantiations of Privacy-ABC systems are IBM's Identity Mixer [36] and Microsoft's U-Prove [52]. Both systems currently support only a subset of the features presented here, but will be extended to support the full feature set as part of the ABC4Trust project. Here, we sketch how the different features can be realized cryptographically and give pointers to relevant related literature; a full security analysis of the combined system is beyond the scope of this paper.

At the core of a Privacy-ABC system is a signature scheme with efficient protocols to prove possession of signatures. Identity Mixer and U-Prove build on the

Camenisch-Lysyanskaya [24] and the Brands [11] signature schemes, respectively, but also other schemes exist [25,4]. An issued credential is a signature (or, for single-show schemes such as [11], a batch of renewable signatures) under such a scheme on the user's attributes. Some schemes inherently support key binding [24,25,4], others can be extended by using a randomly chosen attribute as secret key [11].

Ordinary pseudonyms are cryptographic commitments [46,29] to the secret key; scope-exclusive pseudonyms can be realized as the output of a verifiable random function [30,17] applied to the secret key as seed and the scope string as input. Inspection can be obtained through verifiable encryption [27] of the inspectable attributes. Revocation of Privacy-ABCs can be done through signed revocation lists [43], through dynamic accumulators [23,44,18], or through efficient updates of short-lived credentials [19].

The *glue* binding all primitives together in a presentation token is provided by generalized zero-knowledge proofs of knowledge of discrete logarithms [47,20] made non-interactive through the Fiat-Shamir transform [33]. These proofs are also used for equality predicates over attributes; inequality predicates are proved with range proofs [9,13].

4 Language Framework

Given the multitude of distributed entities involved in a full-fledged Privacy-ABC system, the communication formats that are used between these entities must be specified and standardized.

None of the existing format standards for identity management protocols such as SAML, WS-Trust, or OpenID support all Privacy-ABCs' features. Although most of them can be extended to support a subset of these features, we define for the sake of simplicity and completeness a dedicated language framework which addresses all unique Privacy-ABC features. Our languages can be integrated into existing identity management systems.

In this section we introduce our framework covering the full life-cycle of Privacy-ABCs, including setup, issuance, presentation, revocation, and inspection. As the main purpose of our data artifacts is to be processed and generated by automated policy and credential handling mechanisms, we define all artifacts in XML schema notation, although one could also create a profile using a different encoding such as Abstract Syntax Notation One (ASN.1) [37] or JavaScript Object Notation (JSON) [28].

The XML artifacts formally describe and orchestrate the underlying cryptographic mechanisms and provide opaque containers for carrying the cryptographic data. Whenever appropriate, our formats also support user-friendly textual names or descriptions which allow to show a descriptive version of the XML artifacts to a user and to involve her in the issuance or presentation process if necessary.

For didactic purposes we describe the different artifacts realizing the concepts from Section 3 by means of examples. For the sake of space and readability, these examples do not illustrate all features described in the previous section; we refer the reader to [15] for the full specification. In what follows, we explicitly distinguish between user attributes (as contained in a credential) and XML attributes (as defined by XML schema) whenever they could be confused.

4.1 Credential Specification

Recall that the credential specification describes the common structure and possible features of credentials. For example, assume the Republic of Utopia issues electronic identity cards to its citizens containing their full name, state, and date of birth. Utopia may issue Privacy-ABCs according to the credential specification shown in Figure 2.

```
1  <CredentialSpecification KeyBinding="true" Revocable="true">
2    <SpecificationUID> urn:creds:id </SpecificationUID>
3    <AttributeDescriptions MaxLength="32">
4        <AttributeDescription Type="urn:creds:id:name" DataType="xs:string" Encoding="xenc:sha256">
5          <FriendlyAttributeName lang="EN"> Full Name </FriendlyAttributeName>
6        </AttributeDescription>
7        <AttributeDescription Type="urn:creds:id:state" DataType="xs:string" Encoding="xenc:sha256"/>
8        <AttributeDescription Type="urn:creds:id:bdate" DataType="xs:date" Encoding="date:unix:unsigned"/>
9    </AttributeDescriptions>
10 </CredentialSpecification>
```

Fig. 2. Credential specification of the identity card

The XML attribute **KeyBinding** indicates whether credentials adhering to this specification must be bound to a secret key. The XML attribute **Revocable** being set to *"true"* indicates that the credentials will be subject to issuer-driven revocation and hence have a built-in revocation handle. The assigned revocation authority is specified in the issuer parameters.

To encode user attribute values in a Privacy-ABC, they must be mapped to integers of a limited length. The maximal length depends on the security parameter (basically, it is the bit length of exponents in the group) and is indicated by the **MaxLength** XML attribute (Line 3), here 32 bytes. In our example, electronic identity cards contain a person's full name, state, and date of birth. The XML attributes **Type**, **DataType**, and **Encoding** respectively contain the unique identifier for the user attribute type, for the data type, and for the encoding algorithm that specifies how the value is to be mapped to an integer of the correct size (Lines 4,7,8). Attributes that may have values longer than **MaxLength** have to be hashed, as is done here for the name using SHA-256. The specification can also define human-readable names for the user attributes in different languages (Line 5).

4.2 Issuer and Revocation Parameters

The government of Utopia, that acts as issuer and revocation authority for the identity cards, generates an issuance key pair and publishes the issuer parameters, and generates and publishes the revocation authority parameters, which are illustrated in Figure 3.

The **ParametersUID** element assigns unique identifiers for the issuer and revocation authority parameters. The issuer parameters additionally specify the chosen cryptographic Privacy-ABC and hash algorithm, the credential specification that credentials issued under these issuer parameters will follow, and the parameters identifier of the revocation authority that will manage the issuer-driven revocation. The **SystemParameters**, **CryptoParams**, and **KeyBindingInfo** contain cryptographic algorithm-specific information about the public key.

```
<IssuerParameters>
  <ParametersUID> urn:utopia:id:issuer </ParametersUID>
  <AlgorithmID> urn:com:microsoft:uprove </AlgorithmID>
  <SystemParameters> ... </SystemParameters>
  <CredentialSpecUID> urn:creds:id </CredentialSpecUID>
  <HashAlgorithm> xenc:sha256 </HashAlgorithm>
  <CryptoParams> ... </CryptoParams>
  <KeyBindingInfo> ... </KeyBindingInfo>
  <RevocationParametersUID> urn:utopia:id:ra </RevocationParametersUID>
</IssuerParameters>
```

```
<RevocationAuthorityParameters>
  <ParametersUID> urn:utopia:id:ra </ParametersUID>
  <RevocationMechanism> urn:privacy−abc:accumulators:cl </RevocationMechanism>
  <RevocationInfoReference ReferenceType="url"> https:utopia.gov/id/revauth/revinfo
    </RevocationInfoReference>
  <NonRevocationEvidenceReference ReferenceType="url"> https:utopia.gov/id/revauth/nrevevidence
    </NonRevocationEvidenceReference>
  <CryptoParams> ... </CryptoParams>
</RevocationAuthorityParameters>
```

Fig. 3. Issuer and revocation authority parameters

The revocation authority parameters can be used for both issuer- and verifier-driven revocation. They specify a unique identifier for the parameters, the cryptographic revocation mechanisms, and references to the network endpoints where the most recent revocation information and non-revocation evidence can be fetched.

4.3 Presentation Policy with Basic Features

Assume that a user already possesses an identity card from the Republic of Utopia issued according to the credential specification depicted in Figure 2. Further, assume that all residents of Utopia can sign up for one free library card using an online issuance service. To get a library card the applicant must present her valid identity card and reveal (only) the state attribute certified by the card. This results in the presentation policy depicted in Figure 4.

```
<PresentationPolicy PolicyUID="libcard">
  <Message>
    <Nonce> bkQydHBQWDR4TUZzbXJKYUM= </Nonce>
  </Message>
  <Pseudonym Alias="nym" Scope="urn:library:issuance" Exclusive="true"/>
  <Credential Alias="id" SameKeyBindingAs="nym">
    <CredentialSpecAlternatives>
      <CredentialSpecUID> urn:creds:id </CredentialSpecUID>
    </CredentialSpecAlternatives>
    <IssuerAlternatives>
      <IssuerParametersUID> urn:utopia:id:issuer </IssuerParametersUID>
    </IssuerAlternatives>
    <DisclosedAttribute AttributeType= "urn:creds:id:state"/>
  </Credential>
</PresentationPolicy>
```

Fig. 4. Presentation policy for an identity card

We now go through the preceding presentation policy and describe how the different features of Privacy-ABCs can be realized with our language. We first focus on the basic

features and describe extended concepts such as inspection and revocation in our second example.

Signing Messages. A presentation token can optionally sign a message. The message to be signed is specified in the policy (Fig. 4, Lines 2–4). It can include a nonce, any application-specific message, and a human-readable name and/or description of the policy. The nonce will be used to prevent replay attacks, i.e. to ensure freshness of the presentation token, and for cryptographic evidence generation. Thus, when making use of the nonce, the presentation policy is not static anymore, but needs to be completed with a fresh nonce element for every request.

Pseudonyms. The optional **Pseudonym** element (Fig. 4, Line 5) indicates that the presentation token must contain a pseudonym. A pseudonym can be presented by itself or in relation with a credential if key binding is used (which we discuss later).

The associated XML attribute **Exclusive** indicates that a scope-exclusive pseudonym must be created, with the scope string given by the XML attribute **Scope**. This ensures that each user can create only a single pseudonym satisfying this policy, so that the registration service can prevent the same user from obtaining multiple library cards. Setting **Exclusive** to *"false"* would allow an ordinary pseudonym to be presented. The **Pseudonym** element has an optional boolean XML attribute **Established**, not illustrated in the example, which, when set to *"true"*, requires the user to re-authenticate under a previously established pseudonym. The presentation policy can request multiple pseudonyms, e.g., to verify that different pseudonyms actually belong to the same user.

Credentials and Selective Disclosure. For each credential that the user is requested to present, the policy contains a **Credential** element (Fig. 4, Lines 6–14), which describes the credential to present in detail. In particular, disjunctive lists of the accepted credential specifications and issuer parameters can be specified via **CredentialSpecAlternatives** and **IssuerAlternatives** elements, respectively (Fig. 4, Lines 7-9 and 10–12). The credential element also indicates all attributes that must be disclosed by the user via **DisclosedAttribute** elements (Fig. 4, Line 13). The XML attribute **Alias** assigns the credential an alias so that it can be referred to from other places in the policy, e.g., from the attribute predicates.

Key Binding. If present, the **SameKeyBindingAs** attribute of a **Credential** or **Pseudonym** element (Fig. 4, Line 6), contains an alias referring either to another Pseudonym element within this policy, or to a Credential element for a credential with key binding. This indicates that the current pseudonym or credential and the referred pseudonym or credential have to be bound to the same key. In our preceding example, the policy requests that the identity card and the presented pseudonym must belong to the same secret key.

Issuance Policy. To support the advanced features described in Section 3, we propose a dedicated *issuance policy*. A library card contains the applicant's name and is bound to the same secret key as the identity card. So the identity card must not only be presented, but also used as a source to carry over the name and the secret key to the library card, and the library should learn neither of them during the issuance process. Altogether,

to issue library cards the state library creates the issuance policy depicted in Figure 5. It contains the presentation policy from Figure 4 and the credential template that is described in detail below.

```
1  <IssuancePolicy>
2    <PresentationPolicy PolicyUID="libcard"> ... </PresentationPolicy>
3    <CredentialTemplate SameKeyBindingAs="id">
4      <CredentialSpecUID> urn:utopia:lib </CredentialSpecUID>
5      <IssuerParametersUID> urn:utopia:lib:issuer </IssuerParametersUID>
6      <UnknownAttributes>
7        <CarriedOverAttribute TargetAttributeType= "urn:utopia:lib:name">
8          <SourceCredentialInfo Alias="id" AttributeType="urn:creds:id:name"/>
9        </CarriedOverAttribute>
10     </UnknownAttributes>
11   </CredentialTemplate>
12 </IssuancePolicy>
```

Fig. 5. Issuance policy for a library card. The presentation policy on Line 2 is depicted in Figure 4.

Credential Template. A credential template describes the relation of the new credential to the existing credentials that were requested in the presentation policy. The credential template (Fig. 5, Lines 3–11) must first state the unique identifier of the credential specification and issuer parameters of the newly issued credential. The optional XML attribute **SameKeyBindingAs** further specifies that the new credential will be bound to the same secret key as a credential or pseudonym in the presentation policy, in this case the identity card.

Within the **UnknownAttributes** element (Fig. 5, Lines 6–10) it is specified which user attributes of the new credential will be carried over from existing credentials in the presentation token. The **SourceCredentialInfo** element (Fig. 5, Line 8) indicates the credential and the user attribute of which the value will be carried over.

Although this is not illustrated in our example, an attribute value can also be specified to be chosen jointly at random by the issuer and the user. This is achieved by setting the optional XML attribute **JointlyRandom** to *"true"*.

4.4 Presentation and Issuance Token

A *presentation token* consists of the *presentation token description*, containing the mechanism-agnostic description of the revealed information, and the *cryptographic evidence*, containing opaque values from the specific cryptography that "implements" the token description. The presentation token description roughly uses the same syntax as a presentation policy. An *issuance token* is a special presentation token that satisfies the stated presentation policy, but that contains additional cryptographic information required by the credential template.

The main difference to the presentation and issuance policy is that in the returned token a **Pseudonym** (if requested in the policy) now also contains a **PseudonymValue** (Fig. 6, Line 6). Similarly, the **DisclosedAttribute** elements (Fig. 6, Lines 10–12) in a token now also contain the actual user attribute values. Finally, all data from the cryptographic implementation of the presentation token and the advanced issuance features are grouped together in the **CryptoEvidence** element (Fig. 6, Line 17). This data includes, e.g., proof

```
1  <IssuanceToken>
2    <IssuanceTokenDescription>
3      <PresentationTokenDescription PolicyUID ="libcard" >
4        <Message> ... </Message>
5        <Pseudonym Alias="nym" Scope="urn:library:issuance" Exclusive="true" />
6          <PseudonymValue> MER2VXISHI=</PseudonymValue>
7        </Pseudonym>
8        <Credential Alias="id" SameKeyBindingAs="nym" >
9              ...
10         <DisclosedAttribute AttributeType="urn:creds:id:state" >
11           <AttributeValue> Nirvana </AttributeValue>
12         </DisclosedAttribute>
13       </Credential>
14     </PresentationTokenDescription>
15     <CredentialTemplate SameKeyBindingAs="id" > ... </CredentialTemplate>
16   </IssuanceTokenDescription>
17   <CryptoEvidence> ... </CryptoEvidence>
18 </IssuanceToken>
```

Fig. 6. Issuance token for obtaining the library card

that the contained identity card is not revoked by the issuer and that it is bound bound to the same secret key as the pseudonym.

4.5 Presentation Policy with Extended Features

Assume that the state library has a privacy-friendly online interface for borrowing digital and paper books. Books can be browsed and borrowed anonymously using the digital library cards based on Privacy-ABCs. Paper books can be delivered in anonymous numbered mailboxes at the post office. However, when books are returned late or damaged, the library must be able to identify the reader to impose an appropriate fine. Repeated negligence may even lead to exclusion from borrowing further paper books while borrowing digital books remains possible.

Moreover, assume that the library offers special conditions for young readers that can be used by anyone below the age of twenty-six years. As library cards do not contain a date of birth, a user must prove to be below that age by combining her library card with her identity card. Altogether, for borrowing books under the "young-reader"-conditions, users have to satisfy the presentation policy depicted in Figure 7.

A presentation policy that is used for plain presentation (i.e., not within an issuance policy) can consist of multiple policy alternatives, each wrapped in a separate **PresentationPolicy** element (Fig. 7, Lines 2–34). The returned presentation token must satisfy (at least) one of the specified policies.

The example presentation policy requires two **Credential** elements, for the library and for the identity card, which must belong to the same secret key as indicated by the XML attribute **SameKeyBindingAs**.

Attribute Predicates. No user attributes of the identity card have to be revealed, but the **AttributePredicate** element (Fig. 7, Lines 30–33) specifies that the date of birth must be after July 16th, 1986, i.e., that the reader is younger than twenty-six. Supported predicate functions include equality, inequality, greater-than and less-than tests for most basic data types, as well as membership of a list of values. The arguments of the predicate function may be credential attributes (referred to by the credential alias and the attribute

```
<PresentationPolicyAlternatives>
  <PresentationPolicy PolicyUID= "young−reader" >
    <Message> ... </Message>
    <Credential Alias="libcard" SameKeyBindingAs="id" >
      <CredentialSpecAlternatives>
        <CredentialSpecUID> urn:utopia:lib </CredentialSpecUID>
      </CredentialSpecAlternatives>
      <IssuerAlternatives>
        <IssuerParametersUID> urn:utopia:lib:issuer </IssuerParametersUID>
      </IssuerAlternatives>
      <DisclosedAttribute AttributeType= "urn:utopia:lib:name" >
        <InspectorAlternatives>
          <InspectorPublicKeyUID> urn:lib:arbitrator </InspectorPublicKeyUID>
        </InspectorAlternatives>
        <InspectionGrounds> Late return or damage. </InspectionGrounds>
      </DisclosedAttribute>
    </Credential>
    <Credential Alias="id" >
      <CredentialSpecAlternatives>
        <CredentialSpecUID> urn:creds:id </CredentialSpecUID>
       </CredentialSpecAlternatives>
       <IssuerAlternatives>
         <IssuerParametersUID> urn:utopia:id:issuer </IssuerParametersUID>
       </IssuerAlternatives>
    </Credential>
    <VerifierDrivenRevocation>
      <RevocationParametersUID> urn:lib:blacklist </RevocationParametersUID>
      <Attribute CredentialAlias ="libcard" AttributeType="urn:utopia:lib:name" />
    </VerifierDrivenRevocation>
    <AttributePredicate Function= "...:date−greater−than" >
      <Attribute CredentialAlias ="id" AttributeType= "urn:creds:id:bdate" />
      <ConstantValue> 1986−07−16 </ConstantValue>
    </AttributePredicate>
  </PresentationPolicy>
</PresentationPolicyAlternatives>
```

Fig. 7. Presentation policy for borrowing books

type) or constant values. See [15] for an exhaustive list of supported predicates and data types and note that an attribute's encoding as defined in the credential specification has implications on which predicates can be used for it and whether it is inspectable [15, Sec. 4.2.1].

Inspection. To be able to nevertheless reveal the name of an anonymous borrower and to impose a fine when a book is returned late or damaged, the library can make use of inspection. The **DisclosedAttribute** element for the user attribute *"...:name"* contains **InspectorPublicKeyUID** and **InspectionGrounds** child elements, indicating that the attribute value must not be disclosed to the verifier, but to the specified inspector with the specified inspection grounds. The former child element specifies the inspector's public key under which the value must be encrypted, in this case belonging to a designated arbiter within the library. The latter element specifies the circumstances under which the attribute value may be revealed by the arbiter. Our language also provides a data artifact for inspection public keys, which we omit here for space reasons.

Issuer-Driven Revocation. When the presentation policy requests a credential that is subject to issuer-driven revocation (as defined in the credential specification), the credential must be proved to be valid with respect to the most recent revocation

information. However, a policy can also require the use of a particular version of the revocation information. In the latter case, the element **IssuerParametersUID** has an extra XML attribute **RevocationInformationUID** specifying the identifier of the specific revocation information. The specification of the referenced **RevocationInformation** is given in [15]. Presentation tokens can accordingly state the validity of credentials w.r.t. a particular version by using a **RevocationInformationUID** XML element in the corresponding Credential element.

Verifier-Driven Revocation. If customers return borrowed books late or damaged, they are excluded from borrowing further paper books, but they are still allowed to use the library's online services. In our example, this is handled by a **VerifierDrivenRevocation** element (Fig. 7, Lines 26–29), which specifies that the user attribute *"...:name"* of the library card must be checked against the most recent revocation information from the revocation authority *"urn:lib:blacklist"*. Revocation can also be based on a combination of user attributes from different credentials, in which case there will be multiple **Attribute** child elements per **VerifierDrivenRevocation**. The presentation policy can also contain multiple **VerifierDrivenRevocation** elements for one or several credentials, the returned presentation token must then prove its non-revoked status for *all* of them.

5 Security Discussion

For our framework to be useful, the various parties need to be given security guarantees: a verifier wants to be sure that if a presentation token verifies then all the statements made in it are indeed supported by the issuer and have not been revoked. Furthermore, the verifier wants to be sure that inspection will succeed when needed. The issuer wants to have similar guarantees w.r.t. issuance tokens. The user wants assurance that her privacy is maintained, i.e., that no more information is leaked than what she willingly released in a presentation token.

Formally proving that such guarantees are fulfilled, provided the underlying cryptography is sound, is far beyond the scope of this paper and is the topic of future work. Likewise, we do not go into detail here on the complex trust relations between the different participants in a Privacy-ABC system, or on which particular privacy or security threats can arise when some of these participants collude. Since presentation policies can always ask to reveal much more information than strictly necessary, one could also consider adding yet another authority to the system to approve "reasonable" policies. Finally, in order to deploy a Privacy-ABC system, one also needs a public-key infrastructure (PKI) to certify issuer parameters, revocation authority parameters, and inspectors' public keys.

For the individual cryptographic building blocks, security proofs are given in the cryptographic literature, and Camenisch and Lysyanskaya give a proof for their credential system [21]. However, proving the security for the different combinations of the building blocks has not been done, although there is no reason to believe that such combinations would not be secure. Thus, the first step in proving security of our language framework is to provide precise security notions that capture the high-level security properties stated above and to prove that the composition of the cryptographic building blocks as imposed by our languages achieves those. Next, one would have to show that

the mapping of our languages to the concrete cryptographic realizations is sound. Only then one could attempt to formally prove that the security guarantees as stated above are fulfilled.

6 Conclusion

We presented a language framework enabling a unified deployment of Privacy-ABC technologies, in particular, of U-Prove and Identity Mixer. Our framework improves upon the state of the art [52,26] by covering the entire life-cycle of Privacy-ABCs, including issuance, presentation, inspection, and revocation, and by supporting advanced features such as pseudonyms and key binding. The framework offers a set of abstract concepts that make it possible for application developers to set up a Privacy-ABC infrastructure and to author policies without having to deal with the intricacies of the underlying cryptography.

In an upcoming companion paper, we demonstrate the soundness of our languages by providing a formal semantics that specifies the effects of issuing, presenting, verifying, inspecting, and revoking credentials on the user's credential portfolio and on the knowledge states of the involved parties. A complete description of our framework including the language description as well as the formal semantics is available as a technical report [14].

The proposed language framework has been implemented as part of the ABC4Trust project, where it will be rolled out in two pilot projects. Preliminary tests indicate that our language framework adds a noticeable but reasonable overhead to the cryptographic routines, comparable to the overhead incurred by, for example, XML Signature [51] with respect to the underlying signing algorithm.

Our language framework supports a number of different authentication mechanisms including the mentioned privacy-preserving ones but also standard mechanisms such as X.509. However, most of them will not support the full set of features but we are currently working on a protocol framework that allows the combination of different cryptographic mechanisms to address this.

Acknowledgements. The authors thank Bart De Decker, Robert Enderlein, Lan Nguyen, and the anonymous referees for their valuable comments. The research leading to these results was supported by the European Commission under Grant Agreement 257782 ABC4Trust.

References

1. Ardagna, C.A., Camenisch, J., Kohlweiss, M., Leenes, R., Neven, G., Priem, B., Samarati, P., Sommer, D., Verdicchio, M.: Exploiting cryptography for privacy-enhanced access control. J. of Comput. Secur. 18(1) (2010)
2. Ardagna, C.A., Cremonini, M., De Capitani di Vimercati, S., Samarati, P.: A privacy-aware access control system. J. Comput. Secur. 16(4) (2008)
3. Appel, A.W., Felten, E.W.: Proof-carrying authentication. In: ACM CCS 1999 (1999)
4. Au, M.H., Susilo, W., Mu, Y.: Constant-Size Dynamic *k*-TAA. In: De Prisco, R., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, pp. 111–125. Springer, Heidelberg (2006)
5. Bowers, K.D., Bauer, L., Garg, D., Pfenning, F., Reiter, M.K.: Consumable credentials in linear-logic-based access-control systems. In: NDSS 2007 (2007)

6. Belenkiy, M., Camenisch, J., Chase, M., Kohlweiss, M., Lysyanskaya, A., Shacham, H.: Randomizable Proofs and Delegatable Anonymous Credentials. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 108–125. Springer, Heidelberg (2009)
7. Bichsel, P., Camenisch, J., Preiss, F.-S.: A comprehensive framework enabling data-minimizing authentication. In: ACM DIM 2011 (2011)
8. Bonatti, P., Samarati, P.: A unified framework for regulating access and information release on the web. J. Comput. Secur. 10(3) (2002)
9. Boudot, F.: Efficient Proofs that a Committed Number Lies in an Interval. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 431–444. Springer, Heidelberg (2000)
10. Brands, S., Demuynck, L., De Decker, B.: A Practical System for Globally Revoking the Unlinkable Pseudonyms of Unknown Users. In: Pieprzyk, J., Ghodosi, H., Dawson, E. (eds.) ACISP 2007. LNCS, vol. 4586, pp. 400–415. Springer, Heidelberg (2007)
11. Brands, S.: Rethinking Public Key Infrastructures and Digital Certificates; Building in Privacy. MIT Press (2000)
12. Chaum, D., van Heyst, E.: Group Signatures. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 257–265. Springer, Heidelberg (1991)
13. Camenisch, J., Chaabouni, R., Shelat, A.: Efficient Protocols for Set Membership and Range Proofs. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 234–252. Springer, Heidelberg (2008)
14. Camenisch, J., Dubovitskaya, M., Lehmann, A., Neven, G., Paquin, C., Preiss, F.-S.: A language framework for privacy-preserving attribute-based authentication. Technical Report RZ3818, IBM (2012)
15. Camenisch, J., Krontiris, I., Lehmann, A., Neven, G., Paquin, C., Rannenberg, K., Zwingelberg, H.: H2.1 – ABC4Trust Architecture for Developers. ABC4Trust heartbeat H2.1 (2011)
16. Chaum, D.: Untraceable electronic mail, return addresses, and digital pseudonyms. Comm. of the ACM 24(2), 84–88 (1981)
17. Camenisch, J., Hohenberger, S., Lysyanskaya, A.: Balancing Accountability and Privacy Using E-Cash (Extended Abstract). In: De Prisco, R., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, pp. 141–155. Springer, Heidelberg (2006)
18. Camenisch, J., Kohlweiss, M., Soriente, C.: An Accumulator Based on Bilinear Maps and Efficient Revocation for Anonymous Credentials. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 481–500. Springer, Heidelberg (2009)
19. Camenisch, J., Kohlweiss, M., Soriente, C.: Solving Revocation with Efficient Update of Anonymous Credentials. In: Garay, J.A., De Prisco, R. (eds.) SCN 2010. LNCS, vol. 6280, pp. 454–471. Springer, Heidelberg (2010)
20. Camenisch, J., Kiayias, A., Yung, M.: On the Portability of Generalized Schnorr Proofs. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 425–442. Springer, Heidelberg (2009)
21. Camenisch, J., Lysyanskaya, A.: An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 93–118. Springer, Heidelberg (2001)
22. Camenisch, J., Lysyanskaya, A.: An Identity Escrow Scheme with Appointed Verifiers. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 388–407. Springer, Heidelberg (2001)
23. Camenisch, J., Lysyanskaya, A.: Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 61–76. Springer, Heidelberg (2002)
24. Camenisch, J., Lysyanskaya, A.: A Signature Scheme with Efficient Protocols. In: Cimato, S., Galdi, C., Persiano, G. (eds.) SCN 2002. LNCS, vol. 2576, pp. 268–289. Springer, Heidelberg (2003)

25. Camenisch, J., Lysyanskaya, A.: Signature Schemes and Anonymous Credentials from Bilinear Maps. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 56–72. Springer, Heidelberg (2004)
26. Camenisch, J., Mödersheim, S., Neven, G., Preiss, F.-S., Sommer, D.: A card requirements language enabling privacy-preserving access control. In: SACMAT 2010 (2010)
27. Camenisch, J.L., Shoup, V.: Practical Verifiable Encryption and Decryption of Discrete Logarithms. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 126–144. Springer, Heidelberg (2003)
28. Crockford, D.: The application/json media type for JavaScript Object Notation (JSON). Internet Engineering Taskforce (IETF) RFC 4627 (2006)
29. Damgård, I., Fujisaki, E.: A Statistically-Hiding Integer Commitment Scheme Based on Groups with Hidden Order. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 125–142. Springer, Heidelberg (2002)
30. Dodis, Y., Yampolskiy, A.: A Verifiable Random Function with Short Proofs and Keys. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 416–431. Springer, Heidelberg (2005)
31. Douceur, J.R.: The Sybil Attack. In: Druschel, P., Kaashoek, M.F., Rowstron, A. (eds.) IPTPS 2002. LNCS, vol. 2429, pp. 251–260. Springer, Heidelberg (2002)
32. Ferraiolo, D., Kuhn, R.: Role-based access control. In: NIST-NCSC 1992 (1992)
33. Fiat, A., Shamir, A.: How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987)
34. Garg, D., Bauer, L., Bowers, K.D., Pfenning, F., Reiter, M.K.: A Linear Logic of Authorization and Knowledge. In: Gollmann, D., Meier, J., Sabelfeld, A. (eds.) ESORICS 2006. LNCS, vol. 4189, pp. 297–312. Springer, Heidelberg (2006)
35. IBM Research Zurich Security Team. Specification of the identity mixer cryptographic library. Technical Report RZ3730, IBM (2010)
36. Identity Mixer, `http://idemix.wordpress.com/`
37. International Telecommunication Union. Abstract syntax notation one (ASN.1). ITU-T recommendation X.680 (2008)
38. Kirkpatrick, M., Ghinita, G., Bertino, E.: Privacy-preserving enforcement of spatially aware RBAC. In: IEEE Trans. on Dependable and Secure Computing 99 (2011) (PrePrints)
39. Lapon, J., Kohlweiss, M., De Decker, B., Naessens, V.: Analysis of Revocation Strategies for Anonymous Idemix Credentials. In: De Decker, B., Lapon, J., Naessens, V., Uhl, A. (eds.) CMS 2011. LNCS, vol. 7025, pp. 3–17. Springer, Heidelberg (2011)
40. Li, N., Grosof, B.N., Feigenbaum, J.: Delegation logic: A logic-based approach to distributed authorization. ACM TISSEC 6(1) (2003)
41. Li, J., Li, N., Winsborough, W.: Automated trust negotiation using cryptographic credentials. In: ACM CCS 2005 (2005)
42. Lysyanskaya, A., Rivest, R.L., Sahai, A., Wolf, S.: Pseudonym Systems (Extended Abstract). In: Heys, H.M., Adams, C.M. (eds.) SAC 1999. LNCS, vol. 1758, p. 184. Springer, Heidelberg (2000)
43. Nakanishi, T., Fujii, H., Hira, Y., Funabiki, N.: Revocable Group Signature Schemes with Constant Costs for Signing and Verifying. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 463–480. Springer, Heidelberg (2009)
44. Nguyen, L.: Accumulators from Bilinear Pairings and Applications. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 275–292. Springer, Heidelberg (2005)
45. Paci, F., Shang, N., Steuer Jr., K., Fernando, R., Bertino, E.: VeryIDX - A privacy preserving digital identity management system for mobile devices. In: Mobile Data Management (2009)
46. Pedersen, T.P.: Non-interactive and Information-Theoretic Secure Verifiable Secret Sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992)

47. Schnorr, C.-P.: Efficient signature generation by smart cards. Journal of Cryptology 4(3), 161–174 (1991)
48. Squicciarini, A.C., Bhargav-Spantzel, A., Bertino, E., Czeksis, A.B.: Auth-SL - A System for the Specification and Enforcement of Quality-Based Authentication Policies. In: Qing, S., Imai, H., Wang, G. (eds.) ICICS 2007. LNCS, vol. 4861, pp. 386–397. Springer, Heidelberg (2007)
49. Nguyen, L.: Accumulators from Bilinear Pairings and Applications. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 275–292. Springer, Heidelberg (2005)
50. Sandhu, R.S., Coyne, E.J., Feinstein, H.L., Youman, C.E.: Role-based access control models. IEEE Comput. 29(2) (1996)
51. Shirasuna, S., Slominski, A., Fang, L., Gannon, D.: Performance comparison of security mechanisms for grid services. In: GRID 2004 (2004)
52. Microsoft U-Prove, `http://www.microsoft.com/uprove`
53. Verheul, E.R.: Self-Blindable Credential Certificates from the Weil Pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 533–551. Springer, Heidelberg (2001)
54. Wang, L., Wijesekera, D., Jajodia, S.: A logic-based framework for attribute based access control. In: ACM FMSE 2004 (2004)
55. Winsborough, W., Seamons, K., Jones, V.: Automated trust negotiation. In: DISCEX 2000 (2000)
56. OASIS. eXtensible Access Control Markup Language (XACML) Version 2.0 (2005)

Efficient Selective Disclosure on Smart Cards Using Idemix*

Pim Vullers[1,**] and Gergely Alpár[1,2,***]

[1] Institute for Computing and Information Sciences,
Radboud University Nijmegen, The Netherlands
{p.vullers,g.alpar}@cs.ru.nl
[2] TNO Information and Communication Technology, The Netherlands

Abstract. In this paper we discuss an efficient implementation for selective disclosure of attribute-based credentials on smart cards. In this context we concentrate on the implementation of this core feature of IBM's Identity Mixer (Idemix) technology. Using the MULTOS platform we are the first to provide this feature on a smart card. We compare Idemix with Microsoft's U-Prove technology, as the latter also offers selective disclosure of attributes and has been implemented on a smart card [10].

Keywords: selective disclosure, attribute-based credentials, smart card, attributes, idemix, u-prove, multos.

1 Introduction

The world is moving into a digital era. Many people spend time on the internet, not just for fun or gathering information, but also for shopping and banking. Not only do our activities happen in the digital world, existing systems are also moving to digital alternatives. Train tickets are being replaced by electronic public transport cards. Identity documents, such as passports, are equipped with chips to hold digital copies of the identity data printed on the document and sometimes even additional information is stored there such as fingerprints or other biometric data.

Unfortunately, most such systems use a simple approach to identify entities; they just attach a unique number to them. While this is convenient for bookkeeping, it also has a big drawback with respect to privacy. Using these unique identifiers, it is easy to trace the user. For example, not only might it be possible

* The work described in this paper has been supported in part by the European Commission through the ICT programme under contract ICT-2007-216676 ECRYPT II.

** Sponsored by Trans Link Systems/Open Ticketing.

*** Partly supported by the research program Sentinels as project 'Mobile IDM' (10522). Sentinels is being financed by Technology Foundation STW, the Netherlands Organisation for Scientific Research (NWO), and the Dutch Ministry of Economic Affairs.

S. Fischer-Hübner et al.(Eds.): IDMAN 2013, IFIP AICT 396, pp. 53–67, 2013.

to track user activities on the world wide web, but real world actions could also be traced through the use of public transport cards or digital identity documents.

These unique identifiers are used to identify entities during authentication and/or authorisation, but actually in most use cases identification is not necessary. For instance, when you want to buy liquor, a merchant only needs to verify that you are of a certain age. The same holds when boarding a train; the system only needs to know whether or not you are allowed to do so, and there is no direct need for the system to know exactly who you are.

A more privacy-friendly approach is possible by using attribute-based credentials. Instead of providing lots of identity information to the service provider, the user can now just provide the required attributes, such that the service can be accessed without the user revealing his identity.

