TLS-Federation – a Secure and Relying-Party-Friendly Approach for Federated Identity Management

Bud P. Bruegger[1] · Detlef Hühnlein[2]· Jörg Schwenk[3]

[1] Comune di Grosseto, Italy
bud@comune.grosseto.it

[2] secunet Security Networks AG, Germany
detlef.huehnlein@secunet.com

[3] Ruhr Universität Bochum, Germany
joerg.schwenk@nds.rub.de

Abstract: Federated Single-Sign-On using web browsers as User Agents becomes increasingly important. However, current proposals require substantial changes in the implementation of the Relying-Party, and concentrate on functionality rather than security against real-world attacks like Cross Site Scripting (XSS) and Pharming. We therefore propose a different approach based on Transport Layer Security (TLS), which is implemented in any web browser and web server, and which is immune against all currently known attacks.

1 Introduction

Web-based services undoubtedly become more and more important. For example, it is believed (cf. **[Herr07]**) that by 2012, 80% of all private, and 50% of all professional users will cover half of all necessary applications and resources using web-platforms.

This rises a need for identification and authentication schemes which integrate existing user credentials, work across domains, and provide Single Sign-On functionality. As there are different approaches for Federated Identity Management – in particular the protocols and profiles specified within by the Liberty Alliance project (cf. **[LA-ID-FF-P&S]** and **[LA-ID-FF-B&P]**), and by the WS-* specifications (cf. **[WS-Federation]**, **[WS-Mex(v1.1)]**, **[WS-SecPol(v1.1)]** and **[WS-Trust(v1.3)]**), which are supported by identity agents such as Microsoft Card Space **[MS-CardSpace]** or Higgins **[Higgins]** and that are promoted by the Information Card Foundation **[ICF]** – it is natural to analyse the strength and weaknesses of the different approaches in order to determine which is most suitable in which scenario.

For this purpose, we will introduce an abstract model for Federated Identity Management (cf. Section 2.1) which is a generalization of the existing proposals (cf. Section 2.2). This model facilitates the analysis of the similarities, differences, advantages, and disadvantages of the existing approaches (cf. Section 2.3). Since this analysis will reveal that some proposals are vulnerable to well-known web-attacks, such as Cross-Site-Scripting (XSS) and Pharming (DNS spoofing), and since all existing approaches require substantial changes at "typical relying parties", we will introduce a novel approach called "TLS-Federation", which solves both kinds of problems: Firstly, the relying party only needs to support the ubiquitously deployed Transport Layer Security (TLS) protocol **[RFC2246]** which means that relying parties only need standard web servers without any extension. Secondly, the protocol that exchanges session credentials between user and Service Provider excludes wide classes of known web-attacks.

As explained in Section 3, the key idea of TLS-Federation is that the Identity Authority issues **[X.509]**-certificates instead of SAML-assertions or other security tokens, and that these short-lived session credentials are presented to the Service Provider within the extensively proven TLS-handshake protocol **[RFC2246]**. While TLS has been a well-established work horse of strong authentication for years, this paper newly proposes the use of TLS in a federated setting. Although the necessary technology is already ubiquitously available, the innovation of this paper is in the uncommon use of existing PKI-components and the proposal to use TLS and X.509-certificates instead of other federation solutions based on SAML or WS-*.

The rest of the paper is structured as follows: Section 2 contains the necessary background on Federated Identity Management, including the introduction of an abstract model and the discussion of existing approaches. In Section 3, we will introduce the concept of TLS-Federation and reason how it avoids some vulnerabilities and allows that Relying-Parties use standard web servers even in a scenario of Federated Identity Management. Section 4 finally draws some conclusions and provides an outlook on possible further steps.

2 Background on Federated Identity Management

2.1 An Abstract Model for Federated Identity Management

In this section, we will introduce an abstract model for Federated Identity Management which will serve as a reference when summarizing the existing approaches in Section 2.2 and introducing TLS-Federation in Section 3.