In this paper we use the Identity Mixer (Idemix) technology [7,8,9] developed by IBM Research to implement attribute-based credentials. This system allows the user to receive a signed list of attributes from a trusted party which can then be used to convince a service provider. A core feature of this technology, *selective disclosure*, enables a user to control which attributes from this list get revealed to the service provider.

Having public transport cards and identity documents in mind, we focus on smart card implementations. We use cards running the MULTOS platform that offers an API suitable for implementing cryptographic protocols. Our prototype achieves the best performance for Idemix on a smart card thus far, with running times which are acceptable for on-line, and certain off-line scenarios.

While others have implemented Idemix on a smart card [5,12], we are the first to provide the selective disclosure functionality. We compare our implementation against an implementation [10] of Microsoft's U-Prove system [6,11] which offers similar functionality, and currently provides the best smart card performance. The benefit of using Idemix is its multi-show unlinkability property, which allows a single credential to be used multiple times, whereas U-Prove only provides single-show unlinkability and hence requires multiple credentials to provide anonymity instead of just pseudonymity.

2 Attributes and Credentials

Within this article, an attribute will be understood as some property of a person. One can think of "over 18", "male", or "owner of bank account 1234" as examples of attributes. Informally, a person's identity can be seen as the collection of all attributes that hold for this person. In practice, many transactions can be based on a subset of attributes, such as buying a bottle of whiskey by verifying the "over 18" attribute while other attributes are irrelevant.

There are several cryptographic systems for dealing with attribute-based identities. Typically these systems distinguish credentials and attributes. Informally, a credential is a *cryptographic container of attributes*. As a first approximation, one can think of a credential as depicted in Figure 1.

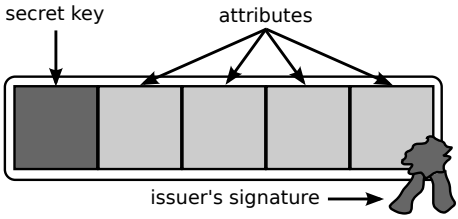


Fig. 1. A first look at an attribute-based credential with four attributes

2.1 Issuance and Selective Disclosure

Credentials are *issued* and *verified*, whereas attributes can be *disclosed* during verification. In the issuance procedure, an issuer and a user together create a new credential. First the user authenticates to the issuer in some reliable but unspecified manner (which may be face-to-face). Once the authentication succeeds, the issuer collects attributes for this user from trusted databases. The user and issuer then carry out a cryptographic protocol in which the attributes are combined into a credential signed by the issuer. The resulting credential contains attributes concerning the user and also his/her secret key.

The fact that the attributes hold for the owner of a credential is guaranteed both by the issuer's signature and by the embedded secret key of the owner. The secret key in the credential is only known by the credential owner and plays an essential role in verification of the credential. It ensures that a credential cannot be transferred from one user to another.

A user may have several credentials, each asserting some collection of attributes. When requesting a service from a service provider, the user is required to authenticate using one (or more) of his/her credentials. In the verification process the user can choose to only provide certain credentials; also, given a specific credential, the user may choose to reveal only a subset of attributes in the credential. By doing this, authentication becomes more privacy friendly. This latter process is called selective disclosure, involving a verification protocol in which only a subset of the credential attributes is revealed to the verifier while the other attributes are only proved to be present in the credential. This allows a user to reveal only the necessary attributes and prove that the credential belongs to him/her. The service provider can verify all information that has been sent, including the issuer's signature.

The roles of a service provider and an issuer can also be combined: after checking one credential, a new one can be issued. For instance, after verifying an "over 18" attribute from an identity credential, a liquor shop might choose to issue a loyalty credential.

In this paper we stick to a simplistic approach to selective disclosure in which an index set $\mathcal{D}$ determines the information to be revealed. Selective disclosure is sometimes defined more generally. We consider only the projection function $F_{\mathcal{D}}$ that is determined by a disclosing subset $\mathcal{D}$ of attribute indexes. For example, if there are four attributes and $\mathcal{D} = \{1, 3\}$, then $F_{\mathcal{D}}(a_1, a_2, a_3, a_4) = (a_1, a_3)$.

However, a more general definition would allow F to be any function of the attributes, for example, $F(a_1, a_2, a_3, a_4) := (a_1 > 18, a_2 + a_3)$.

2.2 Security and Privacy

The cryptographic nature of the credential-as-container concept includes the following four security aspects.

- The issuer's digital signature ensures *authenticity*: the credential originates from the issuer, and this issuer asserts that the attributes hold for the person.
- This signature also guarantees *integrity*: the attributes contained in the credential have not been altered since they were issued.
- A credential is *non-transferable* as it is bound to the secret key of the person involved in the issuance protocol. This secret key should be well protected, for instance via storage in the secure memory of a smart card with a PIN.
- A credential *hides* its content, so it does not reveal the attributes it contains.

Furthermore, a credential protects the privacy of its owner through the following cryptographic properties.

- *Issuer unlinkability* ensures that any information gathered during issuing cannot be used to link a verification of the credential to its issuance.
- *Multi-show unlinkability* guarantees that when a credential is verified multiple times, these sessions cannot be linked.

The privacy of users is protected by these unlinkability properties even if the credential issuer and all verifiers collude. As briefly reviewed below, a variety of ways of achieving the properties above have been proposed.

Blindly signed single-show credentials[1] are credentials in which the issuance involves creating a blind signature which conceals the resulting credential from the issuer. Therefore, the verification instances of this credential cannot be related to the issuing phase. Examples of this approach are Brands credentials [6], which are used in Microsoft's U-Prove system [11], and Baldimtsi and Lysyanskaya credentials [3].

Randomisable credentials employ special cryptographic techniques enabling certain credential structures to be randomised using blinding factors while preserving their verifiability. In such systems, users can randomise their credentials before they are verified. Such credentials have been proposed by Verheul [14], and implemented by Batina et al. [4], although this technique currently only supports binary attributes; for example, instead of a descriptive string of nationality, the answer to the question "Is French?" is stored.

Zero-knowledge proofs allow a user to prove ownership of a credential without revealing the credential itself. Since the verifier does not see the credential, verification instances are unlinkable and they also cannot be related to the issuing procedure. Camenisch and Lysyanskaya [7,8] combine such proofs with randomisation; their technique was further developed and became IBM's Identity Mixer (Idemix) [9].

[1] Multi-show unlinkability for these schemes can be realised by issuing multiple credentials for the same set of attributes which can later be verified independently.

3 Technologies for Selective Disclosure

In the following sections we briefly study the cryptographic techniques how credentials are constructed and used in U-Prove and in Idemix. In both techniques the concept of *"possessing"* a credential is equivalent to *"knowing"* its secret key. Furthermore, to be able to carry out all the computations, the user (or the device in the user's control) has to know the attributes in the credential.

3.1 Selective Disclosure with U-Prove

U-Prove [6,11] relies on the discrete logarithm (DL) cryptographic assumption. Informally, the DL assumption states that given a group and a generator, it is hard to find the exponent that the generator has to be raised to get a randomly given group element. We give a simplified description of U-Prove below using multiplicative notation (for a full specification we refer to [11]):

- A group is required in which the DL problem is hard. For instance, the following multiplicative group:
  - p, q primes (usual DL setup: $q|p-1$) and generator g in $\mathbb{Z}_p^*$ of order q, i.e., $(1, g, g^2, g^3, \ldots, g^{q-1}) \pmod p$ forms a cyclic multiplicative group of q elements. The group can then be described by (p, q, g)
- Key generation is only performed by the issuer, the user obtains distinct secret keys corresponding to each credential during the issuing protocol.
  - Issuer: generate $x, y_1, \ldots, y_l \in_R \mathbb{Z}_q^*$, compute $h \equiv g^x \pmod p$ and $g_1 \equiv g^{y_1} \pmod p, \ldots, g_l \equiv g^{y_l} \pmod p$, store $sk_I = (x, y_1, \ldots, y_l)$, and publish $pk_I = (h, g_1, \ldots, g_l)$, where l denotes the number of attributes.
- Issuing a credential is performed using an interactive protocol in which an issuer blindly signs a commitment. The attributes $(\alpha_1, \ldots, \alpha_l)$ are assumed to be common input to the issuer and the user; the resulting credential and the corresponding secret key α are only known to the user.
  - The resulting credential is $Cred = (h', \sigma)$ and the user's secret key is α. Credential $Cred$ includes a commitment $h' = g^\alpha \prod_1^l g_i^{\alpha_i} \pmod p$, which information-theoretically hides the attributes (because of the secret key α) and computationally binds the user to them, and a signature $\sigma = (c', r')$ of the issuer on this commitment, such that $c' = \mathcal{H}\left(h' \| g^{r'}(h \cdot h')^{-c'}\right)$.
- Selective disclosure of attributes is achieved using an interactive protocol between the user and a service provider (verifier). The credential $Cred$, a disclosing index set $\mathcal{D}$ and the corresponding attributes $(\alpha_i)_{i \in \mathcal{D}}$ are revealed by the user. Furthermore the user provides a partial commitment $R = g^\alpha \prod_{i \notin D} g_i^{\alpha_i} \pmod p$ to the unrevealed attributes.
  - First, the verifier checks the signature σ in credential $Cred$:

$$c' \stackrel{?}{=} \mathcal{H}\left(h' \| g^{r'}(h \cdot h')^{-c'}\right).$$

- Second, the user proves knowledge of all secret values in the commitment h' in a zero-knowledge protocol, a proof of knowledge

$$pr = PK\{(\alpha, (\alpha_i)_{i \notin D}) : h' \cdot (g_i^{-\alpha_i})_{i \in D} \equiv R \pmod{p}\}.$$

  Without revealing any additional information, it demonstrates that the user knows all the secret values in the partial commitment R: the secret key α and the hidden attributes $(\alpha_i)_{i \notin \mathcal{D}}$. Consequently, the verifier gets convinced that the user possesses the credential $Cred$ that contains the revealed attributes $(\alpha_i)_{i \in \mathcal{D}}$.

3.2 Selective Disclosure with Idemix

Idemix [7,8,9] relies on the Strong RSA assumption. Informally, in a group where this assumption holds, it is computationally infeasible to find *any* root of a given group element. (The RSA assumption is weaker and therefore it provides stronger security as it assumes that a *given* root of a given element is hard to find.)

Unlike in U-Prove, users in Idemix need only one secret key corresponding to *all* credentials and it is called a master key. Optionally, a user may choose to use several master keys but that does not influence the security and privacy properties of the scheme. Furthermore, the Idemix selective disclosure is a hybrid scheme in the sense that it blends all three unlinkability approaches we discussed in Section 2.2: the issuing is a blind signature that does not allow the issuer to learn the master key and the resulting signature; the user can partly randomise a credential in the verification protocol; the user proves to the verifier using zero-knowledge techniques that she possesses a valid credential instead of releasing any information about it.

Idemix's algorithms and parameters can be described as follows (for a detailed description we refer to [9]):

- A group has to be generated in which the Strong RSA problem is hard:
  - Given a special RSA modulus: $n = pq$ where $p = 2p' + 1$, $q = 2q' + 1, p'$, and q' are large prime numbers. The set QR_n of quadratic residues $\pmod{n}$ (i.e. every element r for which there exists some integer x such that $x^2 \equiv r \pmod{n}$) establishes a Strong RSA group.
- Key generation for the issuer and the generation of the system group happen to be the same algorithm here. The reason for that is that the prime components p, q of n determine the group uniquely and they form the issuer's secret key as well.
  - Generate a special RSA modulus $n = pq$. p and q are kept secret as the secret key sk_I of the issuer.
  - Choose, uniformly at random, $R_0, R_1, \ldots, R_l, Z, S \in QR_n$. These values form the set of public parameters together with n.
- Generation of a master key for the user.
  - The user chooses randomly a large integer α (in our case 256 bits) as his master key.

- Issuing a credential. Like a U-Prove credential, an Idemix credential is also a signature on a commitment. The issuing procedure is an interactive protocol between the issuer and the user. Both participants need their secret keys, sk_I and α.
  - The resulting credential includes a commitment $R = R_0^{\alpha} \prod_1^l R_i^{\alpha_i} \pmod n$ to the attributes $\alpha_1, \ldots, \alpha_l$ and the issuer's signature $A = \left(\frac{Z}{S^v \cdot R}\right)^{1/e} \pmod n$ on the commitment. A can only be computed by knowing the divisors of n, that is, by knowing sk_I; therefore, it can only be created by the issuer. The resulting signature is (A, e, v) where e is a prime number chosen by the issuer and v is computed together by the participants but known only to the user. (So, the issuer can store A and e but he doesn't learn α and v.)
- Selective disclosure of attributes is achieved using an interactive protocol between a user and a verifier.
  - Without giving all the details about credential showing, we show how a signature can be randomised. Let r be a random integer from a certain interval. Provided that $A' := A \cdot S^{-r} \pmod n$ and $v' := v + er$,

$$A'^e S^{v'} R \equiv A^e S^{-er} S^v S^{er} R \equiv A^e S^v R \equiv Z \pmod n.$$

    Therefore, a randomised signature (A', e, v') is also a valid signature on R. Note that e should not be revealed during a verification protocol as it provided linkability.
  - Besides the revealed attributes, a user sends A' from the randomised signature and a proof of knowledge pr of all undisclosed attributes and required parameters to the verifier.

$$pr = PK\{(e', v', \alpha, (\alpha_i)_{i \notin \mathcal{D}}) : Z \cdot \left(R_i^{-a_i}\right)_{i \in \mathcal{D}} \equiv \pm A'^{e'} S^{v'} R_0^{\alpha} \prod_{i \notin \mathcal{D}} R_i^{a_i} \bmod n\}$$

  - The verifier checks the validity of the proof pr. (Actually, the prover proves knowledge of a representation of $Z \cdot \left(R_i^{-a_i}\right)_{i \in \mathcal{D}}$ with respect to A', S, R_0, and $(R_i)_{i \in \mathcal{D}}$.)

3.3 Similarities and Differences

As we saw in the previous sections, there are many cryptographic differences between U-Prove and Idemix. Not only do the cryptographic assumptions differ in the two schemes, but also the principle of verification. While a U-Prove credential is revealed every time it is used, an Idemix credential is never revealed to a verifier. Therefore, unlinkability can only be achieved using as many different U-Prove credentials as their verification instances. The same functionality however, can be achieved with only one Idemix credential. Furthermore, while U-prove credentials have different random secret keys calculated during each issuing protocol, Idemix credentials belonging to the same user can operate with one master key. As an application of this feature, several credentials having the same secret key can be proved to belong to the same user.

Besides these differences, there are similarities as well. Both credentials can be abstractly seen as signed commitments. A commitment in this case is basically the product of generators with attributes and a secret key as their exponents. As a result of this resemblance, selective disclosure can be done similarly. Having received some attributes, the verifier can reproduce a partial product from the commitment. Then he can combine it with the presented proof of knowledge about all the other attributes. Only a user having a valid credential can provide such a proof.

4 Idemix on Smart Cards

The main objective of our research is to assess how well privacy-friendly protocols perform when run on a smart card. Hence implementing our prototypes requires an open smart card platform that also provides the necessary cryptographic hardware support. Previous research [13,10] clearly shows that purely software based prototypes do not offer sufficient performance for realistic use, while proper hardware support makes the prototypes practically usable. In this case we used an Infineon SLE78 chip[2] running the MULTOS platform. This platform offers an API which exposes the (hardware supported) modular arithmetic operations required to implement technologies like Idemix and U-Prove.

4.1 Smart Cards

Regardless of the software platform operating the card, all smart cards provide the same external functionality. A smart card is an embedded device that communicates with the environment through Application Protocol Data Units (APDUs) – byte arrays formatted according to the ISO7816 specification [1]. Most notably, the APDUs constrain the communication payload to roughly 256 bytes in each direction for a single APDU exchange. The permanent storage of the card (EEPROM memory) is considered highly secure, accessible only through the APDU commands offered by the application, which in turn are subject to any authentication and secure messaging requirements that the card application may impose.

4.2 The MULTOS Platform

The goal of the MULTOS platform is to provide a secure hardware independent execution platform for smart cards. To this end, they developed a specification for the execution and memory models, explained in more detail below, that all MULTOS implementations must provide. Besides this mandatory part of the specification there are also a number of optional elements, mostly concerning cryptographic functionality that may or may not be available on a specific hardware platform. An overview of which functionality is provided by which card can be found in the MULTOS Implementation Reference [2][3].

[2] This is the successor to the Infineon SLE66 chip used by Mostowski and Vullers [10].
[3] Our card has an M3 generic (ML3-36K-R1) implementation by Multos International.

Execution Model. Applications on a MULTOS card are executed in a virtual machine, called the Application Abstract Machine (AAM). The functionality of this virtual machine is defined by the MULTOS specification to assure that applications are portable, that is, independent of the actual chip used[4]. The AAM is a stack machine that interprets instructions from the MULTOS Executable Language (MEL).

Memory Model. The AAM virtual machine provides each application with its own memory space. Within an application the code space, residing on non-volatile EEPROM storage, and data space, divided over EEPROM (for persistent storage) and volatile RAM, are handled independently of each other. Code is executed while data is manipulated. The memory of an application is protected by a strong firewall. This means that applications cannot access each others memory. The data of an application is divided over three distinct memory areas, listed below.

Static memory is the non-volatile storage for an application. It is private to the application and cannot be accessed by the terminal or any other application. MULTOS offers mechanisms to avoid corruption of the static memory area such that this data remains consistent.

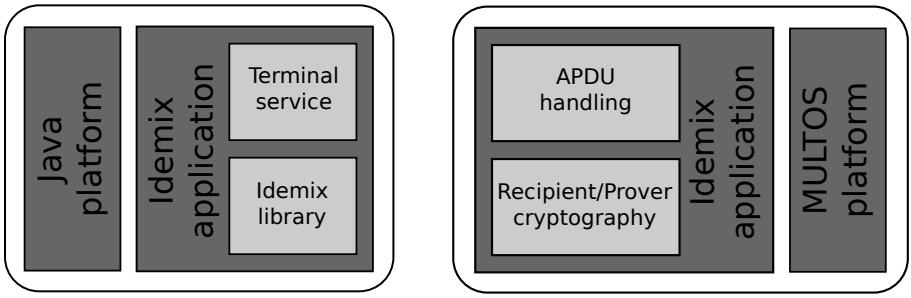
Public memory is the volatile input/output buffer for an application. Incoming command APDUs are held in public memory and outgoing response APDUs are placed here. This buffer is also used to pass information from one application to another when delegation is used. MULTOS guarantees that data in this memory remains private to the running application until it exits or delegates to another application. This means that it can be used as temporary workspace.

Dynamic memory is the volatile storage for an application. It is used to store session data, if any. The size of the session data area is fixed when an application is loaded onto a card and it depends on the amount of variables declared. Furthermore, the dynamic memory contains the stack, which is the application's work area. As mentioned before, the AAM operates as a stack machine, which means that this memory area is used to perform many functions (and provide input for these functions). The maximum size of the stack is fixed by the amount of dynamic memory available. Therefore, applications will need to ensure that their use of dynamic memory does not exceed the limit imposed by the chip [2] on which the application will reside.

4.3 System Architecture

Our system consists of two parts, depicted in Figure 2, a terminal (a) which interacts with a card (b) using APDUs. The terminal application is written in

[4] Application portability can be limited due to specific memory requirements or dependencies on optional parts of the MULTOS specification.



(a) Terminal (b) Card

Fig. 2. System architecture for Idemix on a smart card

Java and uses the Idemix cryptographic library[5] provided by IBM Research. We created an extension to this library, the terminal service, which takes care of all smart card specifics. The service implements the user roles of the Idemix issuance and verification protocols, described by the `Recipient` and `Prover` interfaces respectively. These interfaces are implemented by translating all Java method calls from the library into the corresponding APDU commands and converting the Java data types to raw byte arrays, suitable for APDU communication.

The Idemix application on the card takes care of handling the incoming APDUs and storing the values into the internal data structures. While handling the communication with the terminal is the largest part of the application, the main part is the implementation of the cryptographic operations for the Idemix protocols. This allows the card to perform the user roles without depending on the terminal for any computations or proof generation. The only thing the terminal is responsible for is providing the data in the correct format.

4.4 Smart Card Implementation

Just as Mostowski and Vullers [10] with their U-Prove implementation, we have chosen to use the MULTOS C interface to develop our prototype implementation of Idemix. The programming environment is convenient for smart card programming and allows us some more flexibility for memory management. It will be explained below why this is crucial.

Implementing the Idemix specification did not turn out to be that hard in the beginning, the available API makes it easy to implement the cryptographic protocol. In principle, it is a direct translation from the mathematical description to API calls. The only API restriction we came across was that the `ModularExponentiation` function does not accept exponents larger than the modulus size. For Idemix this only involves exponentiations with base S (see details in Section 3.2). Hence we added a function `SpecialModularExponentiation`

[5] Available for download at `https://prime.inf.tu-dresden.de/idemix/`

which implements the same method as used by Bichsel et al. [5], that is, splitting one exponentiation up into two[6] exponentiations and one multiplication.

Our initial implementation only used static memory to store the variables. This allowed us to work without thinking about which memory segment to use. The drawback of this approach was a bad performance caused by the EEPROM memory which takes a long time, compared to RAM, to write new values.

Once we had a functionally correct implementation, we started to optimise. This was done by moving the buffer, which stores the intermediate results for larger computations, to the public memory. The next step was to move the session variables to the dynamic memory, which required careful organisation due to the limited amount of available storage. However, when the dynamic memory use increased the stack-based execution model started to cause trouble.

Initially, the stack could use the full size of the dynamic memory, such that we had sufficient space to use functions and put the (relatively) large input values on the stack. When using the C-interface, the compiler takes care of managing the stack and putting input values on it when an instruction needs this. However, this makes it difficult to get an idea on how much space the stack actually needs. By trial and error, we discovered that we used quite some amount of memory for the stack, which left us with only limited amount of space for session variables.

To improve this situation, we reduced the number of function calls by inlining some convenience functions. We also switched to using global variables instead of function parameters, such that when we use a function, it does not require much space on the stack. To get the last few, often used, variables into RAM, we decided to split up some computations into smaller parts such that the values to be put on the stack also get smaller. For example, additions can be computed using addition with carry, and multiplications using grade-school multiplication. This adds a number of extra operations, but it is worth the memory gain which results in improved performance.

Finally, we translated most parts of the cryptographic code to MEL assembly which allowed us to optimise the use of stack and reduce the amount of memory operations during calculations. This was required since the provided C-functions moved the values from the stack to the variable locations, while they are needed again in the next operation. By doing this we could reduce the amount of memory required for intermediate values which in the end allowed us to get all necessary variables in RAM.

The resulting code can be found in the `idemix_multos` GitHub repository at `https://github.com/credentials/`. The repositories listed at this page are used within the IRMA project (`https://irmacard.org/`) which aims to develop an infrastructure for using attribute-based credentials on smart cards. A pilot, with government support, will be launched early 2013.

[6] This method actually requires three exponentiations, but we can precompute one exponentiation during initialisation, since we only need this method for the base S.

5 Results and Comparison

We mainly compare our results with the U-Prove implementation by Mostowski and Vullers [10] as this implementation offers similar functionality using the same modulus size (1024 bits) and the same platform. During initial development we even used the same SLE66-based cards, but in the end we managed to get newer SLE78-based cards which we currently use. For the conclusions this makes no difference since already with our SLE66 optimised prototype we got similar results, the SLE78 implementation only improved the running times.

There are two important performance measures: the time it takes to issue a new credential to the card, and the running time of the verification protocol.

5.1 Credential Issuance

For issuance, we measured the running time of the protocol and determined how much time was actually spent on computations and which part was used to transfer and store the values (marked as overhead; see Figure 3). These values offer a clear improvement over the U-Prove issuance times from [10], which take 3623 and 5489 ms for 2 and 5 attributes respectively. Not only are the absolute values better, the additional time it takes to issue more attributes is less than with the U-Prove implementation.

Sterckx et al. [12] implement the Direct Anonymous Attestation (DAA) protocol, which is derived from the Idemix protocols on a Java Card. With a 1024 bits modulus they achieve a running time of 2.4 seconds of which 19% is overhead, which gives a computation time of approximately 1.9 seconds. This is good, but unfortunately the DAA protocol does not support any attributes as it is just targeted at anonymous authentication and hence only uses a secret key.

Bichsel et al. [5] from the Idemix team at IBM Research Zürch also implemented a variant of the DAA protocol on a Java Card. They report a running time of 7.4 seconds for a modulus size of 1280 bits, which is larger than the 1024 bits we used. It is unclear, however, which transaction time they measured. But again, this implementation does not include any attributes.

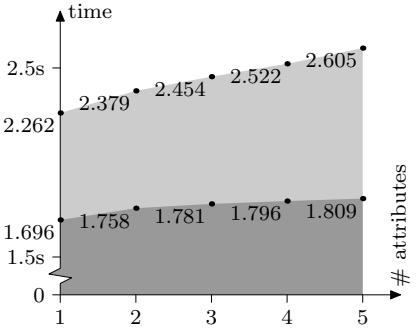


Fig. 3. Credential issuance times (■: computations, ■: overhead; 1024 bits)

5.2 Selective Disclosure of Attributes

For selective disclosure, we measured the running times of four configurations (see Figure 4). These configurations have been chosen because two is the smallest number of attributes for which selective disclosure makes sense and five is the largest number of attributes used in our pilot project. By comparing these graphs, it is clear that each attribute that is not disclosed adds roughly 100ms of computation time, whereas the amount of work per undisclosed attribute remains the same.

Comparing these measurements with those from the U-Prove implementation [10], it is clear that U-Prove is more efficient with total running times below 900ms and less. At this point the multi-show unlinkability property of the Idemix technology has an effect on the performance. To achieve this property, the issuer signature is partly randomised and for the remaining part proved using a

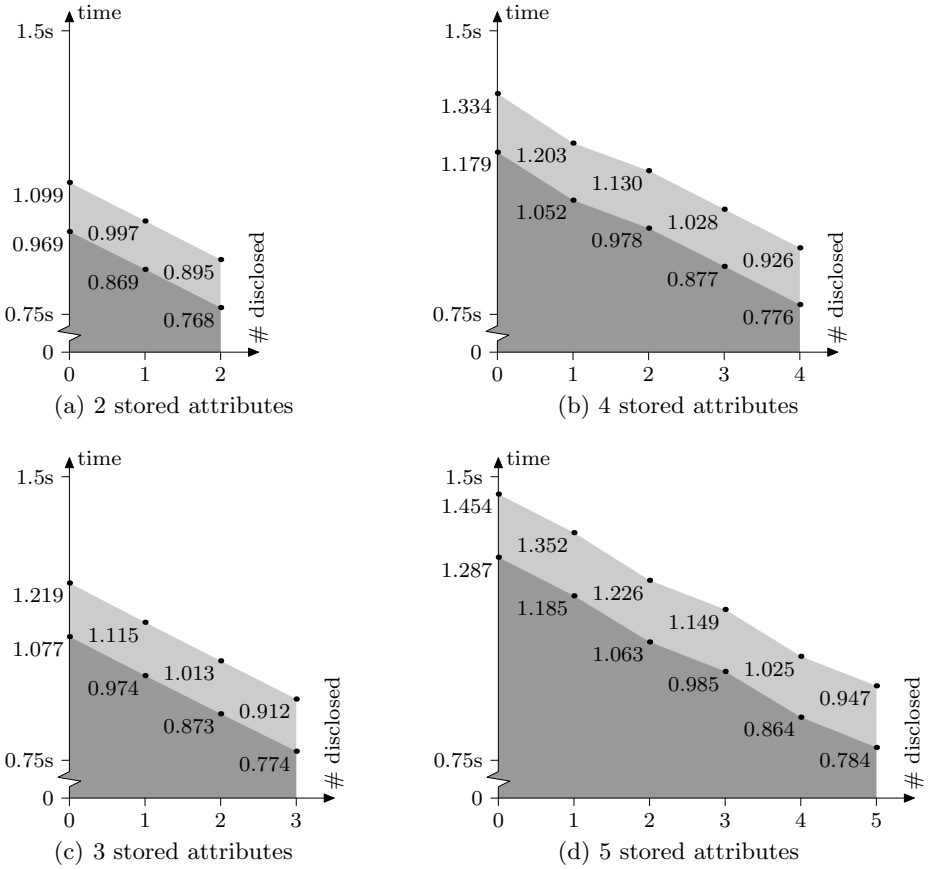


Fig. 4. Attribute proving times (▮: computation, ▮: overhead; 1024 bits modulus)

zero-knowledge proof. This clearly has its drawback compared to U-Prove that only requires computation to hide the undisclosed attributes.

While the multi-show unlinkability property has a negative effect on the running time for Idemix, it requires less storage on the card then U-Prove. This is due to the fact that the card has to store multiple U-Prove tokens to achieve multi-show unlinkability. Given that storage space is rather limited on smart cards[7], this is a huge benefit of the Idemix technology. In the end this integrated multi-show unlinkability property also settled our debate for using Idemix in favour of U-Prove as the technology to be used within the IRMA project.

Comparing our work with the DAA implementations by Sterckx et al. [12] and Bichsel et al. [5] makes no real sense since they do not offer selective disclosure of attributes. It is, however, clear that our implementation provides a performance improvement over these implementations since we can even hide the secret key and two attributes in 1.1 seconds whereas Sterckx et al. already need 4.2 seconds to hide only the secret key.

6 Final Remarks

In this paper we demonstrated that Idemix's selective disclosure can be efficiently implemented on a smart card. Although the running time of the issuing and verification protocols restricts the range of use cases, we can be optimistic already about these results.

- Additional attributes only increases the running time with a small factor.
- A single credential is sufficient to preserve both unlinkability properties.

Our implementation only offers a 1024 bits security level. We have chosen this modulus size since it is lowest acceptable level security wise, whereas it is the highest acceptable level performance wise. Mostowski and Vullers [10] already showed that using a 2048 bits modulus more than doubles the computation time for the primitive operations. This is not even taking the shortage of RAM, due to larger values, into account.

To the best of our knowledge the only other smart card implementation of attribute-based credentials besides what was already mentioned is by Batina et al. [4] who implement Verheul's self-blindable credentials [14]. Their implementation is faster than our Idemix implementation and also offers multi-show unlinkability. However, their credentials only contain a single binary attribute as explained in Section 2.2, and hence do not provide selective disclosure similar to Idemix or U-Prove.

Due to the multi-show unlinkability feature of the Idemix protocol this implementation has been selected to be used in a pilot project. The goal of this project is to gain more experience in actually using these kinds of privacy-preserving technologies and in their usability features on smart cards.

[7] A typical smart card only has 36 to 144 KB of EEPROM available for storing application data.

Acknowledgements. We are grateful to Patrik Bichsel, for making the necessary modifications to the Idemix library, to Wouter Lueks, for his help with the performance tests, and to Bart Jacobs, Jaap-Henk Hoepman and the anonymous reviewers for their valuable comments which helped to improve this work.

References

1. ISO 7816-4 Identification cards – Integrated circuit cards – Part 4: Organization, security and commands for interchange. ISO, Geneva, Switzerland (2005)
2. MULTOS implementation report. Tech. Rep. MAO-DOC-TEC-010 v2.4, MAOSCO Limited (2012)
3. Baldimtsi, F., Lysyanskaya, A.: Anonymous credentials light. IACR Cryptology ePrint Archive 2012, 298 (2012)
4. Batina, L., Hoepman, J.-H., Jacobs, B., Mostowski, W., Vullers, P.: Developing efficient blinded attribute certificates on smart cards via pairings. In: Gollmann, D., Lanet, J.-L., Iguchi-Cartigny, J. (eds.) CARDIS 2010. LNCS, vol. 6035, pp. 209–222. Springer, Heidelberg (2010)
5. Bichsel, P., Camenisch, J., Groß, T., Shoup, V.: Anonymous credentials on a standard Java Card. In: CCS 2009, pp. 600–610. ACM (November 2009)
6. Brands, S.A.: Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy. MIT Press, Cambridge (2000)
7. Camenisch, J., Lysyanskaya, A.: An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 93–118. Springer, Heidelberg (2001)
8. Camenisch, J., Lysyanskaya, A.: A signature scheme with efficient protocols. In: Cimato, S., Galdi, C., Persiano, G. (eds.) SCN 2002. LNCS, vol. 2576, pp. 268–289. Springer, Heidelberg (2003)
9. IBM Research Zürich Security Team: Specification of the Identity Mixer cryptographic library, version 2.3.4. Tech. rep., IBM Research, Zürich (February 2012)
10. Mostowski, W., Vullers, P.: Efficient U-prove implementation for anonymous credentials on smart cards. In: Rajarajan, M., Piper, F., Wang, H., Kesidis, G. (eds.) SecureComm 2011. LNICST, vol. 96, pp. 243–260. Springer, Heidelberg (2012)
11. Paquin, C.: U-Prove cryptographic specification v1.1. Tech. rep., Microsoft Corporation (February 2011)
12. Sterckx, M., Gierlichs, B., Preneel, B., Verbauwhede, I.: Efficient implementation of anonymous credentials on Java Card smart cards. In: WIFS 2009, pp. 106–110. IEEE (September 2009)
13. Tews, H., Jacobs, B.: Performance issues of selective disclosure and blinded issuing protocols on Java Card. In: Markowitch, O., Bilas, A., Hoepman, J.-H., Mitchell, C.J., Quisquater, J.-J. (eds.) WISTP 2009. LNCS, vol. 5746, pp. 95–111. Springer, Heidelberg (2009)
14. Verheul, E.R.: Self-Blindable Credential Certificates from the Weil Pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 533–550. Springer, Heidelberg (2001)

Identity Management and Integrity Protection in Publish-Subscribe Systems

Anders Fongen and Federico Mancini

Norwegian Defence Research Establishment
anders.fongen@ffi.no

Abstract. The use of Identity Management (IdM) may leverage the trust in a distributed Publish-Subscribe (PubSub) system. An IdM provides mutual authentication between publishers, subscribers and message routers, enforces access control on message delivery and integrity control of message content. Access control is also a means to reduce traffic in a PubSub network since unauthorized message traffic will not be forwarded. A framework for providing identity management in a generic PubSub systems is presented and analyzed in this paper. The trust in the system relies to some extent on the use of hardware units for the protection of software integrity.

1 Introduction and PubSub Security Principles

The integration of a security model into a Publish-Subscribe distribution system is the objective of this paper. Its implementation employs services from an Identity Management System and the Trusted Platform Module (TPM) hardware unit.

In a *Publish-Subscribe* (PubSub) system, the information is not disseminated based on the addresses (or any individual property) of the receivers, but on *subscriptions*, through which the receivers specify their interest for information. Two forms of subscriptions are prevalent: One is to associate the information elements with a set of *topics* which are used as selection criteria in subscriptions, the other is to specify conditions for information of interest (like `''currency==USD''` or `''temp>90F''`). In this paper the first form will be used since this is the most widespread approach.

The concept of *Identity Management* (IdM) introduces services for assured management of identities and their associated attributes: *Key pairs* for signing, encryption and authentication, *roles* for access control purposes [1] and other *attributes* that can support application services. For the remainder of this paper, the term *Attributes* also includes the concept of *Roles*.

Attributes may express the authorization of an entity and they should therefore be attested by a trusted party called an *Identity Provider* (IdP). The data structure used for that purpose is the *Identity Statement* (IS), where the identity, keys and attributes are stored and protected by the signature of the trusted IdP. The IS is somewhat similar to a Public Key Certificate, but an IS has a shorter life time and contains attributes as well as the public key. The short lifetime of an IS eliminates the need for a revocation arrangement.

S. Fischer-Hübner et al.(Eds.): IDMAN 2013, IFIP AICT 396, pp. 68–82, 2013.

The PubSub distribution principle has been demonstrated to be highly scalable. It allows route aggregation, multicast employment, loose coupling between parties, message queuing and sometimes message ordering guarantees.