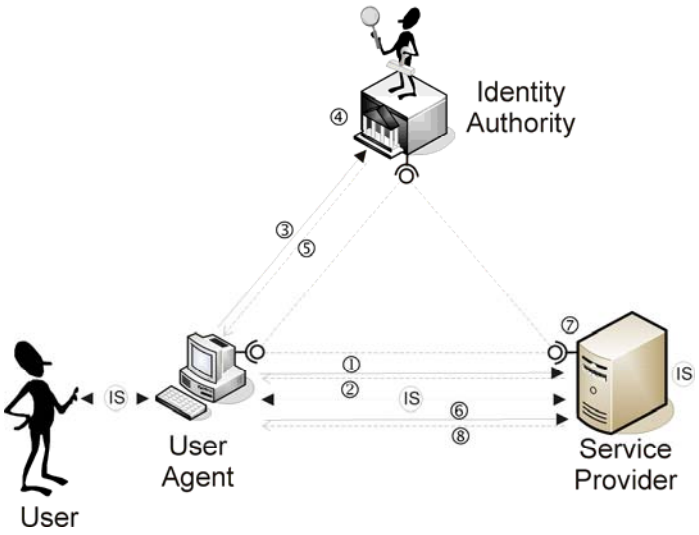

Figure 1: Abstract Federated Identity Management Model

Within this model, there are the following actors that are connected via some standardized communication interface:

- **User (U) with User Agent (UA)**

  The *User* uses a *User Agent* in order to communicate with both, the Identity Authority and the Service Provider, in order to access a service offered by the Service Provider. It is assumed that the User has a *Source Credential*[1] that is recognized by the Identity Authority and optionally also by the Service Provider.

- **Service Provider (SP)**

  The *Service Provider* offers some service to the User. Before access to this service is granted, the User needs to be authenticated. In the case that the Service Provider recognizes the Source Credential, the authentication can be performed directly by the Service Provider. Otherwise, the authentication needs to be delegated to the Identity Authority that is trusted by the Service Provider.

- **Identity Authority (IA)**

  The *Identity Authority* validates the Source Credential and issues a *Session Credential* to the User. The Session Credential is then validated by the Service Provider as a prerequisite of grating access.

---

[1] According to the broad definition in **[ModTerm]** "a *credential* is a piece of information attesting to the integrity of certain stated facts". In our specific case the "facts" may in particular be (identity) attributes of the User and Credential may be a pair of user-ID and password, a certificate according to **[X.509]** or **[ISO7816-8]**, a SAML-token according to **[SAML-v1.1]** or **[SAML-v2.0]** or any other signed security token according to **[WS-Security-1.1]**, such as a Kerberos-ticket according to **[RFC1510]** for example.

Based on these definitions, a *Federated Identity Management System* can be viewed as a sequence of entities that transform a Source Credential of a User into a Session Credential that is consumable by a Service Provider who makes a decision on whether to grant access to a sensitive service.

Before the User is able to use the service offered by the Service Provider, the following steps are typically performed:

1. **UA → SP:** The User Agent contacts the Service Provider and requests some service.
2. **SP → UA:** The Service Provider answers the request, usually including information about supported protocols and Identity Authorities.
3. **UA → IA:** The User Agent connects to the Identity Authority in order to authenticate with the Source Credential and request a Session Credential that can be presented to the Service Provider.
4. **IA:** The Identity Authority authenticates the User based on the Source Credential or alternatively uses an existing session in which authentication has already been performed previously.
5. **IA → UA:** If the authentication was successful, the Identity Authority returns a Session Credential to the User Agent.
6. **UA → SP:** The User Agent sends the Session Credential received from the Identity Authority to the Service Provider.
7. **SP:** The Service Provider validates the Session Credential and verifies the access rights of the now authenticated User.
8. **SP → UA:** The Service Provider serves the requested resource to the User Agent.

Furthermore, there may be additional steps for Identity Selection (IS), in which the User, the User Agent, and/or the Service Provider interact in order to select an appropriate electronic identity and hence Identity Authority to be contacted in step 3.

2.2 Review of existing approaches

In this section, we will summarize the most important existing approaches of identity federation and Single Sign-On, referencing the abstract model presented in Section 2.1. This section is structured as follows: In Section 2.2.1, we will recall the different profiles defined within the Liberty Alliance project and Section 2.2.2 will sketch the main processes within the WS-* specifications.

2.2.1 Liberty Alliance Profiles

The current version of the Liberty Alliance specifications contains three different profiles for identity federation and Single Sign-On:

- Liberty Artifact Profile **(LA)** (cf. **[LA-ID-FF-B&P]**, Section 3.2.2)

- Liberty Browser Post Profile **(LBP)** (cf. **[LA-ID-FF-B&P]**, Section 3.2.3)

- Liberty-Enabled Client and Proxy Profile **(LECP)** (cf. **[LA-ID-FF-B&P]**, Section 3.2.4)

In the following, we briefly describe how the eight steps of the abstract model of federation apply to the three Liberty profiles:

1. **UA → SP:** The User Agent contacts the Service Provider by sending some HTTP-request. Optional LECP-support would be indicated by a specific Liberty-enabled HTTP-header.