1.1 Security Concerns

On the other hand, the PubSub principle also poses a set of security problems which are often overlooked, like:

- Is the published information to be trusted as unmodified and authentic?
- Who is the publisher, and is it authorized to publish this information?
- Will the publications only reach authorized and authenticated subscribers?
- Who operates the message routers, and are the routers able to maintain privacy of subscriptions, and confidentiality and integrity of messages?

Protection of systems, including PubSub systems, is a multi-faceted problem. As Wang et al. [2] points out, it includes authentication, access control, information integrity, service integrity, confidentiality, accountability and availability. This paper focuses on how authentication and access control mechanisms provides information integrity and confidentiality, and how hardware-based units may support the need for service integrity.

The simple and widespread solution to authentication is to let every system keep its own registry of user names and passwords, an approach which is known to scale poorly and to create inconsistent and stale user information. A system based on centralized identity providers can offer a centralized registry and mutual authentication of client and service.

Access control in PubSub systems can be divided into those who enforce their policies either on connections, on subscriptions or on individual messages. The system being presented in this paper is the latter category, where each message being published is given an access policy which is checked for every subscriber before delivery.

Access control systems can be identity based, where privileges are assigned to individual users, or role based, where privileges are given to *roles* which again are assigned to users. Role Based Access Control [1] are known to scale better in terms of management resource requirements, but *role engineering* is necessary in order to avoid a combinatorial explosion of the role set. The use of RBAC access control in PubSub systems was investigated by Belokosztolszki et al. [3], but their work enforces the access policy only on subscription level. IBM's WebSphere MQ takes a similar approach, where access to "topics" (called *nodes*) are controlled by an ACL-based protection mechanism.

Our work extends, in some respects, the protection mechanism proposed by Fiege et al. [4], where the use of Attribute Certificates controls the reception of individual messages, although our work pays more attention to the problems related to identity management and how the access control mechanisms can mediate the message flows.

In the context of a PubSub distribution system, the use of Identity Management may increase the trust in the integrity and the confidentiality of the messages, and the control of who can post and subscribe to messages. This can be done in a scalable manner with loose coupling between the responsible parties:

- The integrity of messages is the concern of the subscriber, which should be able to express publication requirements and verify the correctness of the content.

- The confidentiality is the concern of the publisher, which should be able to specify the required authorizations for the subscriber to receive it.
- The authenticity of identities (routers and clients) is the responsibility of the Message Routers, which serves as the contact point to the transport infrastructure and requires that all entities authenticate themselves prior to operation.
- The integrity of identifiers, keys and attributes associated with an entity (subscriber or publisher) is the responsibility of the Authority, which operates an Identity Provider service for the purpose of issuing identity statements.

This "separation of concern" pattern reduces the necessary amount of coordination and control traffic, and offers a flexible arrangement for the expression and enforcement of the security requirements.

From an architectural perspective, access control adds an extra dimension to message routing in a PubSub system. There is a set of topics (hierarchical or orthogonal) associated with publications and subscriptions which mediate the message flow from publishers to subscribers. The additional requirement listed above can be distributed between routers so that messages do not travel further than necessary. There is no need to pass on a message if there are no authorized subscribers downstream (even if they subscribe to the given topics). Similarly, there is no need to pass on messages from publishers which are not approved by any downstream subscribers.

1.2 Integrity Protection of Routers

The descriptions in the previous sections indicate that the confidentiality of messages rely on the integrity of the message routers. If they fail to operate according to the subscription requirements in the messages, confidentiality may be compromised.

The approach taken in this paper is to employ hardware units called Trusted Platform Modules (TPM) to seal the configuration of the router nodes in order to detect malware etc. The TPM is non-bypassable and tamper-proof. The use of TPM will be discussed in Section 5.

1.3 Gismo IdM

The IdM prototype being presented in this paper was once developed with a service oriented, tactical environment in mind, with ample opportunities for cross domain operation and prudent networking protocols. In addition to the IdP services, it also offers a range of authentication protocols for use in different contexts. It is called *Gismo IdM* from a project called "General IT Security for Mobile Operations" at the Norwegian Defence Research Establishment.[5] Gismo IdM supports the PubSub environment excellently.

The contribution of this paper is a novel security model for PubSub service environment and its implementation, which partly relies on the services from an IdM, partly on a hardware based integrity protection. The integration of access control in the message routing decisions has not been observed in existing publications.

The remainder of the paper is organized as follows: Section 2 gives an architectural overview of the PubSub system which is being presented. In Section 3 the mechanisms

related to publication selection and forwarding are described. Section 4 gives a detailed presentation of the underlying identity management systems (Gismo IdM). Furthermore, a presentation of how the TPM hardware module can cooperate with the IdM in order to support service integrity is discussed in Section 5. The paper presents its conclusions and suggestions for further research in Section 6.

2 Outline of the PubSub Implementation

Figure 1 outlines the design of the proposed PubSub system. Observe the following properties:

- There are three types of actors: Publishers, Subscribers and Message Routers (simply called Routers for the remaining of the paper). A process can be both publisher and subscribers at the same time since they use the same interface.
- All actors belong to one *Community of Interest* (COI).
- The identities of the actors in a COI are governed by an *Identity Provider* (IdP) which issues *Identity Statements* upon request. The identity statements together with the (pre-loaded) private key enable the actor to authenticate itself when a connection is established.
- Messages (also called Publications) flow from one publisher to zero or more subscribers.

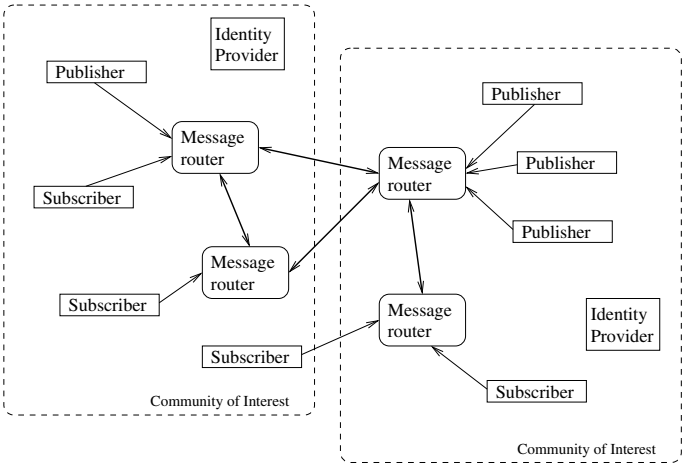


Fig. 1. PubSub system design outline

Figure 2 illustrates the representation of a subscription. It consists of a set of topic strings, which are to be interpreted in a hierarchical context: A subscription related to a node in the topic tree also includes all sub-topics under that node. The subscription also includes a logical expression that must evaluate to "true" with respect to the publisher's identity attributes: The subscription shown in the figure requires that the publisher either

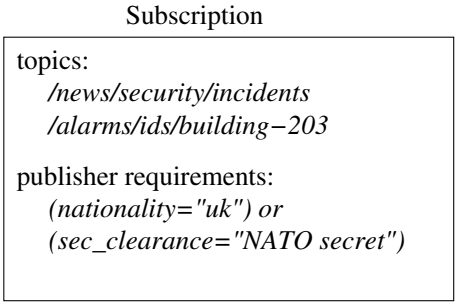


Fig. 2. Subscription representation

has the attribute `nationality=''uk''` or `sec_clearance=''NATO secret''`. Otherwise, the subscription will not apply to messages posted by that publisher.

Publications are, in the same manner, annotated with a set of topic strings to describe its categories, and a boolean expression which must evaluate to "true" with respect to the attributes of the subscriber, called the *subscriber requirement*. Its function is to protect the confidentiality of the message, and it is the responsibility of the message routers to ensure that the confidentiality is not broken.

2.1 Functional Requirements

As a basis for the design and analysis, the functional requirements to how an Identity Management system (IdM) will influence the properties of a PubSub system will now be described:

- The message flows from publishers to subscribers are mediated by topics of subscriptions and publications, as well as the identity attributes of the parties and the related attribute requirements. Messages will never be delivered to unauthorized subscribers or from unauthorized publishers.
- Client (publishers and subscribers) to router connections and router to router connections must be mutually authenticated.
- Cross domain authentication is possible where there are trust relations between the given IdPs.
- Publications are signed for integrity protection, and the identity statement of the publisher is included with the message.
- A subscription to one topic implies subscriptions to all sub-topics as well.

2.2 Non-functional Requirements

To improve the scalability, resilience and reliability of the PubSub system, a set of non-functional requirements must be met:

- A router decision to stop a message on its way to the subscriber should be made as close to the publisher as possible, regardless if it is a matter of topics or authorizations.

- There should be no revocation mechanism for the identity statements. ISes should be issued with so short lifetime that revocation becomes unnecessary. A connection is terminated if the ISes involved in authentication expires without being renewed.
- There should be no need for configurations of the message routers except for loading their key pairs and enter the IP addresses of their link neighbors. No registration of client identities or topics need to take place.
- In the case where topics are hierarchically organized, the syntax representation of a topic need to list all ancestor topics as well (e.g. a full "pathname" from the top of the topic tree).

3 Operational Details

The operation of the system will now be explained at the detail level of algorithms and protocol transactions. Some important data structures will also be described.

3.1 The Subscription Object

A subscriber expresses interest in posted publications based on topics and logical expressions of attribute values. Their syntax definition is as follows:

```
subscriptobj   = topics,expr
topics         = topic+
topic          = string
expr           = operand
                 | expr binary-operator expr
                 | unary-operator expr
operand        = literal | attribute
literal        = string | number
                 | true | false
attribute      = string
unaryoperator  = not | nameexists
binaryoperator = == | > | ismember | hastoken
                 | substring ...
```

The attribute values which enter into the logical expressions are taken from the identity statement of the publisher. This grammar is simplified for the purpose of discussion, e.g., logical operands cannot use the "$>$" operator etc.

3.2 Subscription Matching

Let P denote the set of all publications. The subscription a matches a set A of publications, $A \subseteq P$. A publication p is a member of A if the boolean function $match(a,p)$ returns *true*, formally expressed in the following manner:

$$A = \{p | match(a,p) = true\}, p \in P \tag{1}$$

The selection made by the *match()* function maintains the *integrity* aspect of the delivery mechanism: In addition to the topicality selection, the publisher must meet the requirements set by the subscriber. These requirements are evaluated on the attributes of the publisher's identity statement, which are assigned and sealed by the Authority and distributed by the Identity Provider (IdP).

Please observe that a subscriber will not receive a publication if it doesn't meet the requirements set by the publisher. These requirements are expressed as a boolean expression, included in the publication object and evaluated on the attributes found in the subscriber's identity statement.

3.3 Merging of Subscriptions

An important operation on subscription objects is the *merge* operation, during which a set of subscriptions are joined into one which represents the equivalent selection criteria. The operation is essential when a router wishes to announce the subscription of all its clients. The merging process takes two subscription instances (a,b) and produces an aggregate subscription c:

$$c = merge(a,b) \tag{2}$$

Where the set C matched by c relates to A and B so that

$$P \supseteq C \supseteq A \cup B \tag{3}$$

The `merge` function may create aggregate subscriptions which match superfluous publications. It is an assumption that the aggregate subscription can have a more compact representation if it is allowed a fraction of false positives, which we choose to call the *spill*. The spill set can be expressed by a set function in the following manner:

$$spill(a,b,c) = \{x | x \in C, x \notin (A \cup B)\} \tag{4}$$

which is simply the differential set $C \backslash (A \cup B)$. Since the spill represents a waste of communication bandwidth, the relation between the spill and the size of the aggregate subscription is interesting from an optimization perspective. It is reasonable to expect a size/complexity tradeoff of an aggregate subscription and the amount of spill, approximated by

$$|spill(a,b,c)| = \frac{k}{size(c)} \tag{5}$$

(k is simply a scaling factor). This tradeoff can be dynamically adjusted in the routers, since this is where the merging function is calculated, and this is where excessive publications are identified.

3.4 The Publication Object

A publication (sometimes simply called a message) is created by a publisher and sent to a router. The router will deliver it to locally connected subscribers and to neighboring routers in accordance with the service semantics and the existing subscriptions. The

different aspects of publication and subscription routing will be described later. In this section, only the structure of the publication object will be described.

A publication object represents a publication as it is passed through the system. The elements of the object enables the system to enforce the confidentiality requirements of the message, and the integrity requirements set by the subscriber. The elements of the publication object are:

1. The actual message content (any serializable object)
2. The topics of the message
3. The identity statement of the publisher
4. A boolean expression which expresses the requirements which the subscriber must meet
5. An unique value used for loop detection
6. The publisher's signature

Through the inclusion of the publisher's identity statement and a signature, the subscriber can verify that the publisher meets the attribute requirements, and that the message is unmodified. The publisher is not allowed to include the identity statement, this is done by the router which remembers the identity statement from the authentication phase.

3.5 Routing Tables

When a Message Router starts its operation, it will make connections to a number of other routers, based on its configuration. The population of routers thus forms a mesh networks, as shown in Figure 1, where there are indirect connections between some routers, and *routing information* becomes necessary.

The routes are constructed through the flooding of subscription objects, there are no separate routing messages in the system.

Every router will construct an aggregated subscription based on the subscriptions of all its connected clients, add a time stamp, and regularly pass it to all its router neighbors for the purpose of flooding. For the remainder of the paper, this structure is called the *AggSub* object.

Every router will keep a list of the AggSub objects it has received, and from which neighbor it came from. The time stamp of the object serves two purposes: To detect and discard duplicates during the flooding process, and to establish the best route based on the earliest AggSub object received.

A local timestamp is attached to every entry in this list, so that stale routes and disappeared routers will be removed from the list. The regular dissemination of AggSub objects serves as a leasing mechanism for liveness control.

The purpose of propagating not only the topics of interest, but also the attribute requirement is to discard unwanted/unauthorized publications as early as possible in its forwarding path. The advantages of doing this is twofold: (1) To avoid wasting bandwidth and (2) to reduce the exposition of messages to message routers that are possibly compromised.

3.6 Publication Routing

The distribution and use of AggSub-lists for routing information is simple and avoids separate mechanisms for route construction and subscription dissemination. It does not avoid route cycles, so a mechanism to detect and abort looping of publication messages must be in place.

When a router receives a publication, either from a connected publisher or from a neighbor router, it will consult the tables of connected subscribers and the AggSub list to determine who to forward it to. Locally connected clients are given the publications individually since they are separately connected.

Routers who are entitled to receive the publication are passed the publication through the neighbor for that route. Each neighbor are given at most one copy, and the neighbor from whom the publication was received is given none.

Loops are detected through the inclusion of a UUID in the publication object which are temporarily remembered in the router. Duplicates are silently discarded.

Client Subscription Table. This table is built dynamically based on the connected clients. Each row represents a client and the columns contain the following information:

- An object representing the TCP connection
- The identity statement of the client
- The subscription objects of the client

During connection, the clients authenticate themselves through a protocol where the client demonstrates the possession of its private key associated with its identity statement. The client table is a trusted source of information about the clients and their subject attributes, a trust which is essential to the message passing mechanisms.

4 Identity Management Details

The PubSub system being presented in this paper requires a trusted party which issues credentials and an infrastructure for the distribution of these credentials. The credentials are needed to authenticate a client or a router, and to attest the attributes associated with the subject for access control purposes. The message forwarding mechanisms described earlier in the paper is considered to be access control operations, since they support the requirements of message integrity and confidentiality.

Systems that support these operations are commonly known as *Identity Management* (IdM) systems. At one end of the IdM system range we find Single Sign On (SSO) systems with client-authentication only and strong coupling between the parties. At the other end of the range there are web-service oriented system which require validation of credentials in both ends of a connection, with the ability to operate in a cross domain environment with loose coupling between the management domains.

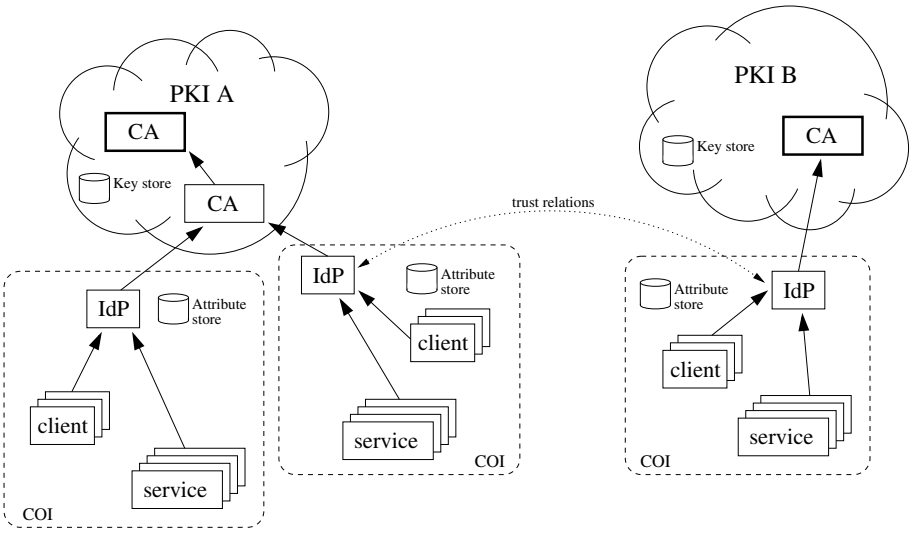


Fig. 3. The functional components of the Gismo IdM. Observe that the IdP serves one single COI, and the trust relations are formed between COIs, not domains. Key management is handled by the PKI whereas the attribute management is done by the IdPs on the COI level.

For the PubSub system being described in this paper, the existing *Gismo IdM* experimental IdM system was used as a trusted third party [5].

A architectural view of the Gismo IdM can be seen in Figure 3. Its notable properties are:

- Identity statements contain subject identifier, subject attributes, subject's public key, issuer identifier, issuer's public key and an expiration time.
- Identity statements are issued with a short lifetime. No revocation arrangement is therefore needed.
- Identity statements are issued by an *Identity Provider* to subjects that are properly registered in, e.g., a PKI. The subject's private key and the IdP's public key must be pre-installed in the subject's computer for the identity statement to be useful.
- *Guest identity statements* can be issued to subjects who can present an identity statement issued by an IdP to which there exists a *trust relationship*. Guest identity statements allow parties belonging to different management domains to authenticate themselves.

For the use of the Gismo IdM in a service oriented (SOA) environment in a military disadvantaged network some efficient authentication protocols were developed. They minimize the number of protocol round trips by piggybacking authentication data on the actual service request and service response, and they minimize the required state space in the server through nontraditional replay protection mechanisms.[6]

One of these authentication protocols has been used in the PubSub system as a means for mutual authentication of router-to-router and router-to-client connection. Consequently, all clients and routers need to be registered with an IdM instance and have their key pair installed prior to connecting to a router.

4.1 Cross Domain Connections

In those cases where the parties of a connection belong to different domains, i.e., they trust different IdPs, they cannot directly validate each other's identity statements. In order to authenticate across IdM domains, *guest identity statements* were introduced. Figure 4 shows how a client X_a from IdM domain a presents its identity statement, termed $(Id_x)_a$, to the IdP of domain b. On the condition that there exists a trust relationship between the domains a and b, IdP_b can issue a guest identity statement with the same content and its own signature. The guest IS is termed $(Id_x)_b$ and may be validated by any member of IdM domain b. For the opposite direction, authentication will require the identity statement IdP_a issued by IdP_b in order to construct a signature chain back to the trust anchor of a. This IS is termed $(Id_a)_b$ and is enclosed in the guest IS response.

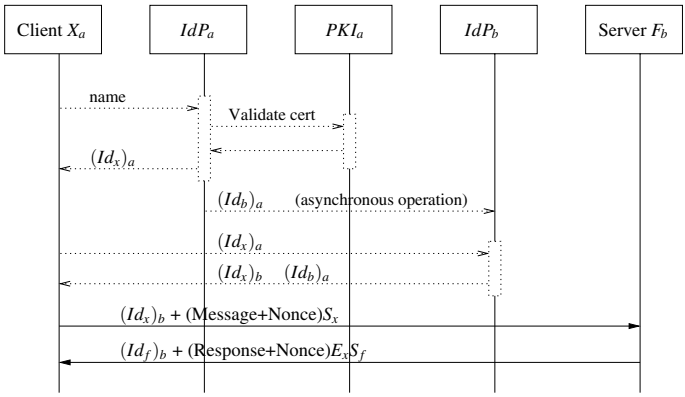


Fig. 4. The authentication protocol for the PubSub service. There is no replay attack protection since they are not considered as threats, but the response need to be protected for reasons of response replay and information compromise.

Figure 4 also shows the details of the authentication protocol. The design choice has been to sign the request for integrity protection, and to encrypt the response as a means of replay attack protection. The response also contains the necessary information to authenticate the responder.

This protocol was developed as a service invocation protocol in [6], and has been used as a connection protocol in the PubSub system. The symmetric key which is used in the encryption of the response (as a part of the asymmetric encryption operation) is subsequently used for the protection of the message traffic.

4.2 Protection of Message Integrity

In addition to the control of the delivery mechanisms, the IdM-supplied credentials also offer the protection of publication integrity. As pointed out in Section 3.4 the publication object includes the identity statement of the publisher and a signature which seals the content of the message.

The receiver can check the integrity of the message and the authenticity of the publisher by verifying the signature and validating the identity statement.

Validation requires trust in the IdP of the publisher, which exist if they belong to the same IdM domain or to domains with a trust relationship. If, however, they belong to domains between which there exist a *chain* of trust relations, then the validation cannot take place. Trust relations are per definition *lateral*, not transitive.

4.3 Renewal of Identity Statements

In a SOA environment where services are short lived, i.e., they finish very soon after being invoked, it suffices to check authenticity and authorizations during invocations. In a long lived service environment, where the services are active for hours after invocation, the credentials should be checked for expiration during this period.

The PubSub system is a long lived service, where the connection to a router may be held active for hours and days. Since the credentials used during connection establishment have an expiration time shorter than that (typically a small number of hours), their validity should be monitored so *no connection can be active if the credentials are expired or invalidated*.

For the connection to be maintained across credential expiration periods, the identity statement must be *renewed* during a connection period. The owner of an identity statement must contact the IdPs before expiration to get a new issue and send it to all its connected neighbors (clients and routers). Re-authentication is not necessary.

The recipient of the identity statement should validate it and check that the subject attributes meet the connection requirements. It should also replace the old set of attributes so that new subscriptions and publications reflect this set.

Existing subscriptions may be incorrect with respect to the new attribute set. Subject attributes are not included in the subscriptions when propagated through the network (only topic information and publisher requirements are). Therefore, the flow of publications will not need to be altered due to the new identity statement. The new set of subject attributes will always be present in the router which connects the client and make sure that the publications are delivered to clients based on their most recent attribute set.

5 Trusted Binding of Message Router Operation

The confidentiality of publications is the responsibility of the message routers, which means that they must be trusted to operate correctly and not leak information to unauthorized subscribers.

Even if a message router has correctly authenticated itself to its neighbors, one cannot exclude the possibility for a message router to be infected by malware, operate incorrectly or to have been compromised in some other way. To some extent, it might possible to strengthen the trust between nodes by resorting to a *trusted binding* between a node identity and its own hardware and software configuration.

The concept of trusted binding has been introduced by Hegland et al. [7] as a way to reduce the burden of authentication in a network with low resources, and is tightly

connected to the idea of transitive trust. If an entity *A* has authenticated and trusts an entity *B*, and *B* has a trusted binding with *C*, then *A* will also trust *C* without further need for authentication. In a way, this is what already happens in our framework: a message router *A* trusts another message router *C* because it presents a proof of authentication (the identity statement) issued by an IdP_b, which *A* trusts. However, we want to extend this concept to be able to trust, besides the identity of a message router, also its *integrity*. We accomplish this by integrating a TPM based attestation component in our identity management framework.

A TPM (Trusted Platform Module) [8], is a cryptoprocessor that enables a machine to perform, among other things, a *trusted boot* and thereafter to report in a trusted way (*attest*) its running software and hardware configuration to a third party. The trusted boot measures all software components being run on the machine (by computing the hash of their binaries) from the boot moment until the OS takes control, and stores these measurements in the TPM registers (PCRs) in a tamperproof manner. The measurements can then be signed with the TPM private key (which never leaves the TPM), and sent to a third party which can verify them against a database of approved configurations. The procedure is illustrated in Figure 5. This can allow an IdP to verify not only the identity of a message router, but also that it was booted with genuine and updated software. This extra information can then be added to the identity statement as a new subject attribute: {`TMPintegrity=AssuranceLevel2`}. Neighboring nodes may require this assurance in order to accept the nodes' participation in the router network and, possibly, to create clusters of routers with different assurance levels.

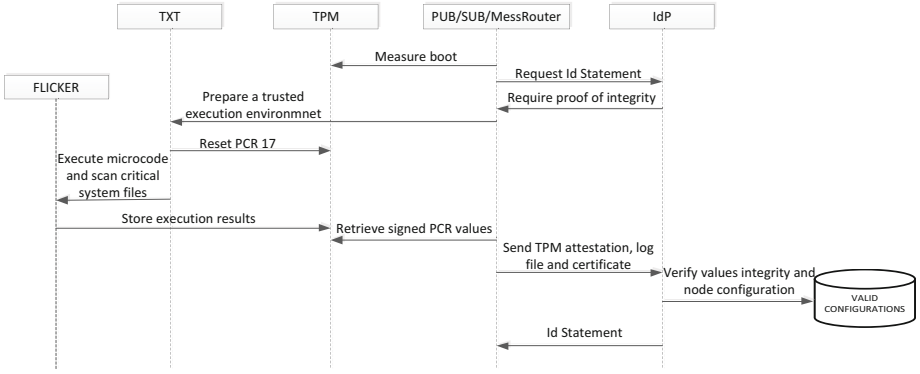


Fig. 5. This diagram shows an example of attestation process (at very high level) that leverages on both the TPM, a classical *trusted boot* process, and a TXT enabled Intel processor which can be used to execute Flicker microapplications in a trusted setting. In this case the microapplication is a virus/malware scanner that is run securely before sending the node integrity attestation to the Identity Provider. The result of the scan is also sent as part of the TPM attestation.

Similar work has been done with IdM systems based on Kerberos [9] and OpenID [10], although the main goal is to improve the user experience by eliminating the need for passwords.

One limitation of the trusted boot approach is that the trust in the system is built by the boot process and passed transitively to the OS, which takes over the system once the boot is finished. This means that after this point we should trust the OS to measure correctly itself and the processes it is running in order to perform a trusted attestation. Unfortunately, OSes are often quite large and complex and are likely to contain vulnerabilities that can be exploited by an attacker. Obviously, an infected machine cannot be trusted to report correct information about itself to a third party. Besides, if a system is compromised, the machine must be rebooted again in order to establish a new chain of trust. This is why the trusted boot is said to be based on a *static root of trust*.

Although continuous monitoring and measuring of all the system in search for infections or malfunctions is unrealistic, research is ongoing to define new techniques that can help attesting the status of a running machine. Property based attestation is one example [11]. A better alternative, however, might consist in using a *dynamic root of trust*. The recent Intel and AMD processors offer a special instruction that allows the system to create a secure and trusted hardware environment for the execution of some given code, even if the rest of running system is compromised [12]. Therefore, trust in the system can be established *dynamically* at any time, independent of the boot process. The TPM is also involved in this process so that a node can attest that a certain piece of code was executed securely.

In particular, a framework called Flicker [13] allows to write micro applications that can be run safely at any time even the current OS is compromised. A report about the code execution can also be generated and certified with the help of the TPM in a fashion similar to the attestation process.

Associated with the PubSub research, a study is ongoing to understand how a dynamic root of trust based application like Flicker can be integrated with our framework. Our aim is to design an identity management system where authentication and valid identity statements can be issued also if some nodes in the network are compromised, and would normally not be trusted. The remaining probability for incorrect operation may be subject to calculation on forwarding path length: The higher number of routers used in a forwarding path for information, the higher is the aggregated probability for incorrect operation. These calculations can also consider the strength of crypto algorithms in use, the number of COI domains involved etc.

6 Conclusion and Future Research

This paper has described a novel model for protecting integrity and confidentiality of messages in a PubSub system. Pivotal to the model are the services of an IdM, which supports the authentication and access control operations, and the integrity protection offered by a non-bypassable hardware unit called TPM.

A proof of concept prototype has been written in Java to study the functional aspects of the proposed PubSub principles. A deeper evalution in the context of a military information system is planned in the near future. Also, the protection mechanisms that rely on a TPM hardware unit is being developed for the time being as an ongoing research project.

References

1. Sandhu, R., Ferraiolo, D., Kuhn, R.: The NIST model for role-based access control: towards a unified standard. In: RBAC 2000: Proceedings of the Fifth ACM Workshop on Role-Based Access Control, pp. 47–63. ACM, New York (2000)
2. Wang, C., Carzaniga, A., Evans, D., Wolf, A.L., Wolf, E.L.: Security issues and requirements for internet-scale publish-subscribe systems. In: 35th Hawaii International Conference on System Sciences (HICSS-35), Big Island (2002)
3. Belokosztolszki, A., Eyers, D.M., Pietzuch, P.R., Bacon, J., Moody, K.: Role-based access control for publish/subscribe middleware architectures. In: Proceedings of the 2nd International Workshop on Distributed Event-Based Systems, DEBS 2003, pp. 1–8. ACM, New York (2003)
4. Fiege, L., Zeidler, A., Buchmann, A., Kilian-Kehr, R., Mühl, G., Darmstadt, T.: Security aspects in publish/subscribe systems. In: Third Intl. Workshop on Distributed Event-based Systems, DEBS 2004. IEEE (2004)
5. Fongen, A.: Architecture patterns for a ubiquitous identity management system. In: ICONS 2011, Saint Maartens, IARIA (January 2011)
6. Fongen, A.: Identity management without revocation. In: SECURWARE 2010, Mestre, Italy, IARIA (July 2010)
7. Hegland, A.M., Winjum, E., Hedenstad, O.E.: A framework for authentication in nbd tactical ad hoc networks. IEEE Communications Magazine 49(10), 64–71 (2011)
8. Trusted Computing Group: TPM Main Specification, `http://www.trustedcomputinggroup.org/resources/tpm_main_specification` (Online, Accessed March 2012)
9. Leicher, A., Kuntze, N., Schmidt, A.U.: Implementation of a Trusted Ticket System. In: Gritzalis, D., Lopez, J. (eds.) SEC 2009. IFIP AICT, vol. 297, pp. 152–163. Springer, Heidelberg (2009)
10. Leicher, A., Schmidt, A., Shah, Y., Cha, I.: Trusted Computing enhanced OpenID. In: 2010 International Conference for Internet Technology and Secured Transactions (ICITST), pp. 1–8 (November 2010)
11. Nagarajan, A., Varadharajan, V., Hitchens, M., Gallery, E.: Property based attestation and trusted computing: Analysis and challenges. In: Proceedings of the 2009 Third International Conference on Network and System Security, NSS 2009, pp. 278–285. IEEE Computer Society, Washington, DC (2009)
12. Grawrock, D.: The Intel Safer Computing Initiative: Building Blocks for Trusted Computing. Engineer to Engineer Series. Intel Press (2006)
13. McCun, J.M.: Reducing the Trusted Computing Base for Applications on Commodity Systems. PhD thesis, School of Electrical and Computer Engineering, Carnegie Mellon University (2009)

Extended HTTP Digest Access Authentication

Henning Klevjer, Kent Are Varmedal, and Audun Jøsang

Department of Informatics, University of Oslo
{hennikl,kentav,josang}@ifi.uio.no

Abstract. User authentication to a server is typically done by presenting a username and a password in some protected form to the server, and having the server verify that those credentials correspond to an identity previously registered and authorized for access. It is crucial that attackers never get access to operational passwords, which typically is achieved by encryption in transit, or through a challenge-response protocol between the client and server computer platforms. However, these mechanisms do not protect passwords at the moment when they are entered into the client computer, which leaves the password exposed to attacks by malware on the client. We present a method for protecting passwords from being exposed on client platforms. The method is an extension of the well-known HTTP Digest Access Authentication which is a challenge-response protocol specified as part of HTTP. The method relies on an external *mostly offline* personal authentication device called OffPAD which communicates with the client platform. We show how the presented authentication scheme increases security as well as enhances usability with regard to identity management. In addition to describing the OffPAD device, we argue that the HTTP Digest Access Authentication standard does not conform to today's best practices, and suggest improvements.

1 Introduction

Passwords have for some time been considered old fashioned, difficult to use and a low quality authentication factor [1]. Poor usability has been the main focus of the critics: Requiring the user to select *secure* passwords for every online service is a major usability issue [20]. Another significant problem is the security state of the systems on which passwords are entered. According to PandaLabs' estimates, about a third (31.63 %) of the world's PCs are infected with some sort of malware (Q2 2012) of which most (78.92 %) are Trojans [17].

In light of these arguments it is reasonable to assume that passwords are vulnerable on the PC; both when they are entered and when they are stored. Passwords can be intercepted by Trojans either by keystroke logging, RAM-scraping, or by screenshots when shown on the screen in clear text. Even automatic log-in and identity management applications such as LastPass[1] are not safe, as they release the clear text password to the web browser (or other application) during

[1] http://lastpass.com

S. Fischer-Hübner et al.(Eds.): IDMAN 2013, IFIP AICT 396, pp. 83–96, 2013.

authentication, leaving it visible in memory for the Trojan to steal. LastPass is a popular relief among technically literate people who typically have many passwords to manage, another issue we address in this paper.

Over time, the typical Internet surfer will accumulate a large number of online identities. Each identity normally consists of a username or other unique identifier, and a password that (ideally) is unique and hard to guess. The increasing number of accumulated identities leads to *identity overload*, which means that the user is unable to manage all her identities (i.e. remembering all the different identities and corresponding passwords), at least not without compromising security in some way.

From the service provider's (SP) perspective identity management is relatively simple; it consists of storing identities and credentials of all its users in a single directory, which is typically part of a CRM (Customer Relationship Management) system. This *silo model* - where each SP controls the identities of its own customers - is simple to set up and manage, thus widely adopted in the industry. A downside of this model is that it quickly leads to identity overload for users [9], because each new online service gives another identity for the user to manage. Unfortunately there is currently no widely accepted local user-side solution for identity management that at the same time is secure and simple to use. As a result, users tend to cope with identity overload by reusing the same password for many different services, or by using insecure methods for storing passwords, which violate policies and best practice. It is shown that most users reuse difficult passwords for accounts protecting high value data, and use easily guessed passwords for low value data [2].

Some countermeasures have been introduced, such as federated identity management (FIM). FIM relies on mutual trust within a group of SPs, and optionally on a centralised identity provider. Through federation the member-SPs are able to share identities between their respective silo domains. By this scheme, a user can assume one single identity for an arbitrary selection of services, as long as they are in the same federated trust network. Federation between heterogeneous service providers in particular has never really taken off[2], probably due to the trust issues that arise when an identity is shared between SPs.

While FIM to some degree reduces the identity overload problem, there is no reason to believe that there will ever be one single identity federation, universally deployed, covering the needs of all different kinds of identity providers. Identity federations are faced with the risk of being a single point of failure and compromise for all services covered by a federation domain. If a user compromises his federated password or an attacker gains access to his identity management account, his federated identity is compromised across all federated domains.

In this paper we describe a method for local user-side identity management based on an authentication device called *OffPAD*, combined with an extension of

[2] One can argue that identity providers such as Google, Facebook (Connect) and OpenID have had a huge impact on identity federation; however, the services covered by these are rather similar (blogs, message boards, etc.), and not reaching across heterogeneous domains or domains requiring high-level assurance of authenticity.

the well-known HTTP Digest Access Authentication protocol. A brief overview of the existing HTTP Digest Access Authentication standard is provided next. We then describe our method of combining the OffPAD with extended HTTP Digest Access Authentication. The advantage of our method is that it totally prevents exposure of passwords on potentially vulnerable client platforms, and thereby represents secure local user-centric identity management solution.

1.1 HTTP Digest Access Authentication

HTTP Digest Access Authentication (short: DAA) originates from the challenge-response authentication framework described in the original HTTP 1.0 specification [3]. It is a web standard for access control to a service or domain called *realm* by user authentication over HTTP. DAA was first defined in 1997 in RFC 2069 [5] and refurbished in RFC 2617 [6] in 1999. Its intended use is on the World Wide Web, but it is perfectly implementable for protection of local resources, or in any situation where application level access control is required[3].

DAA was introduced as an extension to its predecessor *Basic Access Authentication*, which is insecure without traffic encryption [6, 8]. The most critical weakness of Basic Access Authentication is that passwords are passed in clear text (Base64 encoded) over the Internet. DAA does not transmit passwords in clear, but instead uses a challenge-response protocol. Using DAA to access a protected realm requires each user to be:

- registered with sufficient credentials (username and password) in the access control list (ACL) of the system enforcing the realm's access control (i.e. be authorized for access to that realm), and
- able to produce those registered credentials during authentication to the server.

To understand DAA, consider this scenario: A user wants to access some web resource at `http://example.com/protected/`. The `/protected/` directory (or realm) is protected with DAA, so that only authorized users shall be able to access it and its subdirectories.

Trying to access `http://example.com/protected/` initiates the following challenge-response communication between the client and the server, over the HTTP protocol:

1. The client's web browser (*user agent*) issues a HTTP `GET` to retrieve `http://example.com/protected/`
2. The server responds with a `401 Authorization Required` HTTP status code, indicating to the user that access to this resource is protected, requires access approval by the system[4] and that he is currently not authenticated.

[3] The challenge-response theory behind the scheme is applicable also outside HTTP.

[4] In the RFC specification, this stage alerts the need for what the `401` header refers to as *authorization*, but this is a misnomer. What the 401 header actually says is that the user must provide authentication credentials, so that the system can *verify that the user is registered and authorized for access.* The system can then approve or reject access to the resource, depending on the stored access authorization policy.

Along with the status code, the server passes a `WWW-authenticate` header, containing information needed for the system to calculate the correct response for the server.
3. The web browser interprets the `401` status code and prompts the user for username and password.
4. The entered credentials and the information extracted from the incoming `WWW-authenticate` header are hashed. The client issues another `GET`, now with an appended `Authorization` header, containing a response value (i.e. the previous hash, the user's proof of identity) and other values.
5. The server receives the response value and the name of the *protection realm* to which he requests access. As hashing algorithms are one-way functions, there is no way for an adversary to simply extract the password from the hash value. The protection of the response value relies on the quality (*entropy*) of the password, or at least on the preimage resistance of the hash function. At the server side, the credentials that were stored locally at the time of registration are used to calculate another hash value by the same rules and algorithm as on the client side. If the server side calculation is equivalent to the one received from the client, the server can be certain of the client's identity, and approve access based on the access policy. If the user is authenticated and authorized, the server responds with a `200 OK` status code and the contents of the protected resource that the user requested. If either authentication or access control fails (i.e. the user is not authenticated or not authorized), he is presented another `401 Authorization Required` and given another try at proving an authorized identity.

Calculating the Response Value. The `response` value is calculated by the user agent as an answer to the server's authentication challenge (the `WWW-Authenticate` header). It is the result of hashing two independent parts. The first, called `HA1` is a hash of the realm and the user's credentials. The second, `HA2`, is a hash of the HTTP request method and location. Consequently, one can distinguish `HA1` and `HA2` as the secret and non-secret pair, or static and dynamic components of `response` respectively. The static component is the one stored in the ACL at the server side and is used in calculations to produce the correct response value on either side for comparison and validation at the server. The dynamic component changes on every HTTP `GET`.

Here, we assume an ordinary run of DAA[5]:

HA1 = MD5(username:realm:password) [6, p. 12]
HA2 = MD5(method:uri)[6] [6, p. 13]

When both `HA1` and `HA2` have been calculated, the response value is finalized

response = MD5(HA1:nonce:nc:cnonce:qop:HA2)[7] [6, p. 13]

[5] We do not consider the `algorithm` or `qop` fields' impact on the calculation.
[6] Where `method` is the HTTP request method that was used (i.e. any of the methods described in the HTTP standards, such as GET, POST, etc. [4]).
[7] Details on the contents of `response` are omitted in this paper, for brevity. Refer to [6] for details on HTTP DAA.

1.2 Extended HTTP Digest Access Authentication

Extended Digest Access Authentication (short: XDAA) represents an extension of traditional HTTP DAA in two respects: The actual IETF standard RFC 2617 is extended to allow more than just username and password as valid credential sets[8]. The authentication process itself is also extended, physically, in that it is moved to another location. All client-side calculations done in the authentication phase are outsourced to the OffPAD. The OffPAD will be discussed in length in section 3.

Our XDAA is beneficial both for security and usability. By managing the user credentials on an external device, we get a local user-centric identity management system, and no longer require users to remember their passwords. Moving the challenge-response calculations and handling of the values critical to authentication over to a mostly offline device, we reduce the risk of exposing these values. Moving the identity management over to such a device alleviates the cognitive and physical strain on the user during authentication, as well as removing the time penalty brought by user interaction in most situations[9].

In its simplest form (using the OffPAD and no further protection mechanisms), XDAA can be used with any HTTP server supporting original HTTP DAA without change to the server system. The immediate benefit is that the user's credentials themselves are never present. They are never shown on the screen, never exposed in any vulnerable state in the computer's memory and never transferred in clear text.

2 The Problem of Password Exposure

Since identity management on a post-it note, under the keyboard or in the user's brain is not particularly secure or user friendly, a better solution may be to store identities in the computer. Various software password managers exist, both online and offline. Web browsers' password managers is one example of a software password manager, where users store their credentials in the browser and have them automatically entered upon request. In Mozilla Firefox, managed identities are stored locally in encrypted format using a key that is stored alongside. The stored identities (consequently the passwords) are easily decrypted, if not protected by a *master password* [16]. Thus, any Firefox identity store not protected by a master password can be collected by a Trojan or other malware, and the passwords can be decrypted at another location. However, the quality of protection provided by a master password is as usual dependent on the quality (*entropy*) of that master password. Mounting brute force or other guessing attacks is trivial.

In addition, when a specific password has been decrypted it is exposed in clear text in the memory of the client platform and can be intercepted by e.g. by a

[8] This is particularly important for the topics raised in section 7.

[9] Situations where no identity or multiple identities are available for the user to authenticate with, the password is wrong, or another error appears, user interaction is necessary.

Trojan or other malicious parties with access to the client system or its memory. In order to protect passwords from exposure, they must be stored in an offline device that communicates with the client platform. More specifically, this device is an Offline Personal Authentication Device (OffPAD) described in detail next.

3 The OffPAD

As noted, traditional identity management on the user side consists of remembering, and in most cases either writing down or reusing passwords. Storing identities on a secured external device is a possible solution – analogous to writing down passwords and keeping them in a safe. In [10] Jøsang and Pope describe the Personal Authentication Device, a secure device external to the computer. The PAD is used as an identity management system to which the user authenticates once (with a PIN number, password or similar), and for one *session*[10], the user can authenticate to every supported service automatically using the PAD as his authenticator. It allows for authentication of the user, and facilitates user-centric identity management (i.e. a user's management of his own passwords) to happen on this device, rather than in the user's brain.

In [9], Jøsang describes a more secure PAD, the physically decoupled OffPAD. The OffPAD is a PAD that is restricted with regard to connectivity (as *offline* as possible), meaning that it should only be able to communicate by authorized request. This decoupling from networks improves security on the device, as it is less vulnerable to outside attacks.

3.1 Requirements for an OffPAD

We require the following of the OffPAD:

1. Limited connectivity – We suggest Near Field Communication (NFC) or other physically activated communication (so-called *contactless* communication). Caveat: While other (live) means of communications may seem appropriate, depending on the required assurance level, but will demote the device to a PAD.
2. Secure element – An infrastructure for secure messaging and storage such as described in ISO 7816-4[11].
3. Access control on the device – Requiring the holder of the device to authenticate via passphrase, biometry, etc. restricts unauthorized access to the device.

3.2 Using a Mobile Phone as the OffPAD

The current trend of mobile phone malware strongly indicates that the mobile phone is joining the computer as a vulnerable platform. *"In 2011, the Juniper*

[10] Limited either in time or number of connections.

[11] `http://www.iso.org/iso/iso_catalogue_catalogue_tc/catalogue_detail.htm?csnumber=36134`

MTC identified a 155 percent increase in mobile malware across all platforms, as compared to the previous year" [11]. The number of features in mobile phones, especially connectivity features, increase the number of attack vectors, thus the overall vulnerability of the device.

As a counterexample, the French company TazTag which specializes in secure contactless devices are developing a mobile phone (TPH-ONE)[12] which is said to be able to separate the secure element from the phone's operating system (Android), in having a *secure state*, that can be toggled on or off by the user when required. The *secure state* is a security context in which the phone works with the secure element only. The secure element is capable of handling encryption, and hashing of the credentials used for authentication. In the phone scenario, the phone is an OffPAD whenever it is in the secure state.

4 Related Work

Several authentication solutions (particularly unimplemented designs and recommendation) relying on an external device are present in the literature. Examples include the Pico by Stajano [21], MP-Auth by Mannan and Oorschot [14] and Nebuchadnezzar by Singer and Laurie [13]. Below we will briefly introduce each and show the OffPAD is different.

Pico. Pico is a device that authenticates a user through a challenge-response protocol. It stores private keys for communication with every application it supports authenticating to, in its on-board encrypted memory. Each supported application has one asymmetric key pair to communicate with Picos. Stajano explains authentication with the Pico in the following:

> The Pico challenges the app to prove ownership of the app's private key. Once the app does, the Pico sends its long-term public key for that pairing, thus identifying itself to the app, and then, as challenged by the app, proves ownership of the corresponding private key [...], thus authenticating itself to the app [21].

Challenges are presented as 2D visual codes (e.g. QR codes) to the Pico, and collected by the device's embedded camera. Transmission of the response is done over Bluetooth. The Pico solution requires changes to both the client and the server side. Most SPs are probably reluctant to consider changing their visual appearance to support another authentication scheme. Where the Pico is restricted to its own authentication scheme, the OffPAD authentication is done building on a pre-existing technology. Also, the Pico targets authentication to any device, both on- and offline.

[12] `http://taztag.com/index.php?option=com_content&view=article&id=104`

MP-Auth. In 2010, Mannan and Oorschot suggested the MP-Auth (or Mobile Password Authentication) protocol as a means for moving password authentication (not the passwords themselves) to a personal device, protecting them against being collected by malware. In this protocol, an SSL tunnel is established between the user's mobile phone and the server to which he will authenticate. The user's password or credential is then entered on the phone and transmitted, protected by the SSL tunnel, to the server, authenticating the user [14].

MP-Auth's solution relays the communication and entering of a password to a mobile phone, but does not provide the benefit of identity management.

Nebuchadnezzar. Nebuchadnezzar, or *the Neb*, is a 2008 idea by Singer and Laurie. They argue that attempting to establish a trusted path of communication between a "general purpose" operating system and a server is a bad idea [13]. They also present another unreasonable extreme: trying to do every operation on a minimal, secured, locked down operating system, and argue that the only sensible solution is a combination of the two. The position paper further describes the schematics behind a trusted device, the Nebuchadnezzar, which much like the OffPAD is an external device, maximally reduced with regard to features. The OffPAD may be seen as a physical implementation of the Nebuchadnezzar for user authentication over HTTP.

5 A Weakness in the Original HTTP Digest Access Authentication

Here we present the most important weakness of the original DAA scheme and how it can be exploited. In section 6 we look at what protection mechanisms can be used to avoid it. The original specification of HTTP DAA [6] warns of several weaknesses and vulnerabilities, such as:

1. RFC2617 is backwards compatible with its less secure first specification RFC2069;
2. A server challenge may be intercepted and modified to a Basic authentication challenge by a Man In The Middle;
3. Mounting an attack leveraged by an intercepted authentication response value;
4. Mounting an offline attack on the stored password hash.

The first two weaknesses rely on the ability to force the client into using another, more vulnerable authentication scheme. We assume that the OffPAD system can be configured to require authentication to happen without the ability to downgrade. The third vulnerability is shown in the following attack, and the theory can be applied to exploit the fourth.

This situation is analogous to the classic problem of *cracking* a hashed and salted password: In the `HA1` calculation, the static values (username, realm and colons) are analogue to *salt*. In the `response` calculation (section 1.1) we consider

Table 1. An exhaustive search for `response`

A1	HA1	response
user0123:protected:a	798C3C ... 774307	985F960CCA0C4CCBE854EE4D3D260CBE
user0123:protected:b	138821 ... 68B4AA	2F3B4280E8EFFB16BBF27AB827D024FF
user0123:protected:c	9B9D7A ... 8DCD5B	E83AEA5BE94BBEA91263A4CDD4FC9A24
...		
user0123:protected:passworc	2CEE85 ... 9CE7E6	123CC39EA2290D01556505C5BCD4BBDA
user0123:protected:password	3FE8DB ... 0FFD38	64B3C3C5091EE8FC16BB22D0FD838389

the `HA1` value and the static values (nonce, nc, cnonce, qop and HA2 separated by colons) analogue to *password* and *salt* respectively.

An `Authorization` header's `response` value is an expression on the form:

$$HA1 = MD5(s_1||password)$$
$$response = MD5(HA1||s_2)$$

Where || denotes string concatenation, and s_1 and s_2 are the static values, of the format "username:realm:" and "nonce:nc:cnonce:qop:HA2" respectively.

Attempting to break the one-way property of MD5 is not practical at the time of writing; the most effective known preimage attack has a computational complexity of $2^{123.4}$ [19]. To find a usable password, however, we must find a preimage of the `HA1` value, which itself is hashed into `response`. A successful brute force attack on the password will reveal the static secret *HA1* value, which in turn can be validated by the *response* calculation above.

Consider a scenario where an attacker has successfully collected an `Authorization` header. He is then prepared with all the values needed to calculate the `HA1` except the password. Actually, all values making up the entire final `response` are accessible should we find the correct password. Exhaustively searching the available preimages' character space is a usual approach to password cracking on hashed passwords. In this scenario we must customize the password cracking algorithm to first hash the guessed password together with the rest of the `A1`[13] parameters to recreate a suggestion for `HA1`. Second, we must use that `HA1` value in the `response` calculation (using the retrieved nonces and other collected parameters) to produce a possible response value. In the event that the `response` value equals the one collected, we have found a password that would have been usable in the same session with the same system. In table 1, we show how this approach is possible, using these example values.

```
username: user0123      realm:    protected     password: password
nc:       00000001      uri:      /protected/   method:   GET
nonce:    aGVsbG8=feffda052ab1e0707b0d1edeff74eab1eb7cafe4
cnonce:   VGhpcyBpcyBhIG5vbmNlLCBub3RoaW5nIGZhbmN5DQo=

correct HA1:          3FE8DB7B9A01F9715BB4285D300FFD38
correct HA2:          6AA3FBF46FDDCDE617B741460F5411B8
correct response:     64B3C3C5091EE8FC16BB22D0FD838389
```

[13] Note that `x1` is the original preimage of `Hx1` (i.e. the contents of the value before hashing).

If we can validate the found password against several nonces, we can conclude with high certainty that it is indeed the original preimage, and we have decoupled the password from the nonce- and client nonces.

6 Extended HTTP Digest Access Authentication

In this section we discuss the weaknesses noted above and how introducing the new scheme and the OffPAD helps mitigate these. We also discuss what new protection mechanisms should be appended to the scheme, and what more should be done with the mechanisms that did not stand the test of time. Introducing the OffPAD will remove the risk of clear text password exposure on the client computer but will not contribute in any way to the security of the authentication data while in transfer, or to the security of the server-side identity management. A well-known fact is that it is the server most attackers target when looking for user's credentials. Therefore, a number of additional protection mechanisms are introduced, to ascertain the security on the server side as well.

The changes done to the scheme will be transparent on the communication links, but modifications on both the client and the server side are necessary to provide maximal assurance. Introducing the OffPAD as the only additional protection mechanism will integrate seamlessly with any server supporting the original authentication scheme. However, while in line with the challenge-response authentication framework provided with HTTP[14], the protection mechanisms beyond the OffPAD only, require some browser and server-side modification. This is to synchronize the higher quality of protection of the user credentials on both ends.

6.1 Extending DAA to the OffPAD

By relocating the computation of the DAA response value from the client computer to the OffPAD the user can be authenticated without entering a password on the client computer. It also provides us with the ability to authenticate automatically, with the credentials stored on the device. Rather than storing credentials in clear, they can be stored as hashed static values `HA1`, containing "proof of possession" of the credentials. The password is never needed in clear text. We now remove the risk of malware collecting the password from the computer, as it is never entered or shown on the screen, thus never present in memory. Its only appearance in the computer's memory is when the hashed `response` value is passed between the OffPAD and the server, via the computer.

6.2 Weaknesses Not Addressed by the OffPAD

The only mechanism protecting the password is the hash function, which relies on the randomness and on the qualities of the password itself. As presented in section 5, it is feasible for to exhaustively search the character space of a "low quality" password and find a match for its `response` value.

[14] From which HTTP DAA is formed.

Ordinary hash functions have many applications. Many problems are solved from the hash function's possibility to quickly convert a large amount of data into a fixed size value, uniquely identifying the data. Version control systems, digital signatures and data comparison are among the applications that benefit from the speed of these highly efficient functions. When hashing a password for use in user authentication, the MD5 calculation itself is done in an incredibly short amount of time[15]. If the intention is turned around, however, it is easy to see how fast a brute force attack may be carried out.

In December 2012, Jeremi Gosney reported brute force attacks using the MD5 function measuring up to 180 billion calculations per second. The attacks were carried out on five clustered servers, connecting 25 GPUs[16] in total [7]. This means that the character space containing all 95 printable ASCII characters of variable length up to eight characters (i.e. $\frac{95^9-1}{94} - 1$), or about 6.7 *quadrillion* character combinations, can be calculated[17].

$$\frac{6704780954517120}{180000000000} = 37248,78 \text{ seconds } \approxeq 10,3 \text{ hours.} \tag{1}$$

Because of the effectiveness seen in brute force and dictionary attacks against hash- and encryption functions, the need to slow them down was introduced already in the early UNIX time sharing systems [15]. Several thousand iterations of MD5 can slow down the calculation enough to mitigate most brute force attacks. The Password Based Key Derivation Functions (PBKDF1 and PBKDF2) [12] were introduced by RSA in 2000, but do not explicitly mention user authentication. bcrypt by Provos and Mazières (1999), and the more current scrypt by Colin Percival (2012) [18], however, specifically have user authentication and password protection in mind. Common to all key derivation functions is that they use slow consuming calculations, thus provide a stronger protection to the value they protect. The PBKDF functions are used mainly to facilitate password based encryption by generating a key from a password, but every one-way key derivation function may be used for user authentication. It is up to the identity provider to determine the workload of each hash calculation. Despite the age of the Key Derivation Functions and the technology, their use for password protection on the server side has become slightly popular only recently. Slowing the hash calculation down, each user may have to wait a few hundred milliseconds to be authenticated, but this also applies for each single attack. One must use the same amount of time for each guessed password[18].

[15] Albeit a suggestion in the RFC, MD5 has become the de facto standard.

[16] Graphics processing unit – a special computer processor tailored for graphics, which has also proven effective in password cracking.

[17] Here we assume that there is no password of length zero (we subtract $95^0 = 1$ from the original *geometric sum formula*)

[18] Of course; the upper threshold for calculation time is only limited by the local system's resources. This means that an attacker's system, when superior to the verification server in computing power, will be able to guess faster than the remote verification time.

Servers protected by the original DAA scheme advertise their supported one-way function algorithms to the client following the challenge. This enables servers to provide support beyond the two algorithms specified in the standard (MD5 and MD5-sess). Advertising a KDF at the server side will benefit both the server and the client with additional protection of the password at both sides. If the algorithm used is implemented consistently (i.e. calculates the exact same values) at both endpoints, any one-way function or KDF should be transparently applicable to the authentication scheme.

Using KDFs not only protects the user credentials in transit, it provides the same benefits to either communicating entity storing the credentials locally (i.e. the server and OffPAD). If a password database is breached, and the stored passwords are hashed by single MD5, and even salted, most passwords are recoverable in a reasonable amount of time. If the passwords are protected by a KDF and a reasonable workload is applied, brute force attacks are not feasible. If a KDF uses 100 ms on a specific system to hash a dictionary password, it follows that the attacker requires over two hours on average to iterate a 150000 word dictionary and locate the correct one, on the same system. Thus, KDFs provide better protection, even for poorly chosen passwords.

7 Future Work

The OffPAD device may support several authentication mechanisms, not limited to challenge-response protocols. The OffPAD may function as an identity provider in itself, for example as an OpenID provider for the owner. It can also be used as an encryption and signing device, and as a communicating device that facilitates encrypted communication.

When introducing KDFs in the DAA scheme, we require some extra parameters to the one-way function. Where a hash function takes only one value, namely the data to hash, KDFs require (at least) an additional two: The number of iterations (or *workload*) of the hash function and a salt. There are no fields for extra values in the original DAA scheme. One might consider passing the function parameters in their own fields, `workload` and `salt`, which would of course require changes to the authentication framework. Also, it is possible to either pass the parameters along with the `algorithm` field, such as `algorithm=scrypt:1000:Base64([salt])` or in the `nonce` field, along with the challenge.

If one is to use the OffPAD against a server that does only support original DAA, it is still possible to do so securely. The password can either be randomly selected from the key space of MD5 output (i.e 128 random bits), or have entropy that exceeds what is produced by MD5. This way, when the password is hashed, the simplest way to recover the protected password is to mount the best preimage attack, an infeasible calculation (as stated above, of complexity $2^{123.4}$). The passwords can be as long as a book – it is still only the hash that is stored.

8 Conclusion

We have shown how HTTP Digest Access Authentication can be extended, relocating authentication from a possibly compromised system to an external secure device – the OffPAD. We have presented the benefits of the extension, but also looked at some weaknesses of the current scheme, that are possibly present even when authenticating with the OffPAD. Suggestions have been proposed as to how we can replace the old and vulnerable single-iteration response calculation with modern adjustable key derivation functions or randomized passwords to evade brute-force attacks. These may provide additional protection to the passwords, both while in transit and when stored on each endpoint. The proposed OffPAD solution also improves the usability of user authentication. Storing and managing passwords on a secure device rather than in the brain is scalable, less concerning and removes the physical and cognitive strain of entering and remembering passwords.

References

[1] Adams, A., Sasse, M.A.: Users are not the enemy. Commun. ACM 42(12), 40–46 (1999)

[2] AlFayyadh, B., et al.: Improving Usability of Password Management with Standardized Password Policies. In: Rosenberger, C., Achemlal, M. (eds.) Proceedings of the 7th Conference on Network and Information Systems Security (SAR-SSI), pp. 38–45 (2012) ISBN: 978-2-9542630-0-7

[3] Berners-Lee, T., Fielding, R., Frystyk, H.: Hypertext Transfer Protocol–HTTP/1.0. RFC 1945 (Informational). Internet Engineering Task Force (May 1996), http://www.ietf.org/rfc/rfc1945.txt

[4] Fielding, R., et al.: Hypertext Transfer Protocol–HTTP/1.1. RFC 2616. Updated by RFCs 2817, 5785, 6266. Internet Engineering Task Force (June 1999), http://www.ietf.org/rfc/rfc2616.txt

[5] Franks, J., et al.: An Extension to HTTP: Digest Access Authentication. RFC 2069. Obsoleted by RFC 2617. Internet Engineering Task Force (January 1997), http://www.ietf.org/rfc/rfc2069.txt

[6] Franks, J., et al.: HTTP Authentication: Basic and Digest Access Authentication. RFC 2617. Internet Engineering Task Force (June 1999), http://www.ietf.org/rfc/rfc2617.txt

[7] Gosney, J.: Password Cracking HPC. Rump session, Passwords (December 12, 2012), http://passwords12.at.ifi.uio.no/Jeremi_Gosney_Password_Cracking_HPC_Passwords12.pdf (visited on December 17, 2012)

[8] Gourley, D., Totty, B.: HTTP: The Definitive Guide. O'Reilly & Associates, Inc. (2002)

[9] Jøsang, A.: Identity Management and Trusted Interaction in Internet and Mobile Computing. IET Information Security (in press, 2013)

[10] Jøsang, A., Pope, S.: User Centric Identity Management. In: AusCERT Conference 2005 (2005)

[11] Inc. Juniper Networks. Juniper Mobile Threat Report 2011. Tech. rep. Juniper Networks, Inc. (2011)

[12] Kaliski, B.: PKCS #5: Password-Based Cryptography Specification Version 2.0. RFC 2898 (Informational). Internet Engineering Task Force (September 2000), http://www.ietf.org/rfc/rfc2898.txt

[13] Laurie, B., Singer, A.: Choose the red pill and the blue pill: a position paper. In: Proceedings of the 2008 Workshop on New Security Paradigms, pp. 127–133. ACM (2009)

[14] Mannan, M., van Oorschot, P.C.: Leveraging personal devices for stronger password authentication from untrusted computers. Journal of Computer Security 19(4), 703–750 (2011)

[15] Morris, R., Thompson, K.: Password Security: A Case History. Communications of the ACM 22, 594–597 (1979)

[16] MozillaZine. Password Manager - MozillaZine Knowledge Base. (December 2011), http://kb.mozillazine.org/Password_Manager (visited on December 18, 2012)

[17] Panda Security PandaLabs. PandaLabs Quarterly Report (June 2012), http://press.pandasecurity.com/wp-content/uploads/2012/08/Quarterly-Report-PandaLabs-April-June-2012.pdf (visited on November 01, 2012)

[18] Percival, C.: Stronger Key Derivation Via Sequential Memory-Hard Functions. In: BSDCan 2009: The Technical BSD Conference (2009)

[19] Sasaki, Y., Aoki, K.: Finding Preimages in Full MD5 Faster Than Exhaustive Search. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 134–152. Springer, Heidelberg (2009)

[20] Sasse, M.A., Flechais, I.: Usable Security Why Do We Need It? How Do We Get It? In: Security and Usability: Designing Secure Sys Tems that People Can Use, pp. 13–30. O'Reilly Books (2005)

[21] Stajano, F.: Pico: No More Passwords! In: Christianson, B., Crispo, B., Malcolm, J., Stajano, F. (eds.) Security Protocols 2011. LNCS, vol. 7114, pp. 49–81. Springer, Heidelberg (2011)

Executable Model-Based Risk Assessment Method for Identity Management Systems

Ebenezer Paintsil and Lothar Fritsch

Norwegian Computing Center Oslo, Norway
{paintsil,lothar.fritsch}@nr.no

Abstract. Currently, risk assessment methods for identity management systems (IDMSs) are lacking. This makes it difficult to compare IDMSs based on how they enhance privacy and security of system stakeholders. This article proposes the executable model-based risk assessment method (EM-BRAM) with the aim of addressing this challenge. The EM-BRAM identifies risk factors inherent in IDMSs and uses them as inputs to a colored petri nets (CPNs) model of a targeted IDMS. It then estimates or verifies the system's security and privacy risks using CPNs' state space analysis and queries.

1 Introduction

Identity theft crimes online is ever increasing. In 2010, the total cost of online credit card fraud alone was estimated as 4.2 billion US dollars [1]. Currently, identity related crimes are among the fastest growing crimes in the United Kingdom [2].

Privacy enhance IDMSs have the potential of mitigating these crimes and privacy risks. Privacy enhancing IDMSs can allow end-users to act under pseudonyms, to be unlinkable and control the use and release of their partial identities [3]. However, privacy enhance IDMSs greatly differ in their security mechanisms. They prescribe different security mechanisms and focus on different problem areas. The inconsistent and complex mechanisms make systems' comparisons a daunting task for system stakeholders.

Privacy and security risks assessment methods can be used to compare IDMSs in order to improve their security and also select the appropriate systems for stakeholders. Currently, privacy and security risks assessment methods for IDMSs are lacking [4]. Moreover, the traditional risk assessment approaches such as ISO27005 [5] provide no explicit inputs for risk assessment of IDMSs. For instance, the ISO27005 [5] has no explicit risk model for IDMSs.

Furthermore, the traditional risk assessment approaches rely on the intuitions of risk assessors to estimate risk because of lack of data thereby making the process error prone [6], [7].

This article introduces the executable model-based risk assessment method (EM-BRAM) for IDMSs. The method identifies risk factors inherent in IDMSs and uses them as inputs to a colored petri nets (CPNs) model of a targeted IDMS to estimate or verify the system's risk. The method has the potential of reducing subjectivity and uncertainty in risk assessment of IDMSs.

S. Fischer-Hübner et al.(Eds.): IDMAN 2013, IFIP AICT 396, pp. 97–99, 2013.

2 Executable Model-Based Risk Assessment Method

This section introduces the executable model-based risk assessment method for IDMSs. The method relies of the characteristics of tokens used in IDMSs to estimate a system's risk. The first step in the method is risk identification focusing on the characteristics of tokens that can contribute to privacy and security risks in IDMSs. We focus on tokens used in IDMSs because they are personal data sources and gateway to resources. A token is a technical artifact providing assurance about an identity [8]. A token can be an identifier such as username, a claim such as a password, an assertion such as SAML tokens, a credential such as X.509 certificate or combinations of these.

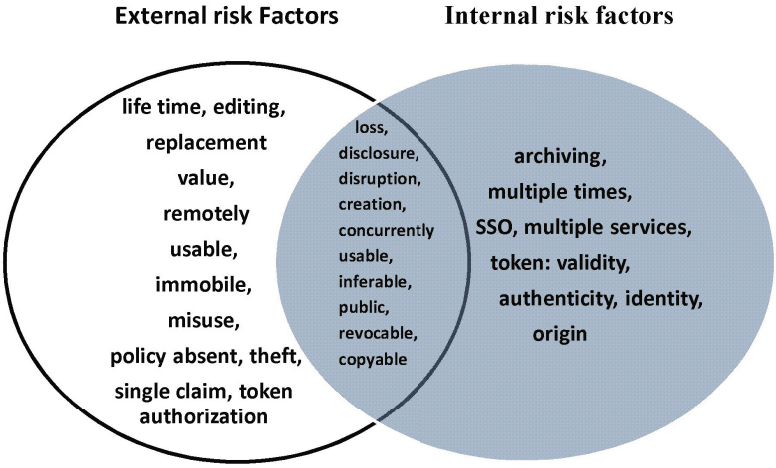


Fig. 1. External and Internal Risk Factors [8]

The characteristics of tokens that contribute to privacy and security risks are identified and categorized into external and internal factors as shown in Figure 1. The internal characteristics serve as the input for the CPNs model of the IDMSs.

The second phase of the risk analysis is risk estimation focusing on the internal factors. Internal factors are those under the control of IDMSs while external factors are outside the control of the IDMS. While internal factor are estimated with CPNs modeling, the external factors may be used for policy decision.

We verify or estimate the internal risk through CPNs [9] modeling because it would be difficult to manually verify how tokens flow in IDMSs and which risk state they find themselves. In addition, CPNs are less mathematical and have a high degree of automation making it relatively easy to use. The automation can potentially reduce cost involved in the risk assessment process and subjectivity in the risk estimation.

The system modeling is followed by the validation of the behavior properties of the system. This enables us to determine the behavioral correctness of the IDMS's model before risk verification or estimation. The validation is automated with CPNTools [9].

Finally, the privacy and security risks are estimated or verified using CPNs queries and ML predicate functions. The queries search through all the execution states of the IDMS model to verify if the risk conditions exist. For example, the following CPNs query can be used to verify whether a token is used for multiple services or single sign-on (SSO). Although SSO reduces human error, it leads to sharing of valuable information across services or domains.

```
fun isMultipleServices()= fn n => isSubstring "bob"
(st_Mark.GoogleSP'ReceivedAssertion 1 n);
```

The query verifies if the alias "'bob"' can be found outside a trusted domain.

3 Conclusion

Lack of risk assessment method for identity management systems (IDMSs) makes it difficult to compare them based on their security and privacy risks levels. This article, proposes the executable model-based risk assessment method for IDMSs. The method can be used to compare IDMSs based on their security and privacy risks levels and has the potential to improve privacy and also reduce the subjectivity inherent in traditional risk assessment methods.

References

[1] Levi, M.: Measuring the cost of cybercrimes. ERCIM News (90) (2012)
[2] Leyden, J.: Id fraud prevention week fights uk's fastest growing crime (2009)
[3] WP3: D3.1: Structured overview on prototypes and concepts of identity management systems. Deliverable 1.1, Future of Identity in the Information Society (2005)
[4] Cabarcos, P.: Risk assessment for better identity management in pervasive environments. In: IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), pp. 389–390 (2011)
[5] ISO: Iso 27005 information security risk management. Technical report, International Organization for Standardization (2008)
[6] Aven, T.: A semi-quantitative approach to risk analysis, as an alternative to qras. Reliability Engineering & System Safety 93(6), 790–797 (2008)
[7] Campbell, H.: Risk assessment: subjective or objective? Engineering Science and Education Journal (1998)
[8] Paintsil, E.: Evaluation of privacy and security risks analysis construct for identity management systems. IEEE Systems Journal PP(99), 1 (2012)
[9] Kurt, J., Lars, K.M.: Colored Petri Nets: Modelling and Validation of Concurrent Systems: Modeling and Validation of Concurrent Systems. Springer, Heidelberg (2009) ISBN:978-3-642-00283-0

Position Paper: Privacy Risk Analysis Is about Understanding Conflicting Incentives*

Einar Snekkenes

NISlab, Gjøvik University College

Abstract. We motivate and give a brief overview of the Conflicting Incentives Risk Analysis (CIRA) method, explaining key ideas and concepts, offering a small example and give an overview of remaining challenges in relation to the use of CIRA in large scale risk analysis projects.

1 Introduction

Privacy may be motivated by a multitude of reasons, e.g. its intrinsic value, legal or regulatory compliance or a general concern regarding improper use of personal identifiable information. For example, if an employer, insurance company or the press got hold of the medical record of a known individual, it may turn out to be very difficult to prevent this information from being used in a way that is unfavourable to the individual.

In this paper we restrict our attention to risks that emanate from deliberate actions. By risk we mean the concern relating to an undesirable surprise (i.e. a loss) caused by the implementation of some action. Risk is always subjective and relative to perception. E.g. I may have a concern in relation to the uncertainty regarding the publication of my medical records, i.e. I experience risk. We can extend this notion of risk to include opportunity risk. By opportunity risk, we mean the uncertainty that an individual will fail to seize the opportunity to implement actions that one could reasonably expect that he should implement.

Risk analysis answers the question: Are there any risks that require some kind of action by the individual exposed to the risk? Risk management is about implementing the necessary action to ensure that unacceptable risks are mitigated.

There is a need to improve the predictability and the coverage of the risk identification process in the context of intended human behaviour. This challenge is a consequence of limited availability of representative historic data relevant for new and emerging systems. Furthermore, to improve the efficiency of the discovery process, there is a need to identify issues that are key to risk discovery, and avoiding activities that shed little or no light on potential problem areas. In the following, we explain how the Conflicting Incentives Risk Analysis (CIRA) method[1] adresses these issues.