2. **SP → UA:** The Service Provider answers with a HTTP-response that contains a `<Lib:AuthnRequest>`-element (cf. Section 3.2.1.1 of **[LA-ID-FF-P&S]**). In case of LA or LBP this element is encoded according to Section 3.1.2 of **[LA-ID-FF-B&P]** and returned within an HTTP 302 redirect or a HTML form post, respectively. In the case of LECP, this element is wrapped in a `<Lib:AuthnRequestEnvelope>`-element (cf. Section 3.2.3.1 of **[LA-ID-FF-P&S]**) which may contain a `<Lib:IDPList>` compiled by the Service Provider, such that the User (Agent) can select the Identity Authority to be contacted in the next step.

3. **UA → IA:** The User Agent sends the `<Lib:AuthnRequest>`-element to the Identity Authority (called Identity Provider in Liberty-terminology). In case of LA and LBP, this element is simply transported within an HTTP GET request. In case of LECP, the full `<Lib:AuthnRequestEnvelope>`-element is sent to the Identity Authority using SOAP **[SOAP-v1.1]**.

4. **IA:** In absence of a previously established authenticated session, the Identity Authority authenticates the User. The details of the authentication step are not defined or constrained by the Liberty Alliance specifications.

5. **IA → UA:** Upon successful authentication, the Identity Authority returns a Credential to the User Agent. The type of the credential depends on the profile and is described in the sequel:

   - **LA**
     In case of LA, the Identity Authority returns a SAML artifact[2] to the User Agent.
   - **LBP**
     In the case of LBP, the Identity Authority returns a `<lib:AuthnResponse>`-element (cf. Section 3.2.2.1 of **[LA-ID-FF-P&S]**), which may contain one or more `<saml:Assertion>`-elements (cf. Section 2.3.2 of **[SAML-v1.1]**) as part of an HTML form.

---

[2] A SAML artifact is an Identity Authority specific reference to a SAML assertion. See Section 3.2.2.2 of **[LA-ID-FF-B&P]**, Section 4.1.18 of **[SAML-v1.1-B&P]** and/or Section 3.6 of **[SAML-v2.0-Bd]** for details.

- **LECP**
  In case of LECP, a `<lib:AuthnResponseEnvelope>`-element (cf. Section 3.2.4.1 of **[LA-ID-FF-P&S]**) is returned within a SOAP-response.

6. **UA → SP:** The User Agent sends the received Credential (`<lib:AuthnResponse>`-element or SAML artifact) to the Service Provider.
7. **SP:** In this step, the Service Provider needs to validate the received Credential. In case of LA it will first use the artifact to obtain the corresponding `<saml:Assertion>` from the Identity Authority using the SAML protocol (cf. Section 3 of **[SAML-v1.1]**). Then, it will verify the signature contained in the `<saml:Assertion>` which in turn may require the validation of a chain of certificates.
8. **SP → UA:** On success, the Service Provider finally returns an HTTP-response that contains the requested resource.

In case of LA or LBP, users lack the possibility of selecting one of their identities (i.e., Identity Authorities). This is because the Service Provider selects the Identity Authority when issuing the HTTP-redirect, while User Agents execute the redirect passively without any possibility of selection or intervention.

For the LECP profile, in step 2, the Service Provider includes a `<lib:IDPList>`-element that can be used for an explicit Identity Selection step performed by the User (Agent).

2.2.2 WS- / Microsoft CardSpace*

The sequence of protocol steps within WS-* with Microsoft's CardSpace system **[MS-CardSpace]** (see also **[MS-CS-TechRef]**, **[AlMi08]** and **[BSB08]**) is described in the following:

1. **UA → SP:** The User Agent sends an HTTP GET request to the Service Provider (called Relying Party in CardSpace-terminology) to access some login page.
2. **SP → UA:** The Service Provider returns the HTML-code of the login page to the User Agent. That CardSpace is to be used for the authentication is indicated by CardSpace-specific object tags within the returned HTML-code.
3. **UA → IA:** The User Agent connects to the Security Token Service (STS) of the Identity Authority using **[WS-Mex(v1.1)]** and **[WS-Trust(v1.3)]** in order to authenticate and request a Security Token, which is appropriate for the specific Service Provider.
4. **IA:** The Identity Authority may authenticate the User in this step if this has not happened before.
5. **IA → UA:** On successful authentication, the Identity Authority returns the Security Token to the User Agent.
6. **UA → SP:** The User Agent sends the Security Token received from the Identity Authority to the Service Provider using **[WS-Trust(v1.3)]**.