* This work is part of the PETweb II project sponsored by The Research Council of Norway under grant 193030/S10.

S. Fischer-Hübner et al.(Eds.): IDMAN 2013, IFIP AICT 396, pp. 100–103, 2013.

2 Summary of CIRA

CIRA identifies stakeholders, actions and perceived expected consequences that characterizes the risk situation. According to CIRA, we have risk if the stakeholder that is in the position to trigger the action and the risk taker would be in disagreement as to weather or not the action should be implemented.

In CIRA, a *stakeholder* is an individual (i.e. physical person) that has some interests relating to the outcome of actions that are taking place within the scope of interest. There are two classes of stakeholders: the *action owners* and the *risk owner*. The *action owner* is in the position to decide if and when the action in question is to be executed. Typically, each stakeholder has associated a collection of actions that he owns. The *risk owner* is the stakeholder whose perspective we take when performing the risk analysis - that is, he is the stakeholder at risk. By *utility* we mean the benefit as perceived by the corresponding stakeholder. Utility comprises of *utility factors*. Chule et. al..[2] identify utility factors relevant for our work. Each utility factor captures a specific aspect of utility e.g. prospect of wealth, reputation, legal compliance, social relationships.

The CIRA tasks identify stakeholders, actions and consequences of actions in terms of perceived value changes to the utility factors. Our first task is to identify who is to take the role of the risk owner (e.g. the data protection officer, a politician that has some ideological desire to create a society where privacy is a common good, the prime minister, a typical citizen). Next, we need to identify what utility factors go into the risk owners perception of utility. I.e. how these utility factors are defined and how the various utility factors are weighted, reflecting the trade-off judgements made by the risk owner. This is required to assess how actions change the values of the various utility factors and thus influence changes to the risk owners perception of the utility offered by the action. We estimate the cumulative utility by using techniques from Multi Criteria Decision Analysis[3].

We then go on to identify the other stakeholders of interest. That is, we identify the stakeholders who are in the position to implement actions that will modify the values of the utility factors (and weights) of the risk owner. For each of these stakeholders, we identify the actions that are at their disposal and to what extent they modify the risk owner's utility factors. See e.g. [4] for a taxonomy of actions relevant in a privacy context.

When modelling the extent to which actions modify utility factors, care must be taken to make sure that the complete picture is captured. In particular, each action can be viewed as a strategy in a potentially complex game[5], where the implementation of the action amounts to the participation in a game.

In short, CIRA identifies situations such as the following: for all the actions that can be taken, can those that are in the position to implement an action obtain a significant benefit and at the same time cause damage to the risk owner (in terms of loss of utility)? Are there actions that one can reasonably expect that the action owner should take, but for which the action owner would have to take a loss in utility and the risk owner have the prospect of a gain?

3 A Small Example

We consider a situation, involving a bank and a customer. The bank is offering financial advice to the customers. The advice is offered by the bank through a financial adviser. The question is: Is the customer at risk? I.e. is there a legitimate concern by the customer in relation to the uncertainty that the bank is set up such that the customer may receive bad advice? From the perspective of the risk analyst, is there an easy procedure that will help the risk analyst to determine if the customer is at risk?

We assume that the financial adviser is rational in the classical economic sense in that the only factor that he considers when providing advice to customers is the monetary value of the bonus he receives if the customer buys the product he suggests. The financial adviser may obtain personal information about customers such as mental capacity, education, financial maturity, degree of dyslexia, relationship status, mental problems etc. through the operators of an identity management system. The bank has very few ethical standards, and those that exist are not enforced. There is at least one financial service offered by the bank that provides the adviser with a significant bonus, and may result in a significant loss for the customer (A1). There is also a service offered by the bank where the advisor will have to do a significant volume of work without this effort resulting in what he perceives to be a fair return in terms of utility and where the customer is expected to gain significant utility (A2).

Applying the CIRA method, we easily see that the customer is facing the following risks: Being offered service A1 and accepting the offer. Not being offered service A2, and consequently not being able to take advantage of it. In both cases, we have situations of conflicting incentives.

In the context of CIRA, risk mitigation amounts to modifying the weights that the stakeholders assign to the relevant utility factors or to what extent actions modify the values of the utility factors. In the above example, changing the weights of the utility factors can be achieved by e.g. forcing financial advisers to complete ethical training and/or using empathy or ethics screening tests before hiring. Changing the impact that actions have on utility factors of the advisers can be realized by modifying the bonus scheme - e.g. converting any bonus awarded to a liability, and offering the customer a compensation that exceeds the customer's loss if there is a complaint from the customer that he received unfavourable advice. This new bonus rule must then be made available to all concerned.

4 Challenges and Limitations

The CIRA method is still at an early stage of development. The method may benefit from further work on issues such as: Identification of a more broad action repertoire, e.g. along the lines of the taxonomy relevant in a privacy context[4], the capturing of uncertainties in relation to estimates using e.g. interval arithmetic [6] or bounded probabilities[7]. We also need a more comprehensive taxonomy of utility factors to capture human goal directed behaviour to state but

a few. The use of intervals or bounded probabilities instead of point values will provide a link between quantitative and qualitative interpretation of CIRA. In many cases, the risk analyst may not have access to the relevant individuals. Then the utility factors and their trade-off may have to be obtained from past behaviour, profiles or from distributions obtained from similar individuals. Some stakeholders may find the CIRA approach to risk analysis rather intrusive, trying to game the analyst, failing to provide the analyst with a correct set of utility factors and/or their weights. The combination of utility factors using linear weights may fail to correctly model the stakeholders real assessment of utility e.g. in the presence of threshold values and utility factor dependencies.

5 Conclusions

We have given an overview of CIRA and explained how this method can be used to identify privacy risks. Using CIRA, the analyst is provided with guidance with respect to the key issues that are the root sources of many risks. We argue that in spite of its lack of maturity, CIRA offers concepts and procedures that will improve the process of identifying and analysing risks relating to intended human behaviour.

References

1. Rajbhandari, L., Snekkenes, E.: Intended Actions: Risk Is Conflicting Incentives. In: Gollmann, D., Freiling, F.C. (eds.) ISC 2012. LNCS, vol. 7483, pp. 370–386. Springer, Heidelberg (2012)
2. Chulef, A., Read, S., Walsh, D.: A hierarchical taxonomy of human goals. Motivation and Emotion 25, 191–232 (2001)
3. Greco, S. (ed.): Multiple Criteria Decision Analysis: State of the Art Surveys. International Series in Operations Research & Management Science. Springer (2005)
4. Solove, D.J.: A Taxonomy of Privacy. University of Pennsylvania Law Review 154(3), 477 (2006)
5. Rasmusen, E.: Games and Information: An Introduction to Game Theory, 4th edn. Wiley-Blackwell (2006)
6. Moore, R.E., Kearfott, R.B., Cloud, M.J.: Introduction to Interval Analysis. SIAM (2009)
7. Ferson, S., Hajagos, J.G.: Arithmetic with uncertain numbers: rigorous and (often) best possible answers. Rel. Eng. & Sys. Safety 85(1-3), 135–152 (2004)

Risk Analysis of Identity Management Approaches Employing Privacy Protection Goals

Marit Hansen

Unabhängiges Landeszentrum für Datenschutz Schleswig-Holstein, Kiel, Germany
marit.hansen@privacyresearch.eu

Abstract. This position paper introduces the approach of privacy protection goals for risk analysis in identity management. It pleads for taking into account external factors such as the data collection via other applications or upcoming legal legislation.

Keywords: Identity Management, Privacy Protection Goals.

1 Protection Goals for Information Security and Privacy

For decades, professionals in information technology have been working with the so-called classic triad of the protection goals "confidentiality", "integrity" and "availability" to assess security properties and risks of information systems. Since 2009, an extended set of protection goals has been proposed that also consider the privacy perspective, and thereby better reflect the interests of the individual whose personal data are being processed [1,2]: In addition to the classic three security protection goals, three complementing privacy protection goals have been introduced:

- "unlinkability": Unlinkability ensures that privacy-relevant data cannot be linked across privacy domains or used for a different purpose than originally intended. Thereby it addresses both the legal principles of data minimization and purpose binding.
- "transparency": Transparency ensures that all privacy-relevant data processing including the legal, technical and organizational setting can be understood and reconstructed. Transparency is the precondition for all kind of user decision, e.g. for giving consent.
- "intervenability": Intervenability ensures that data subjects, operators and supervisory authorities can intervene in all privacy-relevant data processing. Its objective is the application of corrective measures and counterbalances where necessary. The data subject's rights to rectification or erasure are an example for intervenability.

Since beginning of 2012 the State Data Protection Act Schleswig-Holstein (LDSG S-H), Germany demands that data controllers take into account the three security protection goals as well as the three privacy protection goals (Art. 5 par. 1 LDSG S-H). They have also been taken up in the draft report of the European Parliament on the European Data Protection Regulation [3].

S. Fischer-Hübner et al. (Eds.): IDMAN 2013, IFIP AICT 396, pp. 104–106, 2013.

2 Identity Management and Privacy Protection Goals

The six protection goals are not independent from each other. Working with protection goals means to identify how far each of the goals should be implemented and to find a suitable balance between those goals, taking into account the interests of all parties involved. Privacy-enhancing and user-controlled identity management has usually strong requirements concerning the protection of personal data against unwanted linkage across domains or contexts [4]. However, unlike an anonymizing system that aims at full unlinkability, several interactions with the same communication partner should in many cases be linkable for the parties participating in the communication (not for potential observers) which has effect on the choice and (re-)use of pseudonyms. Transparency (beforehand and afterwards) is important for users to understand what the data processing is about, e.g. to enable them to select which attribute values to disclose. In case a relying party asks for information that is not appropriate for the given purpose, users must be able to intervene, i.e. to stop a transaction, to limit the disclosure or to send a complaint.

Note that the protection goals are not only meaningful for designing information and communication technology systems, but can also be used to check organizational procedures or legal regulations.

3 Considering External Factors, Too

For analyzing risks concerning the desired level of guarantees per protection goal, the system scope should not be too narrow. For instance, even photos that have not been biometrically optimized such as in the ePassport pose the growing risk of being linked across domains and contexts because of the big collections via social networks or search engines and the available technologies of biometric matching.

Another huge risk for all European identity management may be caused by the regulation on electronic identification and trust services for electronic transactions (eID Regulation) in case it is passed in the draft version that was presented by the European Commission in June 2012 [5]. The proposed regulation aims at removing barriers in the internal market for electronic interactions. Each Member State may notify an electronic identification scheme that it accepts to access public online services. All Member States must accept the foreign notified schemes. The draft eID Regulation demands that each Member State sets up at least one national online authentication service for their notified eID schemes that is available for relying parties requesting to check the link to a user authenticating with the eID. This online authentication service is liable for the unambiguity of the link to a citizen. On the ground of "technology neutrality" Member States must not impose any specific technical requirements (e.g. obtaining hardware or software) on relying parties.

As elaborated in [6], such a regulation would likely lead to several risks to the user's privacy:

- The need for one always available online authentication service probably means that centralized services would be given preference.
- Liability for the unambiguity of the link between the eID and the citizen would prevent anonymous authentication.
- The liability also would make it necessary for the services to store logfiles on the individual transactions to prove that they did it right. These logfiles – containing the information of users and relying parties involved in cross-border authentication – would have to be retained for as long as authentication result may be challenged.
- eID systems that provide selective disclosure of attributes are not foreseen in the proposed regulation. Such systems would require that relying parties install at least some software which the draft regulation negates.

All in all, the protection goal unlinkability is severely violated by these risks. The other two privacy protection goals, transparency and intervenability, are less addressed in the draft regulation and depend on the implementation. However, in particular the part on liability would have to be clarified concerning transparency and intervenability. It is necessary to amend the eID Regulation so that privacy-enhancing identity management solutions can be used or are even made mandatory for cross-border authentication. Best practice approaches can help to stop a race to the bottom concerning the level of implemented privacy.

References

1. Rost, M., Pfitzmann, A.: Datenschutz-Schutzziele – revisited. Datenschutz und Datensicherheit (DuD) 33(12), 353–358 (2009)
2. Hansen, M.: Top 10 Mistakes in System Design from a Privacy Perspective and Privacy Protection Goals. In: Camenisch, J., Crispo, B., Fischer-Hübner, S., Leenes, R., Russello, G. (eds.) Privacy and Identity 2011. IFIP AICT, vol. 375, pp. 14–31. Springer, Heidelberg (2012)
3. Albrecht, J.P.: Draft Report on the proposal for a regulation of the European Parliament and of the Council on the protection of individual with regard to the processing of personal data and on the free movement of such data (General Data Protection Regulation) (COM (2012)0011 – C7-0025/2012 – 2012/0011(COD)), Committee on Civil Liberties, Justice and Home Affairs (December 17, 2012), http://www.europarl.europa.eu/sides/getDoc.do?language=EN&reference=PE501.927
4. Zwingelberg, H., Hansen, M.: Privacy Protection Goals and Their Implications for eID Systems. In: Camenisch, J., Crispo, B., Fischer-Hübner, S., Leenes, R., Russello, G. (eds.) Privacy and Identity 2011. IFIP AICT, vol. 375, pp. 245–260. Springer, Heidelberg (2012)
5. European Commission: Proposal for a Regulation of the European Parliament and of the Council on electronic identification and trust services for electronic transactions in the internal market. COM(2012) 238/2, Brussels, 2012 (July 04, 2012), http://ec.europa.eu/information_society/policy/esignature/docs/regulation/com_2012_2038_en.pdf
6. ABC4Trust: Privacy-ABCs and the eID Regulation. Position Paper (2013), https://abc4trust.eu/

The Radboud Reader: A Minimal Trusted Smartcard Reader for Securing Online Transactions

Erik Poll and Joeri de Ruiter

Institute for Computing and Information Sciences, Digital Security Group,
Radboud University Nijmegen, The Netherlands

Abstract. We present the design of a device for securing online transactions, e.g. for internet banking, which can protect against PC malware, including Man-in-the-Browser attacks. The device consists of a USB-connected smartcard reader with a small display and numeric keyboard, similar to devices currently used for internet banking. However, unlike existing devices, we rigorously stick to the design philosophy that the device should be as simple as possible; move functionality and control is moved as much as possible to the smartcard. Although this is a simple (and obvious) idea, we are not aware of any solutions pursuing it. Moreover, it has some interesting benefits compared to existing solutions: the device is simpler, provides stronger security guarantees than many alternatives (namely that it will only display text authenticated by the smartcard), and is generic in that it can be used in combination with different smartcards for different applications (for example, for internet banking with a bank card and for filing an online tax return with a national ID card).

1 Introduction

A fundamental problem in securing online transactions is the lack of a trustworthy device to communicate with the human end user. The software on laptops and PCs, but also smart phones, is so large and complex that security vulnerabilities are inevitable. This means that these devices are not trustworthy as input or output channel to communicate to the user (via the display, keyboard or mouse), as malware can manipulate the display and eavesdrop on any input.

Smartcards are a potential improvement in that they provide a secure computing platform. The chances of exploitable security vulnerabilities in the software on a smartcard are *much* lower than for a PC or laptop, given the very small size and the very limited functionality of the code. However, a serious and fundamental limitation of smartcards is the lack of a display and keyboard.[1] This means that a smartcard requires some terminal with a display for output and a keyboard for user input, and this terminal has to be trusted.

[1] Still, the first smartcards with keyboard and display are in commercial use.

S. Fischer-Hübner et al.(Eds.): IDMAN 2013, IFIP AICT 396, pp. 107–120, 2013.

Given the issues above, the more secure solutions that use smartcards to secure online transactions rely on a dedicated smartcard reader with a numeric keyboard and display. The most prominent example is the EMV-CAP standard for internet banking [3,10]. Some of these devices are connected by USB to the computer, which can both increase user convenience – as the user does not have to type over numbers to and from the device – and security, by providing a 'What You See is What You Sign' guarantee. Section 5 compares various existing solutions.

This paper presents an alternative design for a solution, the Radboud Reader, where we rigorously stick to the design principle that the device should be as simple as possible. Where possible functionality is moved to the smartcard rather than the reader and the smart card is given control as much as possible. Although these principles are rather obvious, and the resulting system is quite a natural solution, we are not aware of anyone else proposing a system like this. The resulting system which we present in this paper has some interesting advantages over existing solutions:

- The device is *simpler* than alternative solutions: it contains no secrets, and does not need to support any cryptographic operations or even hashing. Keeping the device simple also reduces the chance of security vulnerabilities in the device.
- Our solution gives *stronger security guarantees* than most other solutions: the device provides a trusted display which only displays text approved by the smartcard, so that the smartcard can check say a digital signature on any text before it is displayed.
- The device is *generic* and can be used in combination with different smartcards for different purposes.

Section 2 describes the high-level design of the system, which is discussed in more detail in Section 3. Section 4 discusses our prototype, Section 5 compares our solution with existing devices and Section 6 concludes.

2 Security Objectives and High-Level Design

The basic set-up, shown in Fig. 1, is the same as with any USB-connected smartcard reader for internet banking: a web browser on a PC communicates with a remote web server over the internet and with a smartcard reader via a USB cable. The smartcard reader has its own display and numeric keypad with two additional buttons marked 'OK' and 'Cancel', so the user can interact with both the PC and the reader for input and output, via their respective displays and keyboards.

The objective is that, unlike the PC, the reader can be a trusted input and output device, meaning that we rely on

S1 authenticity of whatever is displayed on the display,
S2 confidentiality of anything that is entered on the keyboard,
S3 non-repudiation of any transactions confirmed or declined by pressing 'OK' or 'Cancel'.

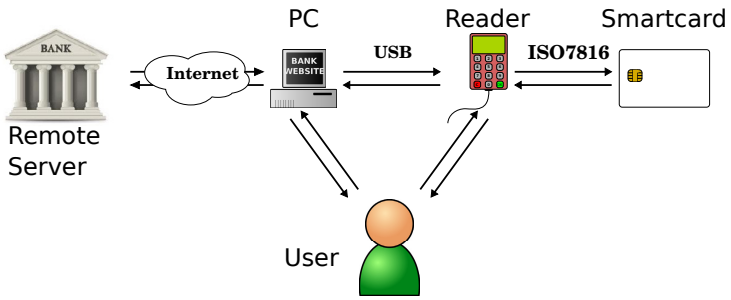


Fig. 1. Set-up

Although a complete transaction might be performed using only the reader, this is not very user friendly given its limited display and numeric-only keyboard. Therefore, the PC can still be used to set up the connection to the service provider and enter the transaction details.

2.1 Attacker Model

The attacker is assumed to be in full control of the network and of the PC, including the USB connection to the reader, but not of the reader or the smartcard, and the communication between them. If one abstracts away from the PC then this is equivalent to assuming a Dolev-Yao attacker [2] on the communication between the remote server and the reader.

Our main concern is an online attacker, with possibly total control over the PC, but without physical access to the reader or the smartcard. Attackers with physical access are only a secondary concern. This means that physical tamper-resistance or tamper-evidence of the reader is not crucial: they are nice properties to have, but in practice one will only want to spend a limited amount to realising them to some degree. Of course, one would want to include protection against shoulder surfing, e.g. by not echoing say a PIN code on the display as it is entered on the reader.

2.2 High Level Design Decisions

For the reader to be a trusted input and output device, we will ensure that

- all output on the display comes from the smartcard, and
- all input to the device is only sent to the smartcard,

as illustrated in Fig. 2. This means that the smartcard can ensure authenticity of the output to the screen, and confidentiality of the input from the keyboard.

By only allowing the smartcard to output to the display, we get a trusted display, where authenticity of displayed messages can be enforced by the smartcard. Allowing the PC to display text would allow malware on the PC to control

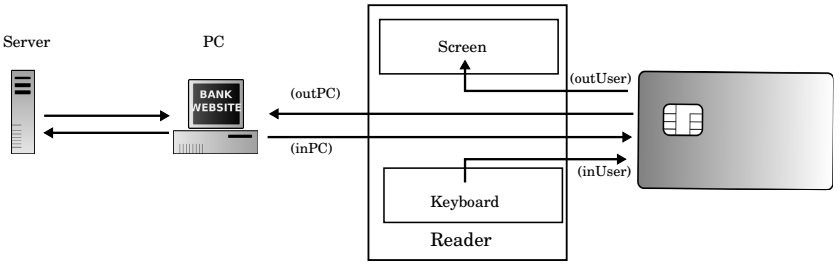


Fig. 2. Information flows in the Radboud Reader. NB all I/O the device offers to the user is with the smartcard, not the PC, and all communication between the user and remote server has to pass through the smartcard.

the display, even though the device could still offer the guarantee of 'What You See is What You Sign'.

So *physically* the reader is between the smartcard and the PC, but *logically* – i.e. considering the information flows – the smartcard is between the PC and the reader.[2]

For the input the Radboud Reader provides a numeric keyboard as well as an 'OK' and 'Cancel' button. Output can be displayed on its display consisting of 80 characters (4 x 20).

Note that it is completely up to the smartcard to decide what functionality it provides, and how the data communicated with the PC, and via the PC with the remote server, is secured. Different strategies are possible here:

- The smartcard could set up a secure tunnel to the remote server, analogous to TLS, ensuring integrity and confidentiality of the entire communication session between smartcard and the remote server. In such a set-up, one could let the smartcard *only* communicate with the back-end, and not communicate with the PC at all.
  Many smartcards already provide such a secure tunnelling mechanism, called Secure Messaging in smartcard jargon. The ISO/IEC 7816 standard [9] already describes it, as do many other smartcard standards, including the ICAO standard for electronic passports [8], the EMV standard for payment cards [4], and the Global Platform standard [6].
- Alternatively, one could choose to sign and/or encrypt parts of messages exchanged between the smartcard and the remote server on a more piecemeal basis.

The former approach provides simpler and more robust security. An advantage of the latter approach might be that the browser can see and display some parts

[2] One could even consider using physically different contacts on the smartcard for communication with the display and communication with the PC. Two of the contacts of smartcard are reserved for future use in the ISO/IEC 7816 standard, so this is possible. However, this would be a more expensive solution requiring non-standard smartcards.

of the communication between smartcard and PC, which in the former approach would require additional communication between browser and the back-end.

To login to a website using the reader in combination with a smartcard, one could of course let the smartcard generate a credential to login, using some challenge-response protocol. It is even possible to let the smartcard to supply the username (or say, in the case of bank account, the bank account number) to the remote server when logging in over a secure tunnel, in which case malware on the PC could not even be able to learn this. Paper receipts for credit card transaction no longer show the complete card number but only the last four digits. Similarly, when using the Radboud reader in combination with a smartcard to log-on to some website, there is no reason to show the actual login name on the PC's display if leaking this could give useful information to an attacker.

Inserting a smartcard into a smartcard reader that is attached to a (potentially infected) PC is of course dangerous. Malware on the PC could try to access functionality of the smartcard in unwanted ways, for example by sending PIN code guesses to the card, with a small chance of guessing it right, and a big chance of blocking the card with 3 incorrect guesses. This leads to another security requirement, namely that access to functionality of the smartcard is strictly limited, which we will realise by ensuring that

S4 Input from the PC is forwarded to the smartcard in a specific format so that it cannot address arbitrary functionality on the card.

Effectively, the reader should provide a firewall between the smartcard and the PC, which only lets minimal functionality through.

That the device uses a USB-connection to a PC to connect to the internet and the back-end is not essential; any means of connecting the device to the internet could be used. USB is the obvious choice in that it is widely available, and indeed existing smartcard readers use it. An alternative to this would be to connect with the PC via Bluetooth. Another (more expensive) alternative would be to simply equip the device with GSM and let it by-pass the PC completely, though that requires additional measures to link a session on the device with a session on a PC.

2.3 Functional Requirements

To achieve the properties discussed previously, the Radboud Reader needs to provide the following operations:

1. Starting a session, by selecting the desired application on the smartcard; a smartcard can hold several applications, and one has to be selected at the start of a session.
2. Forwarding data received from the PC to the smartcard.
3. Carrying out instructions received from the smartcard; these instructions can tell the reader to
   a) display text and wait for the user to press 'OK' or 'Cancel';
   b) display text and wait for user input;

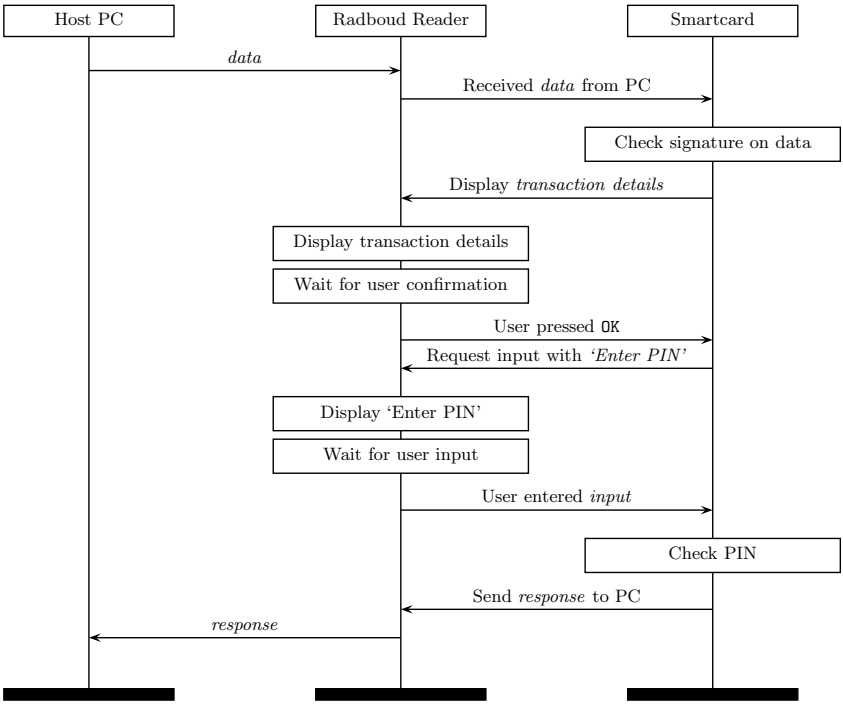


Fig. 3. Typical usage scenario

c) forward data from the smartcard back to the PC, and then wait for new data from the PC.

4. Sending user input from the keyboard to the smartcard, following 3a) or 3b).

Using these operations the smartcard can then construct any interaction with PC and the user that it wants. A typical usage scenario would be that the card receives digitally signed information over the internet and displays this on the device (ensuring that only genuine messages are displayed) and sends out digitally signed responses (to authenticate transactions). Of course, data received and sent can also be encrypted as well as signed to ensure confidentiality. It is up to the smartcard to check or put digital signatures and to en- or decrypt.

Fig. 3 illustrates such a scenario, where the data from the PC is forwarded to the card, who subsequently shows transaction data on the display. If the user agrees with this information, the card requests the PIN code from the user as input. If the PIN code is correct, the card generates a signature which is sent to the reader to be forwarded to the PC.

3 Detailed Design

To implement the functional requirements described above, several decisions have to be taken:

(i) How is the applet selected on the smartcard?
(ii) How do we forward data to the smartcard, when data is received from the PC (i.e. inPC in Fig. 2)?
(iii) How is the distinction made between output from the smartcard that is (a) destined for the display, (b) a request for input from the user, and (c) output destined for the PC? (I.e. between outUser and outPC in Fig. 2.)
(iv) How is input by the user forwarded to the smartcard (i.e. inUser in Fig. 2)? And how can the card distinguish between data coming from the keyboard and from the PC? (I.e. between inPC and inUser.)

These operations have to be realised at the level of the communication between the smartcard and the reader, as laid down in the ISO/IEC 7816 standard. At this level, all communication is via data packets called APDUs (Application Protocol Data Units). An APDU is simply a sequence of bytes in a fixed format. The smartcard is a slave in the master-slave communication between smartcard and reader: the reader sends a command to the smartcard, a so-called *command APDU*, and the smartcard answers with a *response APDU*.

The reader will have to present data to the smartcard in such a way that the smartcard can distinguish between inPC and inUser in Fig. 2. Similarly, the smartcard will have to provide its response in such a way that the reader can distinguish between options 3a), 3b) and 3c) listed earlier. Because we want the smartcard to be in control as much as possible, these responses from the smartcard will in fact be instructions [3] for the Radboud reader.

Selecting an application on the smartcard. The ISO/IEC 7816 standard uses AIDs (Application IDentifiers) to identify applications on a smartcard. An AID is simply a sequence of bytes of a fixed length. AIDs for certain applications have been standardised, and a smartcard can have one application that is selected by default.

Selecting the desired application could be done in several ways: by fixing a unique AID that is selected and hard-coding this in the reader, by hard-coding a list of AIDs that the reader attempts to select, or by letting the reader select the default applet. Letting the PC choose the applet to select is of course not a good option, as it could be abused by malware on the PC.

For simplicity, we choose to use a fixed AID to select the application on the smartcard. We choose an AID here that is not already used for an existing standard application. As soon as the smartcard is inserted in the reader, the application is selected.

Forwarding PC communications to the smartcard. One security-critical piece of functionality of the device is to provide a kind of firewall to shield the smartcard from malicious actions by the PC. One way of doing this would be to block all

[3] To avoid confusion with the standard terminology of commands for messages from the reader to the smartcard, we will call such messages from the smartcard to the reader *instructions* and not commands.

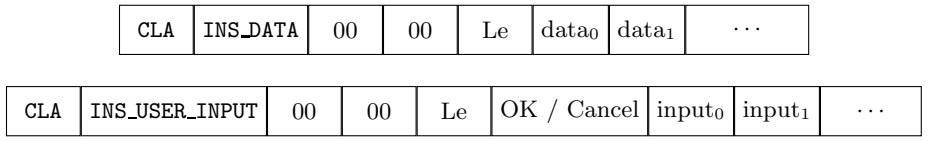


Fig. 4. Data formats for the command APDUs to pass PC data and user input to the card

traffic except a minimal white-list or a single instruction. Another would be to prepend data sent to the smartcard with a fixed prefix.

The former approach is a classical method for firewalls. The PC can send APDUs to the reader, who will forward them to the card only they are allowed according to its rules. These rules would have to be fixed for all possible applications, as they are stored in the reader. (One could make this APDU firewall configurable, with configuration controlled by the smartcard, but this introduces a lot of complexity.) Therefore it needs to be decided in advance what APDUs might be necessary and can be allowed to pass through.

We choose the latter approach, where the data received from the PC is wrapped as payload in an APDU with a dedicated instruction. Using this approach, the reader does not have to process the data it receives in any way: it simply forwards communication received from the terminal with a fixed prefix.

We are free to choose the prefix, but we should make sure this prefix does not have a meaning already in the ISO/IEC 7816 standard. Otherwise the reader could accidentally trigger functionality in a smartcard that was not designed to be used with the USB reader.

3.1 Format of Data Sent to the Smartcard

As explained above, the reader will simply forward data from the PC to the smartcard – i.e. inPC in Fig. 2 – with a fixed prefix. For user input on the device – inUser– the reader will use a different prefix, so that the card can distinguish them. Fig. 4 gives the data formats for this, which use the constants `INS_DATA` and `INS_USER_INPUT` as the so-called instruction byte (the second byte of the command APDUs):

- `INS_DATA` indicates that the APDU contains data forwarded from the PC. The data that is received from the PC is wrapped in an APDU and forwarded as it is.
- `INS_USER_INPUT` indicates that the APDU contains user input entered on the keyboard. The first byte of the data indicates whether the 'OK' or 'Cancel' button was pressed and, in case that 'OK' was pressed, the rest of the data gives the user input.

3.2 Format of Data Sent by the Smartcard

The three instructions that the smartcard can give to the reader use the data formats presented in Fig. 5, where the first byte specifies the instruction:

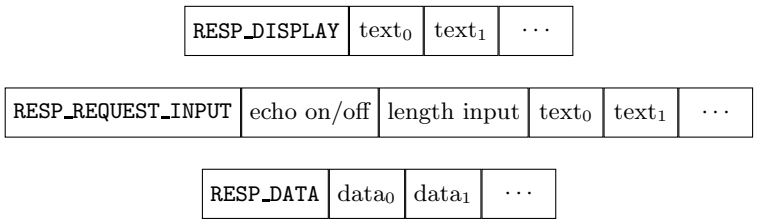


Fig. 5. Data formats for the response APDUs that provide instructions of the smartcard to the reader

- `RESP_DISPLAY` instructs the reader to display the text returned by the smartcard. The text to be displayed should not be longer than 80 characters for our prototype and is supplied after the `RESP_DISPLAY` instruction. When displaying data following a `RESP_DISPLAY` instruction, the reader waits for the user to either press 'OK' or 'Cancel'. This input is then sent to the smartcard using the `INS_USER_INPUT` command.
- `RESP_REQUEST_INPUT` instructs the reader to request user input. The second byte indicates whether input should be echoed, i.e. if the user can see the input on the screen or only masked characters. The third byte indicates the maximum input length. A string is appended to this that will be shown before the input. The length of this string together with the maximum input length cannot be longer than 80 characters for our prototype. After receiving a `RESP_REQUEST_INPUT` instruction, the reader lets the user input data using the keypad. The reader uses a `INS_USER_INPUT` instruction to return the result to the smartcard.
- `RESP_DATA` instructs the reader to forward the data returned by the smartcard to the PC and wait for new data from the PC.

After each command sent to the smartcard, the card can respond with a new instruction, making it possible to perform multiple `RESP_DISPLAY` and `RESP_REQUEST_INPUT` instructions before finally returning data to the PC using the `RESP_DATA` instruction.

4 Prototype Reader

To show the functionality of the Radboud Reader, we constructed a prototype using the Arduino platform[4] (see Fig. 6).[5] The code is approximately 350 lines of code and the components cost around 50 Euro. This prototype can demonstrate the feasibility of the approach and developing it was a useful exercise to make sure we resolved all implementation and design choices that have to be made in realising an implementation. Using this prototype protocols can be developed and tested.

[4] `http://www.arduino.cc`
[5] Available from `http://www.cs.ru.nl/~joeri/`

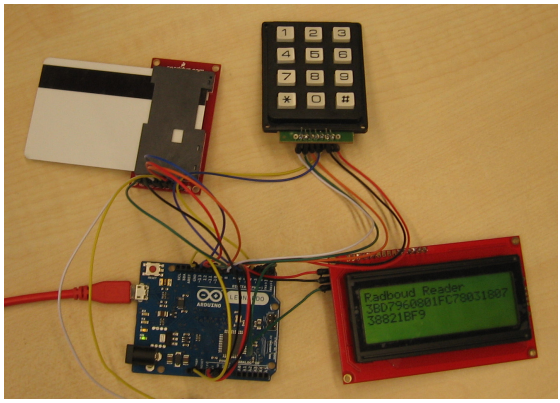


Fig. 6. Prototype based on Arduino platform

We implemented one sample protocol on an actual smartcard. The protocol provides integrity and confidentiality of communication between the smartcard and the issuer. For this, the smartcard contains a symmetric key, that is shared with the issuer, an asymmetric key pair and the public key of the issuer. The issuer also knows the shared symmetric key, the public key of the smartcard and its own asymmetric key pair.

The issuer starts the protocol by sending a command to initialise the protocol. The smartcard responds by sending a nonce and his identity, encrypted using the public key of the issuer. In response the issuer sends the smartcard's nonce together with his own nonce, both encrypted using the public key of the smartcard, back. The smartcard will then generate a random session key, which he encrypts using his shared symmetric key. To ensure integrity of the session key, as it is a random string, a MAC is computed over the encryption.

After the session key is established, so-called Secure Messaging is used to provide confidentiality and integrity of the communication session between the smartcard and issuer, as in an SSH or TLS tunnel.

The Session Sequence Counter (SSC), that is used to prevent replay, is initialised to the concatenation of the nonces of the issuer and the smartcard.

With Secure Messaging, first the data is encrypted using the session key. Subsequently a MAC is computed over the encrypted data. To prevent replays, the first block that is processed in the computation of the MAC is the SSC, whose value is increased by one after each message.

5 Related Work – Differences with Existing Devices

To compare our approach with existing devices, several security features can be taken into consideration:

1. The device may guarantee 'What You See is What You Sign' (i.e. guarantee that the transaction details shown on the display are in fact what is signed), but it may also provide a trusted display that can guarantee that any data shown on the display is authentic (i.e. originates from the service provider).
2. The device can support cryptographic operations, increasing the complexity of the device.
3. The device can include secrets, for instance in the form of secret keys (in case that the device can do cryptographic operations). But it is also possible that the functionality of the device is (partly) secret, in which case we may not be able to tell if it contains say a secret key or if it simply contains some unknown use of a hash function.
4. The device may be unconnected, and not provide any means of communication between it and the PC, or, in case it does, this communication may be bidirectional (e.g., in the case of USB), or uni-directional (e.g., in the case of optic communication from the PC to the device).
5. Finally, there is the question of how generic the device is, i.e. whether the device can be used for several purposes and users.

For some of these characteristics one can still argue whether they improve or weaken security. E.g., having secrets in the device makes it harder to produce a fake version or a software implementation of the devices, but makes generic use harder.

Table 1 gives a comparison of different devices based on these features. Below we discuss these devices in more detail.

Manufacturers typically do not publish technical details about the devices they produce, (though they do sometimes apply for patents, e.g. [7]). This means that for some devices we do not know all the features, unless the working has been reverse-engineered. Ideally vendors would provide this kind of information publicly so the clients can make a better comparison between different devices.

Unconnected smartcard readers with a display and keyboard are widely used for online banking. Many of these systems use EMV-CAP, a proprietary protocol of MasterCard, defined on top of the EMV standard [4]. This has advantages, namely that existing EMV smartcard implementations, which may have undergone costly security certifications, can be re-used, but also introduces the risk of ambiguities in different meanings of the same protocol messages [3]. When using these devices, the user receives a numeric challenge from the bank on his PC. This challenge and PIN code have to be entered on the reader, which then generates a response, using the user's smartcard, that has to be entered on the PC again. The challenges are often just random numbers with no meaning to the human user. They may also include account numbers or amounts, i.e. data more meaningful to the user, which then could protect against Man-in-the-Browser attacks at the expense of the user having to enter more input and relying on the user to know what the meaning of all parts of the challenge is.

EMV-CAP has been largely reverse-engineered [3,10]. In some variants the digital signature (or rather, a 3DES MAC) is computed in the device, and not by the smartcard [10]. Note that not having any cryptographic capabilities in the device, as we do, makes such a bad choice impossible.

Some banks already use a USB-connected reader for internet banking. Companies supplying solutions for this include VASCO and Gemalto. As far as we know, none of these devices provide a trusted display: they display data that is received over the USB cable in plaintext without any integrity checks. This means that malware on an infected PC could show anything it wants, and could even use the display of the reader as part of a sophisticated phishing attack. Even if these devices do not guarantee that what is shown on the display is authentic in any way, they can provide 'What You See is What You Sign' by sending the displayed text to the card to be signed.

One variant of a USB-connected reader, produced by Gemalto, has been reverse-engineered, and found to contain a security flaw [1]. This demonstrates once again the danger in relying on closed, proprietary solutions. It also provides further support for our design philosophy of keeping the device as simple as possible.

The Zone Trusted Information Channel, or ZTIC[6] from IBM comes closer to our approach in the security it provides [11]. The ZTIC is a small USB-connect smartcard reader with a small display and allows user input by means of two buttons (for OK and Not OK) and a wheel that can be turned to input numbers. Unlike the Radboud Reader, the ZTIC has cryptographic capabilities and the keys and certificates to set up a secure SSL/TLS tunnel between the ZTIC and a remote web server. Every device has its own certificates for mutual authentication with the issuer.

One difference between the ZTIC and our proposal is that the ZTIC is a more complicated device, capable of storing keys and doing crypto. Unlike our solution, which is generic and can be used in conjunction with any compatible smartcard, the ZTIC needs to have the certificate for each service provider in order to communicate with it. Another difference, and possible advantage of the ZTIC, is that the common functionality to provide a secure tunnel is provided by the ZTIC, and need not be provided by each smartcard used with the Radboud Reader: using the ZTIC it is guaranteed to have a secure tunnel, using the Radboud Reader this still depends on the smartcard.

The FINREAD project proposed a standardised trusted card reader [5]. This idea never became a success, probably because the FINREAD card reader would be too expensive and complex; they were meant to be tamper-resistant and included a PKI support for controlling applications on them.

The AGSES card reader[7] does not use a USB connection, but receives data via a flickering barcode on the PC screen. There is no communication back from this reader to the PC, so the user has to manually type in the response again. We do not know if the data sent using the flickering bar code is checked for authenticity before it is displayed.

[6] See `http://www.zurich.ibm.com/ztic`. Originally, ZTIC stood for Zurich Trusted Information Channel.

[7] `http://www.agses.net`

Table 1. Comparison with existing readers

	Trusted display	Crypto	Secrets	Connected	Generic
EMV-CAP reader	✗	✓	✗	✗	✗
Gemalto's ABN-AMRO reader [1]	✗	✗	✓[a]	↔	✗
ZTIC	✓	✓	✓[b]	↔	✗
FINREAD	✓	✓	✓[b]	↔	✓
AGSES	?	✓	✓[b,c]	→	✗
Radboud Reader	✓	✗	✗	↔	✓

[a] Unknown functionality
[b] Secret key
[c] Uses fingerprints rather than PIN code

Nowadays, smartcards with integrated keyboard and display are also commercially available. These displays are however very limited, for example, the smartcard offered by NagraID[8] only contains 6 characters.

A solution without even a reader is mTAN. Here the user receives an SMS on his mobile phone with transaction details and a code to confirm it. Here the phone could be seen as a trusted display. However, as mobile phones become more and more complex, they are now becoming popular targets for malware and attacks just like PCs.

6 Conclusions

The number and importance of online transactions is rapidly growing, and protecting such transactions in the face of ever more sophisticated malware is a serious challenge. This is not only an issue in online banking, but also in e-government services, or indeed any online transactions where one would really want the online equivalent of a handwritten signature.

We have presented the design of a simple and generic device for securing online transactions, which protects against any malware on the PC, including Man-in-the-Browser attacks. The device provides a trusted communication channel between a user and any remote service provider, by means of a smartcard issued by that provider.

The essence of our solution, as illustrated in Fig. 2, is quite simple: namely, make sure that all communication with the user passes through the smartcard. We are not aware of any solutions that use this approach, even though conceptually it is quite simple. This solution allows a very simple device, which does not need any cryptographic capabilities or need not store any keys or other secrets.

Because there are no secrets in the Radboud Reader, e.g. in the form of secret keys or secret protocols, anyone can make one. This can be considered a disadvantage (an attacker could make or market fake devices) but also an advantage, as anyone can implement their own device. Note that there is little

[8] `http://www.nagraid.com`

incentive for manufacturers of smartcard readers to come up with solution like ours, where there is no intellectual property or secret in the device, thus allowing anyone to manufacture it and not having any risk of vendor lock-in.

As discussed in detail in Section 5, the Radboud Reader offers stronger security than many existing USB-connected smartcard readers with display and keyboard used for internet banking, except IBM's ZTIC, as these solutions do not offer provide a trusted display, i.e. they can not guarantee that what appears on the display is an authentic message from the remote server.

Unlike other solutions, our solution is completely generic. The functionality hardcoded in the reader only provides some basic building blocks and the smartcard is in charge of using these to build the scenario that some service requires. So the same device can be used by different smartcards for different purposes. As the number of online services that require a high-security solution to secure transactions increases, investing in a single reader that can be used for all of them may prove a practical and economical solution. It may then also become an option to go for a slightly more expensive device, with a larger display; one could even imagine an e-reader for digitally signing electronic documents that operates in the same way as the Radboud Reader, though the limited bandwidth of communication with the smartcard could then become a bottleneck.

References

1. Blom, A., de Koning Gans, G., Poll, E., de Ruiter, J., Verdult, R.: Designed to Fail: A USB-Connected Reader for Online Banking. In: Jøsang, A., Carlsson, B. (eds.) NordSec 2012. LNCS, vol. 7617, pp. 1–16. Springer, Heidelberg (2012)
2. Dolev, D., Yao, A.: On the security of public key protocols. IEEE Transactions on Information Theory 29, 198–208 (1983)
3. Drimer, S., Murdoch, S.J., Anderson, R.: Optimised to Fail: Card Readers for Online Banking. In: Dingledine, R., Golle, P. (eds.) FC 2009. LNCS, vol. 5628, pp. 184–200. Springer, Heidelberg (2009)
4. EMVCo. EMV– Integrated Circuit Card Specifications for Payment Systems, Book 1-4 (2008), `http://emvco.com`
5. CEN Workshop Agreement (CWA) 14174: Financial transactional IC card reader (FINREAD) (2004)
6. Global Platform Organization. Card Specification, Version 2.2 (March 2006), `http://www.globalplaform.org`
7. Gullberg, P.: Method and device for creating a digital signature. European Patent Application EP 2 166 483 A1 (2010) (filed September 17, 2008, published March 24, 2010)
8. International Civil Aviation Organization. Machine Readable Travel Documents – Part 3-2, 3rd edn. (2008)
9. ISO/IEC, ISO/IEC 7816: Identification cards – Integrated circuit cards
10. Szikora, J.-P., Teuwen, P.: Banques en ligne: à la découverte d'EMV-CAP. MISC (Multi-System & Internet Security Cookbook) 56, 50–62 (2011)
11. Weigold, T., Kramp, T., Hermann, R., Höring, F., Xia, C., Baentsch, M.: The Zurich Trusted Information Channel – An Efficient Defence Against Man-in-the-Middle and Malicious Software Attacks. In: Lipp, P., Sadeghi, A.-R., Koch, K.-M. (eds.) Trust 2008. LNCS, vol. 4968, pp. 75–91. Springer, Heidelberg (2008)

Extending EMV Payment Smart Cards with Biometric On-Card Verification

Olaf Henniger and Dimitar Nikolov

Fraunhofer Institute for Computer Graphics Research IGD
Fraunhoferstr. 5, D-64283 Darmstadt, Germany
olaf.henniger@igd.fraunhofer.de, dsnikolov@gmx.de

Abstract. Nowadays, many bank cards are smart cards (i.e. integrated-circuit cards) based on the EMV specifications for payment systems. This paper specifies how biometric on-card verification can be integrated into EMV debit and credit cards in a backwards-compatible way. The biometric verification does not change the EMV transaction flow outside the cardholder-verification step. The proposed payment system has been prototyped using Java cards and an applet for handwritten signature on-card verification.

1 Motivation

Debit and credit cards used to be magnetic-stripe cards. Today, however, a growing number are smart cards equipped with a microprocessor chip in addition to the magnetic stripe. This significantly improves the capabilities for authenticating the card and the cardholder and enables new protocols for securing payment processes. In the absence of an online connection to the banking network, the card is able to represent the card-issuing bank and to authorise payments on its behalf.

For confirming a debit transaction the cardholder must enter a secret personal identification number (PIN). Unlike magnetic-stripe card based debit transactions, not every smart-card based debit transaction requires a paid-for online connection. The PIN entered may be verified offline against the PIN stored in the chip and, in case of a match, the card may authorise the payment. An online account validation and online transaction authorisation are carried out only under certain conditions determined by the card issuer, e.g. if the credit limit stored on the card has been used up or if more time than permitted has passed since the last online connection.

Some payment methods require the cardholder to confirm the payment with a handwritten signature on a slip of paper (e.g. credit-card payment and a form of direct debit transaction popular in Germany called "elektronisches Lastschriftverfahren"). These payment methods do not take advantage of the opportunities that the microprocessor on card offers for raising the confidence in cardholder authentication. The cashier may only visually compare the image of a given signature with the signature image on the back of the card. The handwritten signature dynamics could be verified inside the smart-card chip just as the PIN is verified inside the chip in case of offline PIN verification. This would make the comparison more objective and improve security provided that the required levels of attack resistance and usability are achieved.

S. Fischer-Hübner et al.(Eds.): IDMAN 2013, IFIP AICT 396, pp. 121–130, 2013.

The authorisation of payment transactions is not the only field of application of biometric on-card verification. Biometric on-card verification has been proposed before for protecting other smart card functions access to which should be restricted to the legitimate cardholder [1]. Instead of handwritten signatures also other biometric characteristics could be compared on bank cards in order to improve the binding of payments to the cardholder. However, the major advantage of handwritten signatures over other biometric characteristics is that people are used to presenting their signatures and that signatures are evidence of deliberate decisions.

The EMV specifications [2] (which are named after the organisations Europay, Mastercard, and Visa, who created the first version) specify, based on [3], requirements and building blocks for smart-card based payment systems. The EMV specifications are the basis for several EMV-compliant payment systems. Section 2 reviews how EMV transactions work. International standards [4,5] specify several approaches how to achieve personal verification through biometric methods, but has not been integrated into the EMV specifications yet. The challenge is to extend the EMV specifications in a way that is in compliance with the requirements and side conditions imposed by [3,4]. The main contribution of this work is to specify how biometric on-card verification can be integrated into EMV payment smart cards in a backwards-compatible way. Section 3 extends the cardholder-verification process of the EMV specifications with biometric on-card verification of the cardholder. Figure 1 illustrates the sequence of steps to be carried out for offline EMV transactions on a point-of-sale (POS) terminal equipped with a biometric capture device. Section 4 discusses security and usability issues. Our prototype of the proposed payment system is described in Section 5. Section 6 summarises the results and gives an outlook.

2 EMV Transactions

2.1 Overview

An EMV transaction requires that both, the debit card of the customer who wishes to pay for some goods and the POS terminal, conform to the EMV specifications. An EMV transaction begins with the insertion of the card into the terminal. Afterwards, the terminal determines what applications are installed on the card and the cardholder selects a payment method. The end result of the process is a cryptogram issued by the card. This cryptogram is a transaction certificate (TC) in case of an accepted and authorised transaction or an application authentication cryptogram (AAC) if the transaction is rejected for some reason.

In [6] an EMV transaction is described as consisting of the following steps:

Initiate Application Processing: This step
- Informs the smart card that the processing of a new transaction is beginning,
- Exchanges transaction-related information between terminal and smart card,
- Determines whether the transaction can be processed.

Read Application Data: The terminal reads the data from the card that are needed to process the transaction.

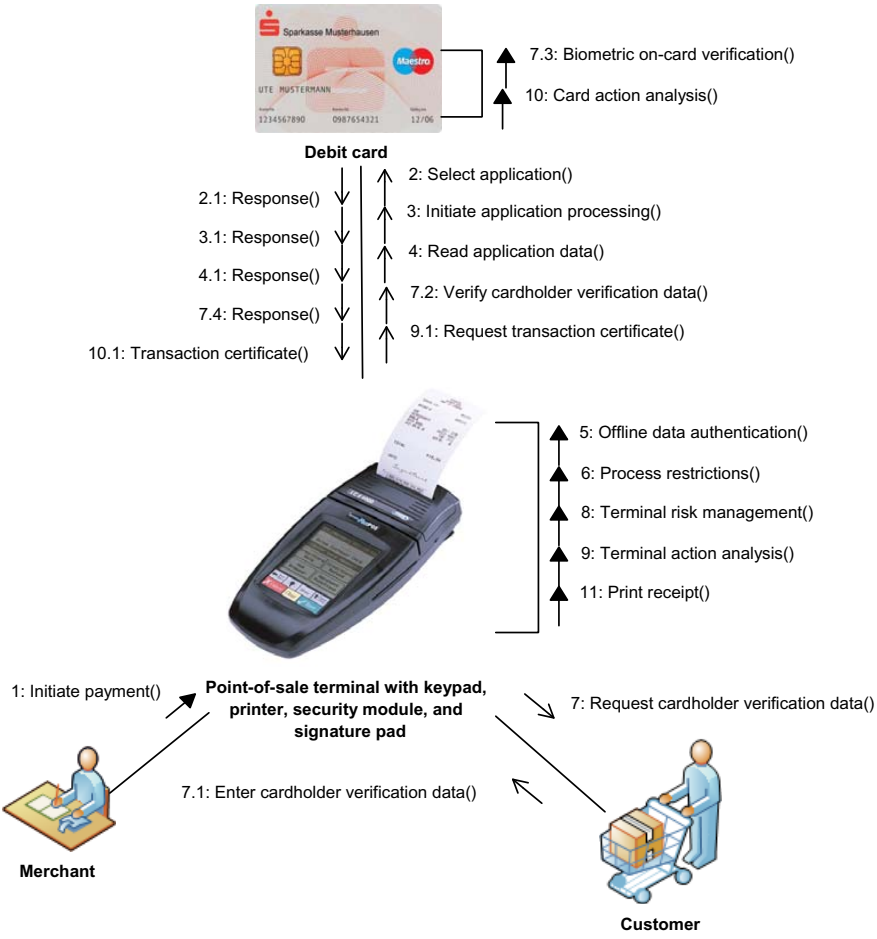


Fig. 1. Communication diagram for offline EMV transactions with biometric on-card verification

Offline Data Authentication: This step is performed if both the terminal and the card support offline checking of the validity and integrity of the data stored on the chip. Different methods based on public/private key pairs are defined for this purpose:

- The static data authentication method verifies the digital signatures of the card-issuing bank on data stored in the card;
- The dynamic data authentication methods preclude counterfeiting of EMV cards by not only verifying the digital signatures on static data, but also verifying signatures created by the card on challenges received from the terminal.

Processing Restrictions: The terminal determines the degree of compatibility of the application in the terminal with the application in the smart card and makes necessary adjustments, possibly rejecting the transaction. It is possible to restrict the application geographically or only to certain types of transactions.

Cardholder Verification: This step is performed to ensure that the person presenting the smart card is the person to whom the card was issued. It is up to the card issuer to choose which cardholder verification methods (CVMs) to apply under what conditions. [6] provides for the following CVMs or combinations thereof:
- Offline PIN,
- Online PIN, and
- Handwritten signature on paper.

Section 3 of this paper proposes to add biometric on-card verification to this list.

Terminal Risk Management: This is the portion of risk management performed by the POS terminal to protect the acquiring and card-issuing banks from fraud. It provides positive issuer authorisation for high-value transactions and ensures going online regularly to protect against threats that are undetectable in an offline environment.

Terminal Action Analysis: The terminal takes a first decision as to whether the transaction should be approved offline, declined offline, or transmitted online.
- If the decision is to proceed offline, the terminal sends a GENERATE AC command asking the smart card to return a TC. However, the smart card may return an authorisation request cryptogram (ARQC) or an AAC instead of a TC as a result of card action analysis.
- If the decision is to reject the transaction, the terminal sends a GENERATE AC command asking the smart card for an AAC.
- If the decision is to go online, the terminal sends a GENERATE AC command asking the smart card for an ARQC. The ARQC is a cryptogram generated by the card from transaction data using an issuer key stored in the card and known in the issuer authorisation system.

Card Action Analysis: The smart card performs risk management on behalf of the card-issuing bank to protect the card-issuing bank from fraud or excessive credit risk. Details of the risk management algorithms are specific to the card issuer and outside the scope of the EMV specifications. The smart card may decide to complete a transaction online or offline or to reject the transaction. The smart card may also decide that an advice message should be sent to the card issuer to inform the issuer of an exceptional condition.

Online/offline Decision: Depending on the card's answer to the previous GENERATE AC command, the terminal decides whether to continue the processing online.

Online Processing: This step may be performed to ensure that the card issuer can review and authorise or reject transactions that are outside acceptable limits of risk defined by the card issuer, the payment system, or the acquirer. The ARQC generated by the card is sent to the card issuer in an authorisation request message. The issuer uses his key to authenticate the ARQC and thereby to authenticate the card. The authorisation response message sent from the issuer authorisation system to the terminal may contain issuer authentication data. This is a cryptogram generated using an issuer key from selected data included in the authorisation response or already known to the card. The terminal forwards the issuer authentication data to the smart card in an EXTERNAL AUTHENTICATE command or a second GENERATE AC command. The smart card may use the issuer authentication data to authenticate that the response message originated from the issuer.

Issuer-to-card Script Processing: A card issuer may provide command scripts to be delivered to the card by the terminal to perform functions that are important for the continued functioning of the application in the card, e.g. unblocking an offline PIN.

Completion: This step closes the processing of a transaction. The smart card indicates willingness to complete transaction processing by returning either a TC or an AAC to either the first or second GENERATE AC command issued by the terminal. If the terminal decides to go online, completion is done when the second GENERATE AC command is issued.

2.2 Details of Cardholder Verification

The terminal uses a CVM list read from the card to determine the cardholder verification method to be performed. A CVM list is a composite data object consisting of:

- X value,
- Y value,
- Cardholder verification (CV) rules list.

The X and Y values are threshold amounts in the application currency of the card application that can be used by the CV rules. The CV rules list is a variable-length list of two-byte data elements. Each CV rule describes a CVM and the conditions under which that CVM is to be applied.

The terminal processes the card's CV rules list entry by entry attempting to perform each applicable CVM until either

- A CVM is performed successfully,
- A CVM required to be performed successfully is performed unsuccessfully,
- The CV rule requires the cardholder verification to fail under the given conditions,
- Or the CV rules list is exhausted.

If a CVM is performed successfully, EMV transaction processing continues with the next step. Otherwise, the cardholder verification is unsuccessful and transaction processing is terminated.

3 Proposed Extensions for Biometric Cardholder Verification

The proposed extensions to the EMV specifications concern only the cardholder verification. From among the options for personal verification through biometric methods described in [4], for reasons of compatibility we have chosen that option that imposes the least changes to the EMV specifications. Given the fact that EMV payment systems are already widely used, the more extensive the changes, the less acceptable they would be. The main ideas of the proposed changes are the following:

1. Extend the commands used for PIN processing (CHANGE REFERENCE DATA, VERIFY, and GET DATA) to support biometric on-card verification by introducing new allowed values for the command parameters P1/P2 and introducing command chaining [7] for CHANGE REFERENCE DATA and VERIFY.

2. Extend the terminal verification result (TVR) and the associated terminal action code (TAC) and issuer action code (IAC) data elements in the data elements dictionary of [6] by one byte to hold information about biometric cardholder verification.
3. Extend the data elements dictionary of [6] by adding the following data elements to support biometrics:
   Biometric Information Template (BIT): The BIT contains information about the mode and format of biometric data and possibly further information [4,8]. Prior to biometric cardholder verification, the BIT for on-card verification is retrieved in order to inform the terminal application about properties of the biometric on-card verification method. The terminal application uses the format owner and format type identifiers from the BIT for identifying the required format of the biometric probe. The comparison algorithm parameters in the BIT provide special parameters of the biometric on-card verification algorithm, e.g. the maximum number of minutiae expected in a biometric probe in case of finger minutiae data or the maximum number of sample points in case of handwritten signature time series data. If the card holds several biometric reference data objects, then a group BIT containing several BITs is used to describe the kind of biometric data to be sent to the card.
   Biometric Reference: The biometric reference is one or more stored biometric samples, biometric templates or biometric models attributed to the cardholder and used for biometric comparison.
   Biometric Retry Counter: The offline PIN in the EMV specifications is associated with a retry counter indicating the number of remaining allowed PIN verification attempts. Its initial value indicates the supported maximum number of PIN verification attempts. A similar counter is needed for every biometric verification method as well.
4. Extend the CVM processing to include biometric cardholder verification consisting of the following steps:
   (a) Read the biometric retry counter and if the CVM is not blocked, continue.
   (b) Retrieve the BIT using a GET DATA command and if it is understandable and supported, continue.
   (c) If the biometric capture device is operational, continue.
   (d) Attempt to capture the biometric data and if successful, format them according to the BIT, and send them to the card within a VERIFY command and get the result of the on-card verification.
   (e) Continue depending on the obtained result.

Figure 2 shows the biometric CVM processing flow in the POS terminal. The diagram is built following the example of the PIN CVM processing flow diagram of [6].

4 Security and Usability Issues

To cut costs, retailers like using direct debit transactions without online authorisation, only requiring a handwritten signature of the customer on a slip of paper. However, such payments do not establish a guarantee of payment. In case of fraud, the cardholder may deny the payment and initiate a chargeback. In general, only EMV payments with

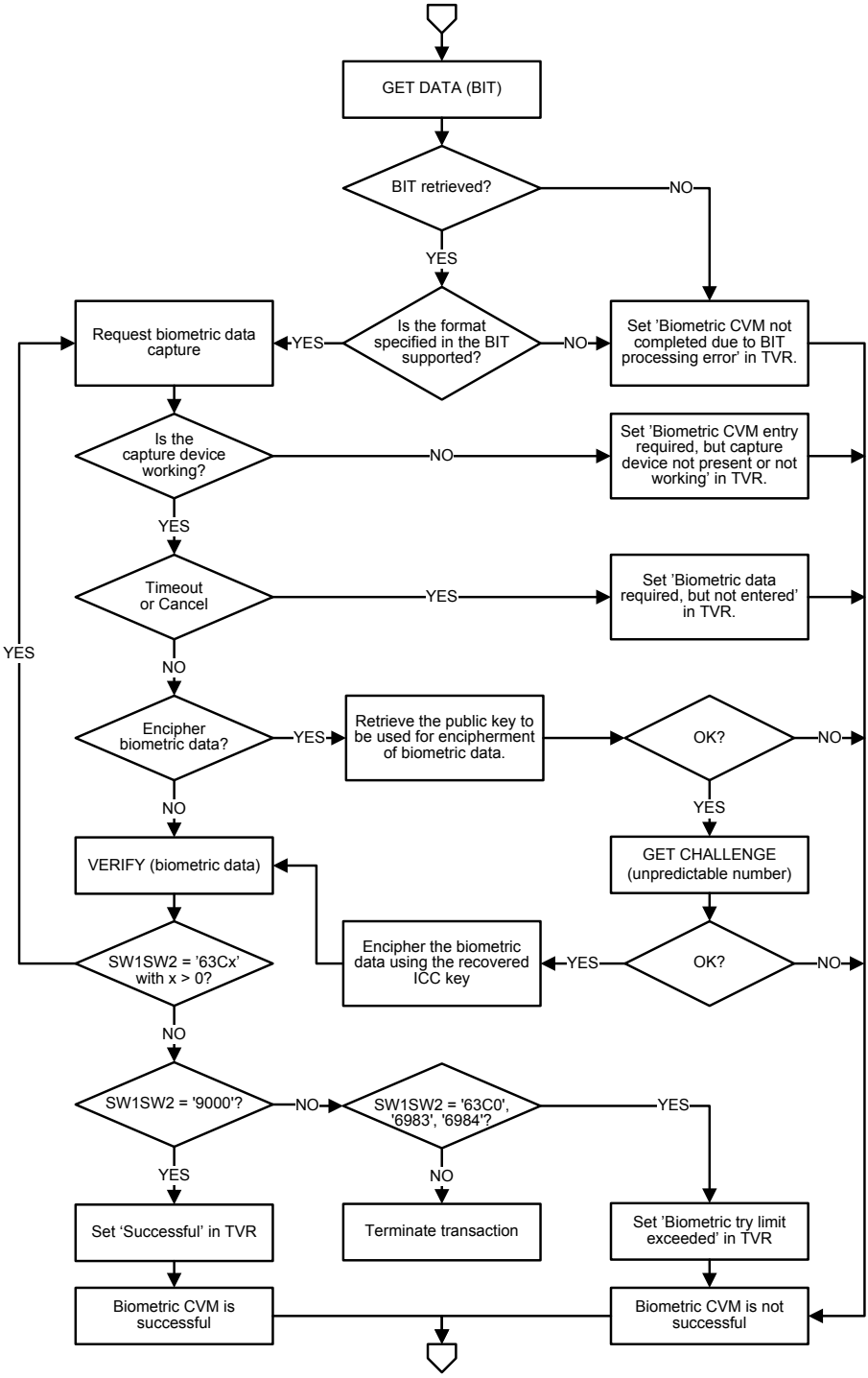


Fig. 2. Biometric CVM processing flow in the POS terminal

PIN entry are considered undeniable (even though [9] demonstrated that a particular implementation of the EMV specifications, "Chip and PIN", is vulnerable to a man-in-the-middle attack).

Biometric on-card verification will make fraud more difficult in case of payments without PIN entry, not least because EMV transactions include verifying the authenticity of the card. However, biometric comparisons are susceptible to false accept and false reject errors. As before, the cardholder should get the debited amount refunded when objecting within a time limit in case of a false accept error. If the card errs, a cardholder whose card has been stolen cannot be blamed. The biometric reference should be of high quality and hard to be forged and the verification threshold should be set such that false acceptances (i.e. misuse of the bank card by means of biometric on-card verification) are very rare.

In case of false reject errors, the card should offer yet another CVM (e.g. offline PIN verification) as a fallback. Therefore, the last entry in the CV rules list should not refer to the biometric on-card verification. A fallback is also necessary because not every payment terminal will be equipped with a signature pad.

5 Prototype

5.1 Java-Card Applet

We have created a prototype of an EMV payment application incorporating the proposed extensions for biometric cardholder verification on a Java card (JCOP 31 v2.2 or similar). The prototype uses the handwritten signature on-card verification applet presented in [10] with improvements discussed in [11]. The applet provides the Biometric Application Programming Interface (API) for Java cards [12].

5.2 Enrolment Terminal

We have built a demonstrator for the personalisation of the card and handwritten signature enrolment based on a PC connected with a card reader and a graphics tablet.

During enrolment the user is required to sign five times. The five captured signatures are sent to the card in chained CHANGE REFERENCE DATA commands. After all signatures have been received by the card, each is compared with each of the others. They all are required to be similar enough to each other (satisfy a predefined threshold) for the process to continue. The results computed for the comparisons of each signature are then compared to the results of the other signatures. The signature with the least distance to the others is stored as biometric reference. The worst distances between the signatures are used to compute a decision threshold that is to be used in verification. Storing a signature of high quality as reference and setting a proper threshold for allowed signature variety are of crucial importance for ensuring low errors rates.

During personalization apart from the signature, personal data, cryptographic data, the CVM List and the BIT are stored on the card. After the phase has been completed successfully, the applet is marked as ready to use in transactions and all functionality for storing or changing data that is expected to be written only once is disabled.

5.3 POS Terminal

To drive the Java-card applet and to show its capabilities, we have also built a terminal demonstrator simulating an offline POS terminal where an EMV transaction occurs and a tool for cryptogram verification. The demonstrator keeps a detailed log of the APDUs exchanged between terminal and card.

6 Summary and Outlook

This paper describes a solution for extending the EMV specifications [6] to include biometric on-card verification methods. This allows deploying biometric on-card verification on debit and credit cards. If sufficiently resistant against direct and indirect attacks and if easy to use, biometric cardholder verification methods can strengthen the binding to the legitimate cardholder. The extensions proposed are fully in compliance with the requirements and side conditions imposed by [3,4].

In order to demonstrate the feasibility of the design, the proposed payment system has been prototyped on Java cards [13] using a Java-card applet for online signature on-card verification [10]. Before this can be applied in real bank cards, it is advisable to thoroughly evaluate the security of the biometric on-card verification product based on officially recognized criteria like the Common Criteria for IT security evaluation [14].

References

1. Struif, B.: Use of Biometrics for User Verification in Electronic Signature Smartcards. In: Attali, S., Jensen, T. (eds.) E-smart 2001. LNCS, vol. 2140, pp. 220–227. Springer, Heidelberg (2001)
2. EMV integrated circuit card specifications for payment systems, Version 4.2 (June 2008)
3. Identification cards – Integrated circuit cards, International Standard ISO/IEC 7816
4. Identification cards – Integrated circuit cards – Part 11: Personal verification through biometric methods, International Standard ISO/IEC 7816-11, 1st edn. (2004)
5. Information technology – Identification cards – On-card biometric comparison, International Standard ISO/IEC 24787 (2010)
6. EMV integrated circuit card specifications for payment systems – Book 3: Application specification, Version 4.2 (June 2008)
7. Identification cards – Integrated circuit cards – Part 4: Organization, security and commands for interchange, International Standard ISO/IEC 7816-4, 2nd edn. (2005)
8. Information technology – Common biometric exchange formats framework – Part 3: Patron format specifications, International Standard ISO/IEC 19785-3 (2007)
9. Murdoch, S.J., Drimer, S., Anderson, R., Bond, M.: Chip and PIN is broken. In: 2010 IEEE Symposium on Security and Privacy (2010)
10. Henniger, O., Franke, K.: Biometric User Authentication on Smart Cards by Means of Handwritten Signatures. In: Zhang, D., Jain, A.K. (eds.) ICBA 2004. LNCS, vol. 3072, pp. 547–554. Springer, Heidelberg (2004)

11. Henniger, O., Müller, S.: Handwritten Signature On-Card Matching Performance Testing. In: Fierrez, J., Ortega-Garcia, J., Esposito, A., Drygajlo, A., Faundez-Zanuy, M. (eds.) BioID MultiComm2009. LNCS, vol. 5707, pp. 268–275. Springer, Heidelberg (2009)
12. Biometric Application Programming Interface (API) for Java Card, NIST/Biometric Consortium Biometric Interoperability, Assurance, and Performance Working Group, version 1.1 (August 2002)
13. Nikolov, D.: Debit and credit cards with handwritten signature on-card matching, Master's thesis, Technische Universität Darmstadt (2012)
14. Information technology – Security techniques – Evaluation criteria for IT security, International Standard ISO/IEC 15408

Dynamic Identity Federation Using Security Assertion Markup Language (SAML)

Md. Sadek Ferdous and Ron Poet

School of Computing Science, University of Glasgow, Glasgow, G12 8QQ, Scotland
m.ferdous.1@research.gla.ac.uk, ron.poet@glasgow.ac.uk

Abstract. Security Assertion Markup Language (SAML, in short) is one of the most widely used technologies to enable Identity Federation among organisations from different trust domains. Despite its several advantages, one of the key disadvantages of SAML is the mechanism by which an identity federation is established. This mechanism lacks flexibility to create a federation in a dynamic fashion to enable service provisioning (or de-provisioning) in real time. Several different mechanisms to rectify this problem have been proposed. However, most of them require a more elaborate change at the core of the SAML. In this paper we present a simple approach based on an already drafted SAML Profile which requires no change of the SAML, rather it depends on the implementation of SAML. It will allow users to create federations using SAML between two prior unknown organisations in a dynamic fashion. Implicit in each identity federation is the issue of trust. Therefore, we also analyse in detail the trust issues of dynamic federations. Finally, we discuss our implemented proof of concept to elaborate the practicality of our approach.

Keywords: Identity Management, Federated Identity Management, Identity Federation, Security Assertion Markup Language (SAML), Trust.

1 Introduction

With the continuous evolution of the online landscape, the number of web-enabled services as well as the user-base expanded rapidly, more and more digital identifiers and credentials were issued to allow users to access those services, and soon their management became challenging, both for organisations and for users. Identity Management (IdM, in short) was introduced by the industry to facilitate online management of user identities, which resulted in various different Identity Management Systems (IMS or IdMS). Formally, Identity Management consists of technologies and policies for representing and recognising entities using digital identifiers within a specific context [14]. There are different types of Identity Management Models such as the SILO Model, Common Identity Domain Model, Federated Model, etc. [14]. Among these models, Federated Identity Management (FIM, in short) has gained considerable attention. It is based on the concept of Identity Federation (also known as Federated Identities or Federation

S. Fischer-Hübner et al.(Eds.): IDMAN 2013, IFIP AICT 396, pp. 131–146, 2013.

of Identities). In the ITU-T X.1250 recommendation, a federation is defined simply as "An association of users, service providers and identity providers" [13]. In other words, a federation with respect to the Identity Management is a business model in which a group of two or more trusted parties legally bind themselves with a business and technical contract [9,17]. It allows a user to access restricted resources seamlessly and securely from other partners from different Identity Domains. An identity domain is the virtual boundary, context or environment in which a digital identifier is valid [17]. FIM offers a good number of advantages to different stakeholders such as separation of duties among different organisations, scalability, improved security and privacy, SSO for users, etc. [5,9]. Single Sign On (SSO) is the capability that allows users to log in one system and then access other related but autonomous systems without further logins. It alleviates the need to log in to every time a user needs to access those related systems. Using a FIM System, users can take advantage of SSO and thus authenticate themselves in one identity domain and receive personalised services across multiple domains without any further authentication. A FIM System has three different actors:*Identity Provider (IdP)* - responsible for managing digital identities of users and providing identity related services to different Service Providers, *Service Provider (SP)* - responsible for providing web-enabled services to the users and *User (Client)* - receives services from an SP.

The issue of trust is a fundamental concept in FIM as different participating organisations need to trust each other inside the federation to a sufficient level to allow them to exchange user information and trust that information. A detailed analysis of trust requirements for FIM can be found in [15]. To summarise them, the SP needs to trust the IdP to authenticate the user using appropriate security mechanisms and release attributes to the SP as per the contractual agreement. Similarly, the IdP has to trust that the SP will not abuse the released attributes and use them only for the stated purpose as per the agreement. Based on this trust level, users will be granted access to a service or rejected. Thus, organisations inside a federation are said to form the so-called Circle of Trust (CoT).

There are several core technologies available that allow the creation of federations such as Security Assertion Markup Language (SAML) [19], OpenID [6], WS-Federation [22], etc. There are several FIM Systems such as Shibboleth [3], OpenID [6], Microsoft CardSpace [2] that respectively utilise these technologies. Among all of these technologies, SAML has been the most widely used technology for deploying federation that requires strong trust assumptions as well as maintains good security and privacy properties. SAML is an XML-based standard for exchanging authentication and authorisation information between different autonomous security domains. It is based on the request/response protocol in which one party (generally SPs) requests for particular identity information about a user and the other party (IdPs) then responds with the information.

Trust in SAML is established by exchanging metadata of the IdP and the SP and then storing them at the appropriate repositories which helps each party to build up the so-called Trust Anchor List (TAL). This exchange takes place in

out-of-bound fashion after a technical contract between the IdP and the SP is signed and has to be done before any interaction takes place between the said IdP and the SP. A metadata is an XML file in a specified format that plays the central role in SAML. It contains several pieces of information such as entity descriptor (id for each party), service endpoints (the locations of the appropriate endpoints for IdPs where the request will be sent to and for SPs where the response will be consumed), certificate(s) to be used for signing and encryption, expiration time of metadata, contact information, etc. and serves three purposes. Firstly, it allows each organisation to discover the required endpoint of another organisation for sending a SAML request/response. Secondly, the embedded certificate can be used by the SP to verify the authenticity of the SAML Assertion. Thirdly, a metadata behaves like an anchor of trust for each party. During the IdP discovery service at the SP, the list only contains those IdPs whose metadata can be found in its meta repositories (in other words in the TAL) and thus considered trusted. Similarly, IdP will respond only to those requests that are initiated from an SP whose metadata can be found in its metadata repositories (in its TAL). Exchanging and maintain the repositories of metadata and thus managing trust becomes extremely difficult as the number of the IdPs and SPs grows and is a well-known problem of SAML [8]. In addition, pre-configuring trust before any interaction hinders the establishment of a federation between two prior unknown parties in a dynamic fashion. Allowing federations to be created dynamically would open up the door for new business scenarios and novel web-enabled services. There have been several proposals and drafts to handle this situation. Most of these works would either require a considerable amount of change of the SAML protocol or only provide mechanisms for creating federations in a semi-automated fashion. Also the trust issues involved while creating federations in a semi-automated fashion are not thoroughly examined. In this paper, we build on some of the existing works and illustrate mechanisms that can be used to establish federations in a fully automatic fashion. Furthermore we thoroughly explore the complex trust issues involved during such scenarios. The contributions of this paper are:

- We have found that the idea of dynamic federation is still ill-defined. Therefore, we have taken the initiative to define formally some key concepts behind a dynamic federation.
- We extend some existing works so that federations can be created in a fully dynamic fashion.
- We explore the trust issues in such scenarios.
- Finally, we describe our developed proof of concept to illustrate the applicability of our proposal.

With this introduction, this paper is organised as follows. We discuss some existing works on dynamic federation and analyse their strengths and weaknesses in Section 2. Then, we define some key concepts of dynamic federation and explore the trust issues in Section 3. The developed proof of concept is discussed for

two different use-cases in Section 4. We present the strengths and weaknesses of our implementation as well highlight a few potential future works in Section 5. Finally, we conclude in Section 6.

2 Related Work

There have been several works to tackle the problem of scalability and dynamism of SAML. The most influential work, called Distributed Dynamic SAML, can be found in [12] where the authors prescribe that to trust any dynamically exchanged metadata, the metadata must be signed and the X.509 certificate that can be used to verify the signature must be included within the metadata. Assuming a trusted Certificate Authority (CA) issues the embedded certificate, each participating organisation will hold the root CA Certificates which can be used to validate the certificate chain in its TAL. Then, the trust in the metadata can be derived by just verifying the signature in the metadata using the embedded certificate with the traditional PKI. The result of their proposal is the formulation of a working draft of a novel SAML Profile called SAML Metadata Interoperability Profile which is a profile for distributed SAML metadata management system [20,21]. Based on the proposal and the working draft, an implementation of dynamic SAML has been developed by UNINETT (*https://www.uninett.no/*) in their SimpleSAMLphp project [18,4]. The SimpleSAMLphp is a native php-based implementation of the SAML protocol stack that allows a SAML IdP or SP to be deployed very quickly. The proposal and the working draft and the SimpleSAMLphp implementation would allow the establishment of federations more quickly than it would be possible previously. However, several crucial questions regarding trust assumptions at different parties have not been explored thoroughly. For example, each party (IdP and SP) validates the certificate of other parties using PKI to establish trust. However, is the established trust enough for any IdP to release sensitive user attributes to an SP which has been added dynamically since there may not be any legal contract between them? And also, since there may not be any legal binding, can the IdP trust that the SP would not abuse the released attribute in any way? Similarly, the SP will need to consider if it can trust any attributes that have been provided by a dynamically added IdP even though the SAML assertion containing those attributes are properly verified.

Furthermore, the SimpleSAMLphp implementation requires that the metadata of the IdP is already present at the SP so that the WAYF (Where Are You From) Service (and IdP discovery service) can display the list of the IdPs to the user. Once the user selects an IdP, a SAML authentication request will be sent to the IdP. If the IdP has the capability to add an SP dynamically (e.g. an IdP deployed with SimpleSAMLphp), it can retrieve the metadata of the SP dynamically and store it temporarily in case the entity ID of that SP is not found in its TAL. To make this possible, the SimpleSAMLphp requires that the entity ID of the SP has to be a URL from where the metadata can be fetched. In summary, the SimpleSAMLphp only allows IdPs to retrieve any SP metadata, not the other way around. We prefer to call it a semi-automatic federation where

the IdP has to be pre-configured at the SP and thus does not fully address the problem of dynamic federation.

There are some other works that also provide proposals for dynamic federations. In [7], the authors propose a SAML extension to accommodate reputation data in XML format. According to their proposal, trust has to be negotiated based on that reputation data before any entity can join the federation. Each entity will maintain a dynamic list called Dynamic Trust List (DTL), instead of the static TAL, which will contain the list of joined entities in the same federation with their reputation data and will be updated dynamically as the federation evolves. To realise their proposal, a novel exchange protocol has to be developed to request and response reputation data. The authors in [25] proposed a dynamic federation architecture, called DFed, based on SAML and Automatic Trust Negotiation (ATN) to establish trust between participating parties in run time. Each DFed party, known as Dynamic Federation Service Providers (DFSP), can act as an IdP and SP. Each DFed consists of different components such as Gate Keeper (GK), Directory Services, Trusted DFSP Repository, SAML Agent and ATN Agent. GK is responsible for the SSO Protocol, Directory Service is responsible for storing attributes and policies, DFSP repository stores the information of federated SPs, SAML Agent is responsible for carrying out the SAML Protocol and ATN agent is responsible for ATN protocol and trust negotiation. All of them function together to realise the Dynamic Federation protocol. It is also clear that DFed also requires that SAML are changed extensively to accommodate the DFed protocols. There is another solution proposed in [24] where the authors propose calculating trust values based on the modified Dijkstra algorithm and to calculate a distributed reputation based on the PageRank algorithm from Google and use the trust and reputation value to create dynamic federations. And like before, this also requires a major change of the SAML Protocol.

Our focus in this work is not to change anything in the core SAML Protocol and therefore we have based our work on the Dynamic SAML. We will extend the existing SimpleSAMLphp implementation so that a federation can be established fully dynamically considering different trust issues.

3 Dynamic Federation

All previously mentioned works have used the term *Dynamic Federation* literally without defining them formally. The lack of any formal definition for Dynamic Federation means that there are scopes for misunderstanding and multiple interpretations. Before we proceed any further, it is therefore essential to define the term *Dynamic Federation* formally which is presented below:

Definition 1. *A Dynamic Federation with respect to the Identity Management is a business model in which a group of two or more previously unknown parties federate together dynamically without any prior business and technical contract with the aim to allow users to access services under certain conditions.*

This definition is a stark contrast with the traditional definition of the identity federation based on SAML in which there must be a legally binding technical

and business contract between participating organisations before they can join any federation. The primary advantage here is the ability to join the federation instantly in real time. However, the lack of any legally binding contract means that organisations must consider that there might be negative consequences involved since no party is bound to behave as it should and therefore take proper precautions. This leads us to the topic of trust which is explored below.

3.1 Trust Issues

According to our proposed definition, participating organisations may not trust each other entirely since they are previously unknown and there is no contract to make them accountable in case there is any unintended incident. The IdPs may not want to release a few sensitive attributes to the SP that has been added dynamically and the SPs may not trust all attributes released by the IdP that has been added dynamically.

Such trust issues have not been considered while drafting the Dynamic SAML which allows any SP to communicate with the IdP and any IdP to communicate with the SP without any sort of verification. Remember that, technically speaking, joining a federation in the Dynamic SAML will just require the parties to exchange and store the respective metadata with each other. The SimpleSAMLphp implementation based on the Dynamic SAML only allows an SP to join with an IdP since it requires the pre-configuration of the IdP in the SP TAL to allow any user to choose that IdP. However, to harness the true potential of dynamic federation, we need to ensure that both parties can be added dynamically. Moreover, the IdP in SimpleSAMLphp does not distinguish between statically and dynamically added SPs. This allows the IdP to release the same level of (sensitive) attributes to both types of SPs. In summary, we have two requirements to fulfil: i) Fully automate the joining procedure in a federation for both parties and ii) Ensure that some sort of trust is established in a dynamic federation. With these two goals in minds we introduce the notion of fully trusted, semi-trusted and untrusted entities.

Definition 2. ***Fully Trusted Entities***. *Fully trusted entities are the IdP and SP in the traditional SAML federation in which there is a legal contract between the IdP and the SP. They are so called since each IdP (or SP) inside a federation trusts any SP (or IdP) in the same federation to behave as intended and can be made accountable in case the other party behaves maliciously.*

Definition 3. ***Semi-trusted Entities***. *Semi-trusted entities are the SPs in a dynamic federation that have been added dynamically to an IdP inside the federation under* ***some conditions*** *without the presence of any contract between them and to whom any user(or users) of the IdP has(have) agreed to release a subset of her(their) attributes. They are so called since the user wants to release a subset of their attributes to these SPs inside the dynamic federation even though the IdP in the same federation may not fully trust such SPs to behave as intended. Therefore, such SPs might not be made accountable by the IdP in cases the they behave maliciously with the absence of any contract between them.*

Definition 4. ***Untrusted Entities****. Untrusted entities are the IdP and SP in a dynamic federation in which they have been added dynamically under* ***some conditions*** *without the presence of any contract between them. They are so called since each IdP (or SP) inside a dynamic federation may not trust at all any other dynamically added SP (or IdP) in the same federation to behave as intended.*

It is important to understand that a dynamic federation may accommodate as many fully trusted entities as possible. As such, a dynamic federation is an extension of the traditional federation.

Now, the term "some conditions" in the definition of the semi-trusted and untrusted entities require further explanations. It can be the combinations of several different conditions by which an SP can be added dynamically to the IdP and vice versa, the conditions for establishing individual trust with each other in such a federation, the condition by which attributes are released to a semi-trusted SP and the condition by which an SP treats attributes of a user from an untrusted IdP. Semi-trusted and untrusted entities of different dynamic federations should have different sets of conditions suitable for their business models and service provisioning scenarios. Here, we present a set of conditions that we have assumed for developing our proof of concept of dynamic federation using SAML.

- Only a valid user of an IdP is allowed to add an SP to that IdP dynamically. This is to ensure that only those SPs that the users want to access for service provisioning are added in a dynamic federation. This is missing in the current implementation of SimpleSAMLphp.
- Once the SP is added to the IdP, the SP must add the IdP to its TAL to ensure that the user can select the IdP next time. This nullifies the need to pre-configure the IdP in the SP.
- Dynamically added SPs must be tagged as untrusted entities in the IdP at the initial stage. Only when a user, after being authenticated at the IdP, has agreed to release a subset of her attributes to the SP, the SP should be re-tagged as a semi-trusted entity.
- A dynamically added IdP should always be tagged as an untrusted entity for the SP.
- IdPs should ensure that it does not release any attributes to an untrusted entity.
- IdPs should ensure that some crucial and sensitive attributes are not released to any semi-trusted entity since there is no guarantee that such attributes will be handled as intended. Therefore, it should allow their administrators to configure what attributes should be released to a semi-trusted entity.
- It is up to the discretion of each SP how they want to treat released attributes from an untrusted IdP. They could use the NIST LoA (Level of Assurance or Level of Authentication) guidance of 1 to 4 where Level 1 conforms to the lowest assurance and 4 conforms to the highest assurance [16]. Usually, the LoA level comes from the IdP and is embedded inside a SAML assertion to provide the level of assurance for a certain authentication mechanism at the

IdP. However, the SP should consider implicitly that LoA 1 is the maximum that can ever accompany the SAML authentication and attribute statements from any untrusted IdP. Since, how the released attributes will be handled depends on the individual SP, it can vary from one SP to another even inside the same federation.

To summarise, we propose that entities have to be federated in a fully automatic fashion without any human intervention to harness the full potential of dynamic federation and while doing so all entities must consider trust issues involved. The conditions outlined before are one of the many ways to fulfil our proposals.

4 Proof of Concept

In this section we will discuss the proof of concept that we have developed to illustrate the applicability of our proposals. We have used SimpleSAMLphp and modified it to meet our requirements. The following subsections will consider two different use-cases: i) IdP-SP Scenario and ii) IdP-IdP-SP Scenario.

4.1 IdP-SP Scenario

The architectural setup for this scenario is that there are one IdP and one SP deployed with the modified version of the SimpleSAMLphp. At the beginning, the IdP and SP are not part of a common federation (they individually may be part of separate federations) and they have no prior knowledge of the other party whatsoever. The IdP is configured to use a MySQL database at its end to store user attributes including username and password in a table called users. In addition, the IdP uses two new tables called *semitrusted and untrusted* to store the entity IDs of semi-trusted and untrusted SPs respectively. Similarly, the SP is also configured to use a MySQL database at its end where it uses a new table called *untrusted* to store the entity IDs of IdPs that have been federated dynamically.

In addition, we also need to provide a mechanism by which an admin of the IdP can configure which attributes to release to a semi-trusted SP. We have opted in for a configuration parameter called *semitrusted.sp* which can be added to the configuration file (called config.php) in the SimpleSAMLphp. A sample configuration parameter could be *'semitrusted.sp'=> array ('username', 'name', 'telephone', 'age', 'position', 'org')* which will configure the IdP to release only these attributes by excluding all other attributes such as *salarygrade & email* attributes, which the IdP normally releases to all trusted SPs but assumes to be too crucial and sensitive, to a semi-trusted SP. The admin can add as many or as less attributes as needed as per the requirements. This configuration parameter works like an attribute release policy. SimpleSAMLphp does not have the concept of an Attribute Release Policy, however, it has something similar called an Authentication Processing Filter (AuthProc) [1]. An AuthProc allows the system to do something extra once the user authentication is done. For example,

among other things, it can be used for filtering out attributes once the authentication is done. There are several authentication processing filters bundled with the SimpleSAMLphp implementation. One of them is the Consent module that is used to display the list of attributes to the user just before they are released. We have modified this module to allow the IdP to show only those attributes that can be found on the *semitrusted.sp* parameter. From the displayed list, the user can choose which attributes she wants to release to the SP.

With this setup, the protocol flow for this scenario is given below.

1. A user visits the SP for the first time to access one of its service. Since there is no security context (e.g. no cookie) of the user at the SP, the user needs to be authenticated at an IdP and therefore she is redirected to the WAYF service and a list with federated IdPs with the SP is shown.
2. Since the IdP and SP are not part of a common federation, the IdP list at the SP does not contain the IdP. However, since the SP supports (more precisely the SimpleSAMLphp that has been used to deploy the SP supports) our proposal of dynamic federation, it contains two additional text fields (Figure 1) which allow the user to enter the entity ID of her IdP and a code to ensure that only the valid users of the IdP have the ability of federating an SP with the IdP.

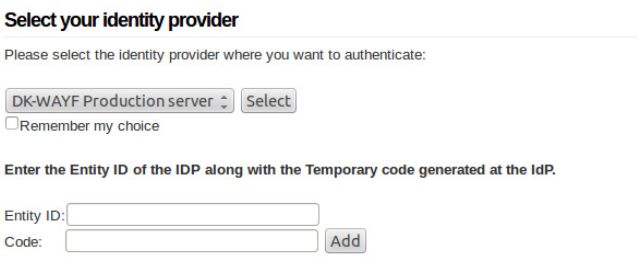


Fig. 1. Additional Text Fields at the WAYF

3. Since the user does not have the code, she logs in to her IdP and generates a code using the Generate button at the Generate IdP Code page. This page also checks the *semitrusted* table to see if there is any dynamically added SP. Since there is none now, it says so. Once the Generate button is clicked, a 4 digit random number is generated and displayed. This random number is also stored temporarily in a database table called *code* which is used during metadata exchange for verification (see below). Here, we have opted to generate a 4 digit code, other implementations may opt for other type of codes according to their own requirements.
4. After generating the code, the user inserts the entity ID of the IdP and the generated code to the WAYF page of the SP and clicks the Add button.
5. Once the Add button is clicked, it is verified that entity ID field or code is not null and that the inserted entity ID is not already part of the federation (dynamically or statically). If any part of the verification process fails,

an appropriate error message is displayed and the user is redirected to the WAYF page where she can insert valid values again.

6. Assuming there is no error during verification, a request to retrieve metadata from that entity ID with some specific values is posted. The request contains the entity ID and the code that the user has entered and two hidden fields called *MetaAdd & ReturnTo*. The value of the *MetaAdd & ReturnTo* fields contain the entity ID of the SP and the URL of the services that the user requested at the first place which initiated the SAML flow. Remember that SimpleSAMLphp implementation of dynamic SAML requires that entity ID be the endpoints from where the metadata can be fetched. We have extended its approach to add verification.
7. Once the Appropriate end point of the IdP receives this request, it checks if there is any field called *MetaAdd*. If found, it knows that this is a special request for exchanging metadata with the requested SP. If not found, it assumes that it is normal metadata fetch request and returns its metadata. Since the request contain a *MetaAdd* field, it, then, checks for a code field and retrieves its value and verifies if the same value can be found at the code table of its MySQL database. If found, it indicates that this request for exchanging metadata is valid. If not found, an error message is returned to the SP.
8. Assuming that the code field contains a valid code, the IdP retrieves the value of the *MetaAdd* field which contain the entity ID of SP and sends an HTTP GET request to that location. GET has been used since there is no other parameters to pass during the metadata fetch process from the SP.
9. Once the metadata is retrieved, the IdP goes thorough the specified verification process for a dynamic SAML (verifying the embedded certificate, verifying the signature on the metadata using that certificate, etc.). If the verification is correct, the metadata is stored in its repository and the SP is added to its TAL list. In addition, the entity ID of the SP is added into the *untrusted* table of the database along with the code and the used code is removed from the *code* table to ensure that it cannot be reused again. If the metadata verification is not correct, an error message is returned to the SP.
10. If everything goes right at the IdP, the metadata of the IdP along with the *(*ReturnTo) field and its value are returned to the requesting endpoint of the SP, where the metadata and *ReturnTo* values are separated. As before, the SP goes thorough the specified verification process for a dynamic SAML. If verification is correct, the metadata is stored in its repository and the IdP is added to its TAL list. In addition, the entity ID of the IdP is added into the *untrusted* table of its database and thus the IdP is tagged as an untrusted IdP.
11. Then, the user is redirected to the URL retrieved from the *ReturnTo* field (the URL of the service requested initially) which in turn takes the user back to the IdP selection page of the WAYF. However, as the IdP has already been added, the list contains the list of the IdP and it is tagged as an untrusted IdP (Figure 2). Moreover, it is shown to the user that the IdP has already

been added as an untrusted IdP so that no other users tries to add it once again which will result in an error.

12. Now, if the user selects the IdP, the usual SAML flow will take place. A SAML authentication request will be sent to the selected IdP and if the user is not already authenticated at the IdP, she will be prompted for login.

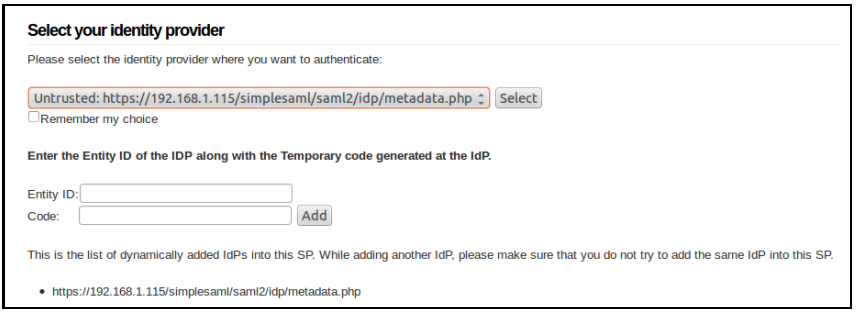


Fig. 2. Added IdP at the WAYF

13. Once the user is logged in, the Consent module is called internally. It reads the *semitrusted.sp* configuration parameter and displays the attributes automatically. Figure 3 shows the consent form. The consent page also states which attributes are filtered out from the full set and why. At this point, the user can choose which attributes she wants to release to the SP.

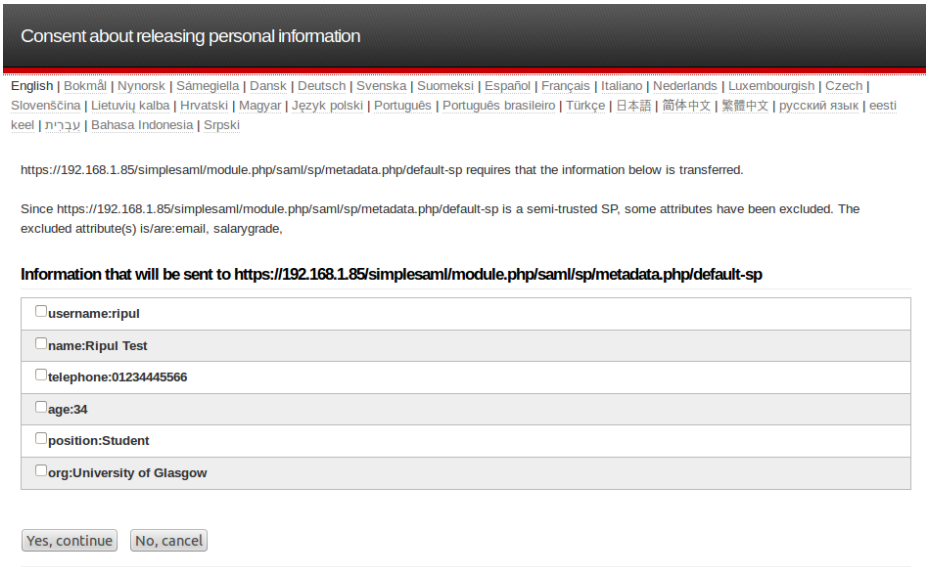


Fig. 3. Attributes to be released at the Consent page

14. If the user has chosen to release any attribute(s) by clicking the 'Yes, continue' button, the entity ID of the SP is removed from the *untrusted* table and inserted into the *semitrusted* indicating that the SP will be tagged as a semi-trusted entity hereafter. If the user chooses not to release any attribute, the entity ID of the SP will remain in the *untrusted* table. At this point, the chosen attributes would be released to the SP.

At this point, the SP knows from the *untrusted* table of its database that these attributes have been released by an untrusted IdP. Therefore, the respective SP will treat these attributes coming from an IdP having LoA level of maximum 1 and authorise the user accordingly.

4.2 IdP-IdP-SP Scenario

The protocol flows described above will allow the creation of federations in a fullly dynamic fashion and also consider the trust issued involved at both ends. However, the main problem is that the SP may not trust all attributes coming from the untrusted IdP even though the IdP is honest. The problem can be resolved if it is possible to link the untrusted IdP with an IdP which is fully trusted by the SP. In such a case, the fully trusted IdP would act like a Proxy IdP as described in [10]. The SP would think that it is talking to the fully trusted IdP while in fact the proxy IdP would delegate the authentication service to the untrusted IdP which is hidden from the SP. The untrusted IdP would release the user attributes to the fully trusted IdP once the user is authenticated which will be then retrieved and returned to the SP in such a way that the SP will think that the attributes have been released by the fully trusted IdP. Another advantage is that the SP no longer needs to support dynamic federations, since the proxy IDP can, in a trustworthy manner, add new IDPs indirectly. As before, we have used SimpleSAMLphp to demonstrate this scenario. The SimpleSAMLphp allows multiple authentication sources (including another SAML IdP) for authenticating a user. This can be enabled by the *MultiAuth* module of SimpleSAMLphp. We have used this feature to add the untrusted IdP as one of the authentication sources for the fully trusted IdP. To the untrusted IdP, the fully trusted IdP would be treated as a normal SAML SP, however, the fully trusted IdP would treat the untrusted IdP as the SAML IdP. To enable this configuration, it needs the authentication source to be pre-configured by exchanging metadata just like a federation. Like the previous scenario, we have modified the SimpleSAMLphp to automate this procedure so that two prior unknown SAML IdPs can be linked (in other words federated) in a fully dynamic fashion. However, this approach introduces an inconsistency which a malicious user might abuse for elevation of privileges. Since the SP would think that the attributes from the linked IdP have come from a trusted source (the proxy IdP), they might be tricked in trusting them. To ensure that it does not happen, the trusted IdP must add a LoA value of maximum 1 into the assertion in cases it has received any attributes from a dynamically linked IdP to them and return the assertion with that lower LoA. This would help the SP to decide how much trust they can put in those attributes.

With this setup, the protocol flow for the scenario is given below.

1. The user logs in to the untrusted IdP and generate a code just like the previous one. This code will be used to link the proxy IdP with this IdP.
2. Now, the user needs to log in to the proxy IdP. Since the IdP has enabled the *MultiAuth* module, the IdP shows all authentication sources (Figure 4). As there is no other authentication sources, it only shows the Userpass source which allows the user to log in to the IdP by using Username/password. The user selects this source and logs in using her username and password. After login, the user clicks the 'Link Another IdP' option.

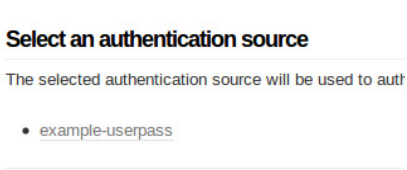


Fig. 4. Only Username/password source at the IdP

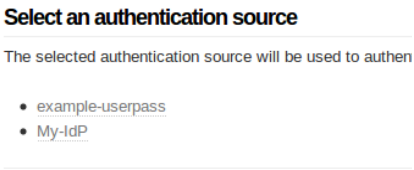


Fig. 5. Linked untrusted IdP as the authentication source in the proxy IdP

3. The user is presented with a page which has three fields: IdP ID field for entering the entity ID of the untrusted IdP, Code field to enter the generated code from step 1 and a Name field to enter a *Nick Name* for the untrusted IdP. This *Nick Name* will help the user to remember the IdP once it is added as one of the authentication sources. This field is not needed at the SP because each IdP is listed using its entity ID, whereas at the proxyIDP the IdPs are listed as authentication sources using a user friendly name. After entering all this information, the user clicks the Submit button.
4. Once the Submit button is clicked, it is checked to make sure that all information has been entered in the three fields. If not, appropriate error messages are displayed. If yes, a request to retrieve metadata from that entity ID is submitted with some specific values just like the way discussed in the previous scenario.
5. At this point, code is verified, metadata between two IdPs are exchanged, verified and stored just like the previous scenario. At this point, the two IdPs are linked.
6. Now, the user visits the SP to access one of the services. Assuming there is no security context, the user is redirected to the WAYF Service where the list of federated IdPs are shown. The user selects the fully trusted IdP (proxy IdP) and a SAML Authentication request is submitted to that IdP.
7. The IdP displays the list of authentication sources. As the untrusted IdP is already linked, the user can see the nick name (My IdP) of the untrusted IdP which was given during the linking phase (Figure 5).
8. Once the user selects the untrusted IdP, she is redirected to the IdP where the user logs in and the consent page with attributes are shown. As the proxy IdP is not tagged as the semi-trusted SP, all attributes that the user chooses will be released to the Proxy IdP.

9. Once the user clicks the 'Yes, continue' button, a SAML assertion with all attributes is sent back to the proxy IdP where all attributes are retrieved. The proxy IdP builds a SAML assertion with these attributes with a LoA value of 1 and it is sent back to the SP.

What the SP will do with all these attributes is not further explored here.

5 Discussion and Future Work

Creating dynamic federations using our approach has several advantages:

- Users can create federations just in time and whenever required. Even though it was not considered in our implementation, any IdP or SP can decide on how long it would allow the other party that has been added dynamically to remain in the federation by using a time threshold. Once the threshold is reached, the respective entry can be removed automatically from the TAL list, thus de-federating the entity.
- By using separate trust domains inside the same federation for fully trusted, semi-trusted and untrusted entities, a federation can host all types and leverage the advantages of all in the same configuration. However, one must keep in mind that the types of treatments semi-trusted entities will receive will fully depend on a particular implementation since the behaviour is not standardised.
- One of the requirements for an ideal IMS is the Segregation of power which is required to ensure that no single entity will have dominant position over other entities and users have the ability to choose a specific entity for a specific scenario [23,11]. Since the traditional SAML enforces users to use only those IdPs that can be in the TAL, segregation of power is not fully exercised [11]. Allowing the dynamic creation of federations would allow the users to choose a specific IdP for a specific scenario and hence ensuring the segregation of power.
- Allowing users to link two of their IdPs would enable any users to use their own IdPs (e.g. a locally installed OpenID Provider) via a trusted IdP. This would help to aggregate attributes from different sources. For example, the local IdP may provide some dynamically created attributes (e.g. location data etc.) which would be difficult for the fully trusted IdP to provide.

However, it should be kept in mind that the IDP-IDP linking (the second use case) means that a highly trusted (Proxy) IDP which can normally issue assertions at high LOAs, has to issue all its assertions at LOA 1 when the user authenticated via a linked IDP. This is not ideal, and may thus lessen the value of the service.

There are a few directions to take from here. The current implementation requires that both SAML IdPs are online for exchanging metadata during the linking procedure. To enable a user to link her local SAML IdP (IdP residing in the user's PC) with the trusted IdP, we need to find a way for the trusted IdP to

communicate with the local IdP for exchanging metadata. It could be another interesting topic to investigate the ways different dynamic attributes could be aggregated from different dynamically added IdPs.

6 Conclusion

In this paper, we have explored the avenue of the dynamic SAML. We have provided a proposal for creating dynamic federations in a fully automatic fashion. Our approach is simple in nature and can be easily adopted by any SAML implementation and requires no modification of the SAML Protocol. We have also examined the trust issues involved in such scenarios and illustrated two use-cases with detailed protocol flows. However, it should be noted that the issues of trust are very complex. The way we have outlined the trust issues may not be suitable for all scenarios. For example, an IdP may be reluctant to trust any SP which is not pre-configured in the traditional static way and thereby hesitant to release any attributes to it. In such cases, it will be very difficult to create federations in a dynamic way. We believe that IdPs and SPs will need to relax the trust requirements if they want to allow their users to take advantage of the dynamic federation. How much relaxation it will be required depends entirely on a specific use-case and will dictate the positive effect a dynamic federation can bring to their users.

References

1. Authentication processing filters in simplesamlphp, `http://simplesamlphp.org/docs/stable/simplesamlphp-authproc`
2. Microsoft Windows CardSpace, `http://www.microsoft.com/windows/products/winfamily/cardspace/default.mspx`
3. Shibboleth, `http://shibboleth.internet2.edu/`
4. SimpleSAMLphp, `http://simplesamlphp.org/`
5. Liberty Alliance Whitepaper: Benefits of Federated Identity to Government (March 2004), `http://projectliberty.org/liberty/content/download/388/2723/file/Liberty_Government_Business_Benefits.pdf`
6. OpenID Authentication 2.0 - Final (December 5, 2007), `http://openid.net/specs/openid-authentication-2_0.html`
7. Arias Cabarcos, P., Almenárez Mendoza, F., Marín-López, A., Díaz-Sánchez, D.: Enabling SAML for Dynamic Identity Federation Management. In: Wozniak, J., Konorski, J., Katulski, R., Pach, A.R. (eds.) WMNC 2009. IFIP AICT, vol. 308, pp. 173–184. Springer, Heidelberg (2009)
8. Bargh, M.S., Hulsebosch, B., Zandbelt, H.: Scalability of trust and metadata exchange across federations (December 2010), `https://tnc2011.terena.org/getfile/693`
9. Chadwick, D.W.: Federated Identity Management. In: Aldini, A., Barthe, G., Gorrieri, R. (eds.) FOSAD 2007/2008/2009. LNCS, vol. 5705, pp. 96–120. Springer, Heidelberg (2009)
10. Chadwick, D.W., Inman, G.L., Siu, K.W.S., Ferdous, M.S.: Leveraging social networks to gain access to organisational resources. In: Proceedings of the 7th ACM Workshop on Digital Identity Management, DIM 2011, pp. 43–52. ACM, New York (2011)

11. Ferdous, M.S., Poet, R.: A comparative analysis of Identity Management Systems. In: 2012 International Conference on High Performance Computing and Simulation (HPCS), pp. 454–461 (July 2012)
12. Harding, P., Johansson, L., Klingenstein, N.: Dynamic security assertion markup language: Simplifying single sign-on. IEEE Security Privacy 6(2), 83–85 (2008)
13. Baseline, I.-T.: capabilities for enhanced global identity management and interoperability (September 2009), http://www.itu.int/ITU-T/recommendations/rec.aspx?rec=X.1250
14. Jøsang, A., Al, M., Suriadi, Z.S.: Usability and privacy in identity management architectures. In: ACSW 2007: Proceedings of the Fifth Australasian Symposium on ACSW Frontiers, pp. 143–152 (2007)
15. Jøsang, A., Fabre, J., Hay, B., Dalziel, J., Pope, S.: Trust requirements in identity management. In: Proceedings of the 2005 Australasian Workshop on Grid Computing and e-Research, ACSW Frontiers 2005, pp. 99–108. Australian Computer Society, Inc., Darlinghurst (2005)
16. NIST. Electronic authentication guideline: Information security (April 2006), http://csrc.nist.gov/publications/nistpubs/800-63/SP800-63V1_0_2.pdf
17. Ferdous, M.S., Jabed, M., Chowdhury, M., Moniruzzaman, M., Chowdhury, F.: Identity federations: A new perspective for bangladesh. In: 2012 International Conference on Informatics, Electronics Vision (ICIEV), pp. 219–224 (May 2012)
18. Andreas Solberg. Dynamic SAML (February 18, 2010), https://rnd.feide.no/2010/02/18/dynamic_saml/
19. OASIS Standard. Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0. (March 15, 2005), http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf
20. OASIS Standard. A profile for distributed SAML metadata management (October 22, 2007), https://spaces.internet2.edu/display/dsaml/A+profile+for+distributed+SAML+metadata+management
21. OASIS Standard. SAML V2.0 Metadata Interoperability Profile, Working Draft 01 (August 1, 2008), https://spaces.internet2.edu/download/attachments/11275/draft-sstc-metadata-iop-01.pdf?version=2&modificationDate=1217876016355
22. OASIS Standard. Web Services Federation Language (WSFederation) Version 1.2 (May 22, 2009), http://docs.oasis-open.org/wsfed/federation/v1.2/os/ws-federation-1.2-spec-os.pdf
23. Future of Identity in the Information Society WP3. Study on Mobile Identity Management (May 2005), http://www.fidis.net/fileadmin/fidis/deliverables/fidis-wp3-del3.3.study_on_mobile_identity_management.pdf
24. Xiang, Y., Kennedy, J., Egger, M., Richter, H.: Network and trust model for dynamic federation. In: Proceedings of the Fourth International Conference on Advanced Engineering Computing and Applications in Sciences, pp. 1–6 (2010)
25. Zuo, Y., Luo, X., Zeng, F.: Towards a Dynamic Federation Framework Based on SAML and Automated Trust Negotiation. In: Wang, F.L., Gong, Z., Luo, X., Lei, J. (eds.) Web Information Systems and Mining. LNCS, vol. 6318, pp. 254–262. Springer, Heidelberg (2010)

Logout in Single Sign-on Systems

Sanna Suoranta[1], Asko Tontti[2], Joonas Ruuskanen[1], and Tuomas Aura[1]

[1] Aalto University, Department of Computer Science and Engineering
[2] CSC - IT Center for Science
Espoo, Finland

Abstract. Single sign-on (SSO) helps users to cope with many online services that require authentication. Systems such as OpenID and SAML-based Shibboleth offer federated identity management where an Identity Provider authenticates the user on behalf of the services. Much research concentrates on making authentication stronger, preventing phishing and making the systems more user friendly but less attention has been paid to the termination of the authentication sessions i.e. logout. It is, however, equally important that the sessions do not remain open when, for example, a student using shared computers in a university library leaves the workstation. In this article, we describe challenges related to logout in federated identity management on web based services and give guidelines for implementing reliable logout from services that use single sign-on.

Keywords: Single Logout, Logout in Single Sign-On Systems, Shibboleth.

1 Introduction

Nowadays both at work and in leisure time people are using many services that have a web browser as the user interface and platform instead of stand-alone applications. Often the services require the user to authenticate in order to provide personalized service, to access to the user's private information or to verify the user's real-world identity. Usually user authentication is based on passwords, but many users choose poor passwords [10] or share the same password between unrelated services [11]. These issues weaken the authentication.

The single sign-on (SSO) systems help users to access several services using a single password; thus, the user does not need to remember service specific passwords. In a federated SSO, the user authentication is separated from the service to be handled on a separate Identity Provider (IdP). Two common systems are OpenID [17] and Shibboleth [21], which is based on SAML [20]. These are mainly used locally or for specific online services. No dominant world-wide SSO system exists as the service providers do not want to outsource their user management [8]. The SSO technologies are an active field of researc and development. Impovements have been proposed to the architecture [4,5,23,27], authentication strength [16,26] , usability [14,24,25], and privacy [1]. On the other hand, less attention has been paid to the termination of the authentication sessions, even though it is a critical part of the authentication process.

SSO systems are common in universities where, in addition to the university itself, also external organizations can offer web-based services for students and staff. For example in Finland, the universities have a common library service that provides access

S. Fischer-Hübner et al.(Eds.): IDMAN 2013, IFIP AICT 396, pp. 147–160, 2013.

to online electronic publications using Shibboleth for authentication. The university students often use shared computers; thus the proper logout from the services is very important so that the next person using the computer cannot gain access to the services using the first one's privileges.

In this article, we have investigated the logout and its implementation in services that use the Shibboleth SSO. We found many problems in the logout procedures: none of the services implements single logout, and usually at least some server side sessions are still active after logout. Based on our findings, we give a list of actions that help in implementing more secure and usable logout for Shibboleth. Some of the solutions are general and could be used to improve also other SSO systems.

The rest of the paper is organized as follows. Next, we discuss single sign-on systems in general, and OpenID and Shibboleth in detail. In Section 3, we go through how logout works in SSO systems in theory, and in Section 4 how it works in practice in the web services of Aalto University. Then we discuss the solutions for logging out in the SSO systems. In Section 6, we describe what the other researchers have done. Then we list the improvements that should be implemented and, finally, conclude the article in Section 7.

2 Single Sign-on

Single sign-on systems allow users to log in to several services with one authentication. The SSO systems can be divided into four categories: local and proxy-based, pseudo and true SSO systems [18]. In the local pseudo-SSO system, a user's device has a password manager that automatically works on behalf of the user when she is logging into a service. The proxy-based pseudo-SSO system is similar, but the password manager is located on a network proxy. The local true SSO system rely on trusted component on the client device, but otherwise is quite similar to the local pseudo-SSO. The proxy-based true SSO system has an external server for authenticating the user for services. In this article, we concentrate on proxy-based true SSO systems, often called federated SSO. For example, OpenIDand Shibbolethare proxy-based true SSO systems. In addition, many service providers allow other web service developers to use their user authentication with their proprietary protocols. For example, Facebook Login [9] and Microsoft's Windows Live ID [15] provide centralized authentication for third party web applications where users can log in using their existing identities at Facebook and Microsoft's services.

Both OpenID and Shibboleth separate the user authentication from the actual service. Figure 1 depicts the general architecture of a federated SSO system. When a user wants to use a service, she first starts her web browser and connects to the service that is running on a server of a Service Provider (SP). A SP can have several services on the server, and some of them can take care of their own session management but often the SP saves the session information on behalf of the service. The SP responds to the unauthenticated user by redirecting the browser to an Identity Provider (IdP), or if there are several IdPs, the user can choose to which one the connection is redirected. The IdP authenticates the user, creates a SSO session for her, and redirects the user's web browser back to the SP with authentication assertions. Then, the SP saves the session

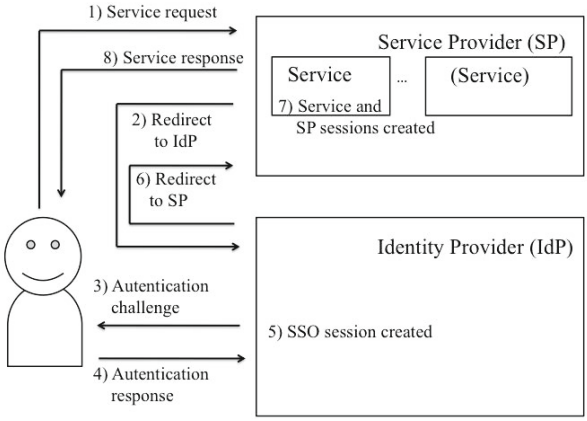


Fig. 1. Architecture of a federated SSO system

information and gives the originally requested access to the user. If the user wants to use another service with the same authenticated identity, the IdP does not need to re-authenticate her. It only checks that the valid SSO session exists, and then it gives the authentication assertions directly to the second SP. However, the IdP does not keep track to which SPs the user has been authenticated. Next, we describe shortly the differences of OpenID and Shibboleth.

2.1 OpenID

OpenID allows user to choose any identity provider she wants. There are plenty of OpenID identity providers [28] but only a few services that accept other than their own IdP.OpenID does not require pre-established trust between SP and IdP [22], and that might be one reason why it has not gained worldwide acceptance as a service authentication solution even though many services use it for access control with their own IdP. Moreover, popular OpenID identity providers such as Google do not verify the user's identity in the registration phase, only that the user has a valid email address [5]. However, strong user authentication is possible, e.g. in Estonia, a mobile phone operator acts as an IdP that provides strong verified authentication [12].

2.2 Shibboleth

Whereas OpenID requires no prior trust relationship between IdP and SP, Shibboleth participants need to trust each other. However, the IdPs and SPs do not need to belong to a same organization, they just need to share common metadata declaring the trusted IdPs and what attributes each service needs. IdP and SP communicate using SAMLmessages through user's browser using HTTP redirect and post command. Authentication and service sessions are stored in cookies on the cliend side.

In Finland, all the universities have joined to a federation called HAKA [6] that handles federated identity management. The universities have their own IdPs and a separate organization, namely CSC that provides high performance computing facilities and Internet access to universities, manages the federation and its metadata on behalf of the universities. The HAKA federation and its services have worked well for reducing the number of necessary passwords. Furthermore, the federation has joined together with the other Nordic federations providing access to research resources over the borders in the Nordic countries [13]. We have chosen to investigate how the services using the Shibboleth SSO manages the logout.

2.3 Implementing the Shibboleth SSO

Shibboleth Consortium [21] offers open source implementations for selected SAML2 profiles as Shibboleth IdP and SP. Their Shibboleth IdP is a Java based server that works on top of Tomcat [2] Java Servlet platform. It offers several ways to authenticate the users: e.g. username and password query, remote user, and previous session. The first and second methods are for initial phase user authentication and the third allows the actual single sign-on after the user's identity has once been verified. The first asks username and password from the user and checks that they are correct using Java Authentication and Authorization Service (JAAS) interface to connect to a LDAP directory (this method is used in Aalto University's IdP) or Kerberos server. The second method uses a web server, such as Apache, to authenticate the user instead of the IdP program. The third method uses authentication cookies that are stored in the web browser after previously and successfully done user authentication. This allows user to log in to several services with single authentication.

The Shibboleth SP is usually connected to web server as additional module or filter. The web server can be e.g. Apache or Microsoft ISS. The web server mediates the authentication information to the web service application that offers the actual service for the user. The web server and SP pair can host several service applications. There are two basic ways for the service to use Shibboleth for authentication. The service taking care of its own user information management during the session can use Shibboleth for user authentication. The other way is that the service outsources all user session management to the SP.

3 Logout in SSO Systems

Logging out means clearing the sessions that are created when the user is signed in. In a traditional web service, the user session is usually terminated either when the user explicitly presses logout button or when the service timeout is reached. In the SSO systems, logging out is not so straightforward. The user might want to log out either from a single service or from the whole SSO session. From the SSO point of view, there are even more forms of logout, e.g. Shibboleth and SAML have the following:

– *Logout in the service application*. If the application takes care of its own user session management, it terminates the user sessions but the SP session needs to be terminated separately. If the SP takes care of the user session management on behalf of the service, the service need to register a handler in SP for logout.

- *Logout in SP*. If SP takes care of all user session management, it offers an API to the services to start the logout in SP. In logout, the SP removes the sessions. In an IdP initiated logout process, the SP removes its sessions and asks the application to end its sessions, too, if necessary.
- *Logout in IdP*. The IdP takes care of the SSO session during which the user can log in to services without re-authentication. If logging into other services is not preferred, the IdP session can be removed. Removing the IdP session does not affect the session any way on the SP and the services.
- *Local logout*. If the user wants to log out from a single service, she uses the local logout that closes both the service and SP sessions related to that service but not the IdP session, nor sessions of other services and SPs.
- *Single logout (SLO) or Global Logout*. If the user wants to log out from all the services, she chooses the single logout. In addition to the currently used service and SP, the IdP goes through all SPs and asks them to log out the user, and ends its own session (IdP initiated logout).
- *Partial logout*. When the service does not work correctly in logout, the result is a often partial logout. Then some of the user's sessions, typically either in the service or in the SP, remains active. The only way to recover from this situation is to close the web browser.

In the SSO systems, the session should end either to local logout or single logout. In practice, the latter is often local logout combined to the IdP logout, not a real SLO. Usability studies [14,25] has shown that when using a SSO system, the users expect also single logout. Correctly implemented SLO is extremely important when shared computers are used e.g. by students in a university library. However, many users have personal computers, thus a local logout from a single service might often be more practical, e.g. when the user will log in to another service with the same IdP after logging out from the first service. If also the IdP session has been terminated, the benefits of using the SSO system are overturn since the user has to sign in again every time.

Next we go through how SLO works in different SSO systems and federations.

3.1 SAML2 Logout

OASIS, the standardization organization behind SAML, took into account the lessons of another federated identity management system, namely Liberty Identity Federation Framework (ID-FF), while developing SAML2. Liberty ID-FF and Web Service Framework (WSF) allow service providers to communicate directly with identity provider through back channel, which allows SLO in Liberty [1].

SAML2 only describes the logout in general and different profiles complete it with conflicting suggestions. For example, the order of removing the sessions on SP and IdP is not clear. Following describes the SAML2 logout with front channel binding that uses web browser to communicate with different actors.

1. The user chooses logout in a web service.
2. The service removes the user's session.
3. The service redirects the web browser to the SP's logout handler. It can set a return parameter if the browser should be redirected to an URL after successful logout.

4. The SP removes the user's SP session.
5. The SP searches the IdP's logout service address from the metadata and redirects the web browser to it with the SAML2 LogoutRequest message.
6. The IdP removes the user's SSO session (PreviousSession)
7. If necessary, the IdP asks an external authentication service to remove its session.
8. The IdP sends the SAML2 LogoutRequest messages through the web browser to all the SPs except that which started the logout. Each SP removes its sessions and asks the services to remove the sessions, too. All the SPs send messages about success of the logout to the IdP through the web browser.
9. The IdP sends information about the success of the logout through the web browser using the SAML2 LogoutResponce message to the SP that started the process.
10. Finally, the SP can either tell the user that logout has been proceeded successfully or even redirect the user to the URL given in step 3.

The SAML2 logout provides SLO but requires the IdP to keep track of all the SPs that the user has logged in, which requires more resources. Moreover, if the user has logged in using different IdPs, only the services using the same IdP will be logged out.

3.2 HAKA Federation Logout

Since the first version of Shibboleth does not provide any logout, the HAKA federation developed its own logout for Shibboleth SSO used in Finnish universities. The result is usually only partial logout where the service and SP sessions remains open in the other SPs than the one starting the process. The beginning of the HAKA logout works as the SAML2 logout in the steps 1-7, and the rest as follows [7].

8. The web browser returns to the URL if SP askedit while it called the IdP.
9. The browser tells the user that she is logged out from the service and the IdP.
10. User is encouraged to close the web browser to be sure that logout has been completed. The purpose is to prevent using the session from other SPs than that started the logout.

However, even though the user would close the web browser, other SPs and their service sessions remain open until timeout. If an attacker finds out the session identifier, he can use them for a while. Shibboleth SP 2 supports SAML2 logout but the IdP 2 does not. Future IdP 3 will probably have a back-channel-based SLO where the messages are passed using SOAP.

3.3 OpenID Logout

OpenID does not provide single logout although IdPs can provide a logout URL for the service providers. For example, Google staff in the developer discussion forum suggests that the service developer should either provide logout only from the service or redirect the browser to Google's logout URL [3]. However, the service should know which IdP the user has used to log in to the service in order to provide the correct logout URL. This is hard because there is no pre-established connection between the SP and IdP. Furthermore, OpenID IdP does not keep track of the services the user has logged in.

4 Logout in Practice

We went through commonly used services of the HAKA federation to find out why they behave differently when a user logs out from the service. Almost all possible logout forms listed in Section 3 were found. In this section, we present the results of our tests.

4.1 Only Service Session Removed

Half of the Finnish universities use Oodi[1] to record all course grades, and for students, to sign in for courses and exams. The front-page of the service reminds the users to exit from all browser windows when logging out to prevent the next user to gain access to the first user's information. The warning is needed since when the user logs out, only service's local session is terminated and both the IdP and SP sessions remain. The service returns to the service front-page and gives no information about logout status.

In the Oodi login process, two sessions are created: a SP session with eight hours timeout and a local Tomcat session with 30 minutes timeout. The Oodi service copies all information from the SP's session to a session presented by JSESSIONID cookie. When the user logs out from the service, only the Oodi service's own session cookie is removed but the SP session still remains, let alone the IdP session. Same way, when the local session reaches timeout, the service does not inform the SP about the logout. This means that the user can get directly back in to Oodi without re-authentication because the SP session is still valid.

In this case, the biggest security problem is that the user cannot get out from the service at all if she does not close the web browser. The service does not tell to the user about still valid sessions. Even if she closes the web browser, both the SP and IdP SSO sessions remain open until timeout. The problem is Shibboleth specific since the session management is divided between the SP and the service application.

4.2 SP Session Removed but Service Session Exists

The libraries of Finnish universities have a joint service called the Nelli[2] portal for scientific articles and databases. When the user logs out from the service, the connection is redirected to IdP. The IdP tells the user has logged out from the SSO session and that there might still be active sessions, and only closing the browser will end them. Indeed, there still is an active session, because if the user immediately returns to the Nelli portal and press the login button, she gets in without re-authentication. However, she does not get in to new services since the IdP session has actually been removed.

The Nelli portal is implemented using MetaLib program that uses Patron Directory Services (PDS) for authentication [19]. When the user logs in to the Nelli portal, the service checks first a PDS_HANDLE cookie. If it exists, its information is copied to the main session. Otherwise the service asks the user information from the SP. Nelli creates also a local session key that is stored in a ML_SESSION_ID (MetaLib Session ID) cookie and included in all page URLs. If this cookie does not exist, it is always

[1] E.g. `https://oodi.aalto.fi`
[2] `http://www.nelliportaali.fi`

created, thus removing it does not affect anything. Removing the key from the URL does not affect either, if the ML_SESSION_ID cookie exists. The Nelli service session does not end until the user navigates to other pages for more than 10 minutes or press the logout button. In addition to service specific cookies, SP removes its session but because the IdP session is still valid, the next user can get in without authentication. Removing first the SP session eliminates the possibility to logout from the IdP because the necessary information was stored in the SP cookie.

When the user logs out from the Nelli portal, the service does not remove the PDS_HANDLE cookie but the SP session is already ended. The problem in this case is that the service session remains open. Thus the user can get in without the service checking from either IdP or SP if their sessions are still valid.

In a way, the logout from the Nelli portal does not work at all. Different sessions are active, even though the user think that they have been ended. The problem is Shibboleth specific but session timeout can be problem in other SSO systems, too.

4.3 Local Logout

The Aalto University Wiki[3] tries to execute local logout. When the user logs out from the wiki, the connection is returned to login page of the wiki service. Only one line of text "You have been successfully logged out. Return to wiki.aalto.fi or login again" separates the page from the original login page that offers several possibilities to log in. The user may not notice the text at all.

Actually the Wiki service tries to execute SAML2 logout instead of HAKA logout since it asks the SP to redirect the connection to its own logout pages and not to IdP. Because the current IdP does not handle SAML2 logout, the SP removes only its session and rest of the session removals fail.

The problem is that the user is not informed from which session she has logged out, and that this was due to compatibility problems. If the user closes the web browser, the IdP SSO session and possible other SP sessions remain open until timeout. If the user does not close the web browser, the next user can get in to all services using the same IdP SSO session, even to the wiki. The problem is generic in the SSO systems.

4.4 Local Logout together with IdP Logout

All the course webpages and announcements are in the Noppa[4] portal that was developed at Aalto University and later brought into use in two other universities. When the user signs in the portal, both the SP and Noppa's local Tomcat sessions are created. All user information is copied to session identified by either JSESSIONID cookie or org.apache.tapestry.locale cookie. The SP session is valid for eight hours, but if it is unused, it expires in one hour. The Noppa portal offers only the local logout together with the IdP logout for the user.

The user can continue using other services even though the IdP session has ended because the IdP does not have means to inform the other services to end their sessions.

[3] https://wiki.aalto.fi/dashboard.action

[4] https://noppa.aalto.fi/noppa/app

Thus, the real SLO is not executed. If the user closes the web browser while other services are still active, their SP and service sessions remain active until timeout. The problem is generic in the SSO systems and may confuse users that cannot separate the real SLO from logging out from the IdP.

4.5 Choosing between Local and IdP Logout

One obsolete service offered to a user a possibility to choose between the local and IdP logout, which was implemented as the HAKA logout from both the SP and IdP. When the user logged into the service, the SP copied the essential user information into long hash and gave it to the service as part of the URL. The service removed the string from the URL and created a cookie with the same name. When the user logged out, the session information was removed from the server memory.

It seems that the logout implementation has been left for the service developers to choose. Moreover, many of the services do not actually provide correct logout, they just end some of the sessions. Furthermore, it is up to the user's enlightment, can she distinguish the local logout from the IdP logout when getting out from different services. The problem is generic since the users do not always know where they want to log out and offering choices may be too complicate for them.

5 Solutions for Logout in the Federated SSO Systems

In centralized SSO systems, an IdP keeps track to which SPs the user has been authenticated, and communicates to all SPs about logout through a backend channel, thus the single logout is possible. Federated SSO systems do not usually use backend channel for communication and an IdP does not know about active sessions on SPs.

There are several steps that make the logout in the federated SSO systems more secure. At least, the user should be able to logout from the service she is using but also ending the SSO session in the IdP is necessary, if chosen. To provide true single logout, the whole federated SSO needs modifications.

5.1 Modifications in Services

When a service is added to the federated identity management, it has to be modified. The developers should decide if the service keeps a session itself or does it use the session provided by SP. If the SP takes care of the sessions, the service just need buttons for the local and IdP logout. Pressing these buttons can start the logout procedures with proper attributes in the SP, and the SP takes care of everything.

If the service takes care of the logout, the procedure is quite the same as above but changes are needed. The service has to remove its own session before calling the logout method of the SP. Equally important is to have a handler to the IdP initiated logout. For this, the service should register a single logout handler for the SP. When a SLO request arrives, the service has to remove all sessions. The request can come through the web browser or using a SOAP interface. In the latter, it is not possible to remove the cookies from the browser, but the service has to remove the session from its own memory. Finally, the lifetime of service session should be shorter than the lifetime of SP session or otherwise the incoming logout requests do not work correctly.

5.2 Cookie Management in Web Browsers

Handling of the cookies has not changed much during the recent years. To mitigate problems caused by attacks, there are rules how cookies should be handled. These rules prevent the IdP and SP from seeing each other's cookies.

For making single logout possible, the rules should be changed. For example, the IdP could create a collection of cookies connected to the SSO session. The SPs could add their own cookies to the collection. Then as a creator of this collection, the IdP could remove cookies and set the deletion rights to all servers of the federation. In the single logout, the IdP could remove all the SP cookies that are left over. However, this change may cause unexpected security weaknesses if not done with extra care.

5.3 Connections between the IdPs and SPs

Single logout changes the actions of the IdP and SP. For example in the original Shibboleth, the IdP takes only care about the validity of SSO sessions and the SP has the necessary user attributes in its memory. The IdPs and SPs could have been rebooted since all necessary state information has been stored in the cookies on the user's web browser and the other information could have been fetched again. For single logout, the IdP has to store information about all SPs where the user is logged in. This would change a lot of the use and importance of IdP's SSO session.

Another way to do automatic logout is based on AJAX. The web browser could send *keep alive* messages to the SPs and IdP when the user has an open window for a service. When the user navigates to some other web page than that of the service, the browser simply stop sending these messages to the SP that can then end the user's session. This way the sessions will not be "left behind". Moreover, when the user has moved out from all the services of the federation, the web browser stops sending the keep alive messages to the IdP, too. Then the IdP can remove the SSO session and, to make single logout absolutely sure, start the IdP initiated logout fom all SPs. However, there are two problems: This approach burdens the IdP and SP with additional messages, and it does not work if JavaScript is turned off or the web browser does not support it.

On the other hand, single logout is possible to implement using a *polling mechanism*. When a user chooses single logout in a service, the service would execute both local and IdP logout, thus ending both its local session on the SP and service, and the SSO session on the IdP. The other SPs would notice that the SSO session is no longer valid because they ask periodically its existence using e.g. a SOAP interface of the IdP. Then, they too could execute the local logout, and all the sessions would end through true single logout. Of course, the interval of the queries should not burden the IdP. The downside is that with this addition Shibboleth would not be SAML2 compliant software.

5.4 Clear User Interface

It is hard for a user to separate a local and single logout and understand their differences when some services of the federation provide the first and some the latter. It would be nice if the user could see on a page the services where the authentication sessions that are still valid. Technically and from the user interface point of view this is hard.

Especially important for the user is to know that global single logout has really removed all the session information from all the services. Closing the web browser ends the sessions in the client side and solves the problem almost satisfactorily. However, the sessions on the server side remains until timeout but they cannot be used without knowing the session key. Another problem rises if the user has also other web browser windows and tabs open, and does not want to close them, just log out from one of the services. At least the web browsers should be improved to really allow to end the sessions without closing the whole browser.

For example, the WebApp of Microsoft Outlook asks from the user before login if the user has a private computer or if she is using a public computer. This affects how long timeout is used for the session when the user is inactive. In the same way, the login of a service could give two possibilities that finally lead to either local or single logout.

6 Related Work

Logout in SSO systems is not widely researched topic. Often it is only mentioned when some other aspects of the SSO systems are improved. For example, Sun et al. [25] investigated usability of OpenID system because a single sign-on system may confuse users when it redirects the user from a service to an identity provider for logging in. They found out that despite of many passwords, users are still reluctant to use SSO, e.g. for critical services because they do not understand how it works. They implemented a system [24,25] that integrates OpenID login into a web browser and they tested the usability of their system with a small group of users. They list single logout as one important requirement in their system but do not describe its features nor its usability, they only state that it must automatically end all authentication sessions. However, based on the screenshots of their solution, it seems to offer users two possibilities: either single logout or logout only from the current service. In their other article [23], the login process of their system is elaborated in more detail but handling the logout process is not described at all.

Cahill et al. [4] have created a client-based authentication system that has a trusted hardware-based secure container for user credentials in the user's device. The system allows both active and passive authentication. The latter means e.g. facial recognition using the device's camera or RFID badge that is near the device. Thus, service providers do not need to implement timeout for their service sessions, but also SAML-based SLO is implemented. The system requires much from the hardware and the user is bound to use one device. Mustafic et al. [16] describe similar system that combines SSO with continuous behavioral biometric-based authentication. Their system uses keystroke dynamics as proof that the user is still the same as that authenticated in the beginning of the session. If the system notices that the user has changed, it starts the SAML SLO. However, this only works for services that require typing.

Linden et al. [14] present a usability study on logout in a SSO system among students and personnel of a university. They found out that especially students expect also single logout when using single sign-on. On the other hand, the personnel stated that their sessions in the university services have no clear beginning or end. More important for the users was that they knew if they were authenticated or if they were anonymous,

all the time while they used the services. When Linden et al. did their study, the SSO systems were new and many universities implemented the student services as integrated web portals. This may explain some of the misunderstandings that single sign-on means also single logout.

7 Conclusions

In order to create reliable logout for the federated SSO systems, standardization organizations, web browser vendors, and service developers should consider also the termination of the sessions. For example, thorough testing of logout behaviour could improve the current situation. Moreover, the following changes would make logout better:

- unified and standardized process for ending the sessions,
- improving the cookie management in browsers,
- creating a mechanism to check the existence of an IdP session, and
- unified user interface for logout.

The *unified process* for removing the session cookies and managing their timeout periods would prevent the current problems of logout. The SAML single logout is too complex and contradictory to work properly. For example, the order of the session removal is not clear. The standardization should redefine the single logout process because the currently used federation specific definitions are not sufficient, as is shown in Section 4. Examples could also be provided for service developers on how the session management should be handled.

The *cookie management improvements* consist of two things: ending the sessions without closing the web browser by removing the cookies, and allowing the IdP to remove the SP cookies. The former depends on features in web browsers and requires support from browser vendors. The latter could be taken into account in standardization of HTML 5, for example. However, the current model of session management with cookies is not likely to change. Moreover, if the IdP keeps track of all SPs to which the user has logged in, there are implications to user privacy.

Another possibility for checking the existence of the IdP session is to create a *polling mechanism* for SPs. If the SP polls the IdP, the IdP does not need to keep track of the SPs to which the user has logged in. Since the user is not bound to use only a single IdP, the SP may need to poll all the IdPs that the users have chosen. Of course, the polling frequence must be reasonable and not burden the IdP and SPs.

The *unified user interface* means that a federation or the SSO system standardization should define from the user's point of view how the services should end the sessions. Because the user can work either with public or private computers, two options should be provided by all services: single logout that ends both the IdP and all service and SP sessions, and local logout that leaves the IdP session active. Also, all services should inform the user from where she has logged out.

The process of ending the authenticated sessions has not gained the attention it deserves. We tested many services that use Shibboleth as a SSO system and almost all had problems with logout. Because some of the session information remains when the

service ends in partial logout instead of local logout, the next user of a shared computer can access the service with the previous user's account. Where the single logout is suggested in the user interface, it often does not work as expected. Even though the IdP removes the SSO session, the service and SP sessions remain active. Often services rely on closing the web browser but even then the server side sessions remain until timeout. Small changes such as unified logout process and user interface would allow the services to provide at least local logout instead of partial logout. For true single logout, the federated SSO systems require modifications to the standards for cookie handling and federation practices such as a polling mechanism for checking the IdP sessions.

Acknowledgments. We whish to thank Aapo Kalliola and Jaakko Kotimäki for valuable comments.

References

1. Alsaleh, M., Adams, C.: Enhancing Consumer Privacy in the Liberty Alliance Identity Federation and Web Services Frameworks. In: Danezis, G., Golle, P. (eds.) PET 2006. LNCS, vol. 4258, pp. 59–77. Springer, Heidelberg (2006)
2. Apache Software Foundation. Apache Tomcat (2012), http://tomcat.apache.org
3. Balfanz, D.: Sign-out from Google federated login api. Google API discussion forum (February 13, 2009), https://groups.google.com/forum/?fromgroups=#!searchin/google-federated-login-api/sign-out/google-federated-login-api/dBpKzRh1Amc/dEJhGRDwTE0J
4. Cahill, C.P., Martin, J., Phegade, V., Rajan, A., Pagano, M.W.: Client-based authentication technology: User-centric authentication using secure containers. In: DIM'11: Proceedings of the 7th ACM Workshop on Digital Identity Management, pp. 83–92. ACM (2011)
5. Chadwick, D.W., Inman, G.L., Siu, K.W., Ferdous, M.S.: Leveraging social networks to gain access to organisational resources. In: DIM 2011 Proceedings of the 7th ACM Workshop on Digital Identity Management, pp. 43–52. ACM (2011)
6. CSC - IT Center for Science. Haka federation (June 2006), http://www.csc.fi/english/institutions/haka (referred November 8, 2012)
7. CSC - IT Center for Science. Haka Logout (2012), http://www.csc.fi/hallinto/haka/ohjeet/ohjeet-yllapitajille/haka-logout (referred October 30, 2012)
8. Dhamija, R., Dusseault, L.: The seven flaws of identity management, usability and security challenges. IEEE Security and Privacy 6(6), 24–29 (2008)
9. Facebook. Getting started with Facebook login. docs/technical-guides/login/ (2012), https://developers.facebook.com/ (referred November 7, 2012)
10. Florêncio, D., Herley, C.: A largescale study of web password habits. In: Proceeding WWW 2007 Proceedings of the 16th International Conference on World Wide Web (2007)
11. Gaw, S., Felten, E.W.: Password management strategies for online accounts. In: Symposium on Usable Privacy and Security, SOUPS 2006, pp. 44–55 (July 2006)
12. Ideelabor. Openid in Estonia (2008), http://openiddirectory.com/openid-providers-c-1.html (referred February 27, 2009)
13. Kalmar2. Kalmar2 - access to nordic higher education with single login, https://www.kalmar2.org/kalmar2web/front_page.html (referred December 20, 2012)

14. Linden, M., Vilpola, I.: An Empirical Study on the Usability of Logout in a Single Sign-on System. In: Deng, R.H., Bao, F., Pang, H., Zhou, J. (eds.) ISPEC 2005. LNCS, vol. 3439, pp. 243–254. Springer, Heidelberg (2005)
15. Microsoft. Windows live id web authentication sdk (2012), http://msdn.microsoft.com/en-us/library/bb676633.aspx (referred November 7, 2012)
16. Mustafic, T., Messerman, A., Camtepe, S.A., Schmidt, A.-D., Albayrak, S.: Behavioral biometrics for persistent single sign-on. In: Proceedings of the 7th ACM Workshop on Digital Identity Management, DIM 2011, pp. 73–82. ACM (2011)
17. OpenID Community. Openid authentication 2.0 - final (December 5, 2007), http://openid.net/specs/openid-authentication-2_0.html (referred November 8, 2012)
18. Pashalidis, A., Mitchell, C.J.: A Taxonomy of Single Sign-on Systems. In: Safavi-Naini, R., Seberry, J. (eds.) ACISP 2003. LNCS, vol. 2727, pp. 249–264. Springer, Heidelberg (2003)
19. Pennanen, J.: Wiki nelli metalib (August 3, 2009), https://wiki.helsinki.fi/display/Nelli/MetaLib (referred November 7, 2012)
20. Ragouzis, N., Hughes, J., Philpott, R., Maler, E., Madsen, P., Scavo, T.: Security assertion markup language (saml) v2.0 technical overview. Technical report, OASIS (February 2007)
21. Shibboleth Consortium. Shibboleth (2012), http://shibboleth.net/
22. Sun, S.-T., Boshmaf, Y., Hawkey, K., Beznosov, K.: A billion keys, but few locks: the crisis of web single sign-on. In: NSPW 2010: Proceedings of the 2010 Workshop on New Security Paradigms (September 2010)
23. Sun, S.-T., Hawkey, K., Beznosov, K.: Openidemail enabled browser: Towards fixing the broken web single sign-on triangle. In: DIM 2010, October 8. ACM (2010)
24. Sun, S.-T., Pospisil, E., Muslukhov, I., Dindar, N., Hawkey, K., Beznosov, K.: Openid-enabled browser: Towards usable and secure web single sign-on. In: CHI EA 2011: Proceedings of the 2011 Annual Conference Extended Abstracts on Human Factors in Computing Systems. ACM (May 2011)
25. Sun, S.-T., Pospisil, E., Muslukhov, I., Dindar, N., Hawkey, K., Beznosov, K.: What makes users refuce web signle sign-on? an empirical investigation of openid. In: SOUPS 2011: Proceedings of the Seventh Symposium on Usable Privacy and Security. ACM (July 2011)
26. Suoranta, S., Andrade, A., Aura, T.: Strong Authentication with Mobile Phone. In: Gollmann, D., Freiling, F.C. (eds.) ISC 2012. LNCS, vol. 7483, pp. 70–85. Springer, Heidelberg (2012)
27. Takeda, Y., Kondo, S., Kitayama, Y., Torato, M., Motegi, T.: Avoidance of performance bottlenecks caused by http redirect in identity management protocols. In: DIM 2006: Proceedings of the Second ACM Workshop on Digital Identity Management. ACM (November 2006)
28. The OpenIDDirectory. Openid providers (February 2009), http://openiddirectory.com/openid-providers-c-1.html (referred February 27, 2009)

Author Index