7. **SP:** The Service Provider validates the Security Token.
8. **SP → UA:** The Service Provider returns the requested resource to the User Agent.

Furthermore, there are the following additional steps for Identity Selection (IS):

Between step 2 and 3, the User Agent retrieves the applicable policy from the Security Token Service of the Service Provider using **[WS-SecPol(v1.1)]**. Using this information, the User selects an appropriate electronic identity and implicitly an Identity Authority and the type of token for the request in step 3. If the policy of the Service Provider allows self-issued Security Tokens, the identity selection step may entirely be performed within the Identity Authority.

2.3 Discussion of Advantages and Disadvantages

The LA and LBP profiles are designed to provide single sign-on functionality for User Agents that are "dumb" standard web browsers. In this case, the User is not involved in the selection of a suitable electronic identity and thus Identity Authority and hence these profiles fail to support User-centric Identity Management. Even worse, since Security Tokens are usually accessible through the DOM tree of a document, and since those protocols rely heavily on the Domain Name System (the basis of URL addressing) and the browser's Same Origin Policy, the proposed protocols may be susceptible to attacks like Cross Site Scripting (XSS) or Pharming (DNS spoofing) (cf. Section 2.4). Note that a naïve approach, which uses TLS in server authenticated mode only, can not prevent this.

For the implementation of User-centric solutions, the LECP profile and the CardSpace approach seem to be preferable. While both profiles require specific Liberty- or CardSpace-enabled User Agents it would be possible to support both systems with a single User Agent implementation (cf. **[AlMi08]**).

Unfortunately, in *all* four existing cases, the Service Provider needs to be either Liberty- or CardSpace-enabled. It would be highly preferable to design a Federated Identity Management solution where Service Providers needn't deploy specific Liberty- or CardSpace-enabled identity management software, but can simply continue to use standard, off-the-shelf web-servers such as Apache. To solve this problem, we introduce TLS-Federation in Section 3.

2.4 Security

The security of existing approaches depends mainly on the question of whether the User Agent is a standard web browser or a more capable, but proprietary client (cf. **[PfWa03]**). In the latter case, secure systems can be modeled along the lines of Kerberos v5 **[KNT91]** - an approach that proved very successful in MS Active Directory. The most common solution today is to use standard web browsers, however, since they provide platform-independence and low deployment and maintenance costs. The literature reports a variety of attacks on browser-based approaches, including one on MS Passport **[Slem01],** a security analysis of the SAML single sign-on browser/artifact profile **[Groß03]** and the recent attack against CardSpace **[GSX08].** This seems to suggest that it is especially hard or even impossible to secure browser-based approaches.

One reason for the reported security problems can be found in the fact that (nearly) all data sent through a browser is stored in the Document Object Model (DOM) of the browser. The DOM is only protected by the Same Origin Policy (SOP) which can easily be subverted by spoofing of Domain Names, IP addresses, and eventually TLS server certificates. All the data is thus accessible to malicious scripts, a weakness that provides fertile ground for attacks like Cross Site Scripting (XSS) (cf. **[GHP07]**) or Pharming (cf. **[SRJ06]**). Note that Microsoft has acknowledged this problem by introducing `HTTP-Only-`cookies, which are not stored in the DOM of IE 7 **[MS07]**.

Another reason lies in the fact that the cryptographic capabilities of the browser are restricted to TLS (handshake and record layer). TLS, however, only protects data during transmission, not when stored in the DOM. Thus, all authentication tokens issued by existing IAs fail to be associated to the legitimate browser by cryptographic means and can therefore be freely used by illegitimate browsers that steal the token using on of the attacks mentioned above.

The inherent weakness of browser based approaches can be avoided with the use of X.509 client certificates. Here, the authentication token consists of two parts: the certificate as the public value and the corresponding private key. The public value can be accessed even more easily than other DOM data, because it is transferred in clear-text during any TLS handshake[3]. It can, however, not be used by an illegitimate browser on another PC, because for successful authentication, knowledge of the private key is necessary. This private key is protected by the browser or by the operating system, and can even be protected against malware running on the user's PC by storing it on a smart card.

---

[3] Note that in **[WS-Security-1.1]** and hence in CardSpace it is possible to use X.509-based security tokens, which provide a similar level of security as TLS with client-certificates. In this case the User Agent already has or generates a key pair such that claims issued by an STS are linked to the legitimate owner of the private "proof key".

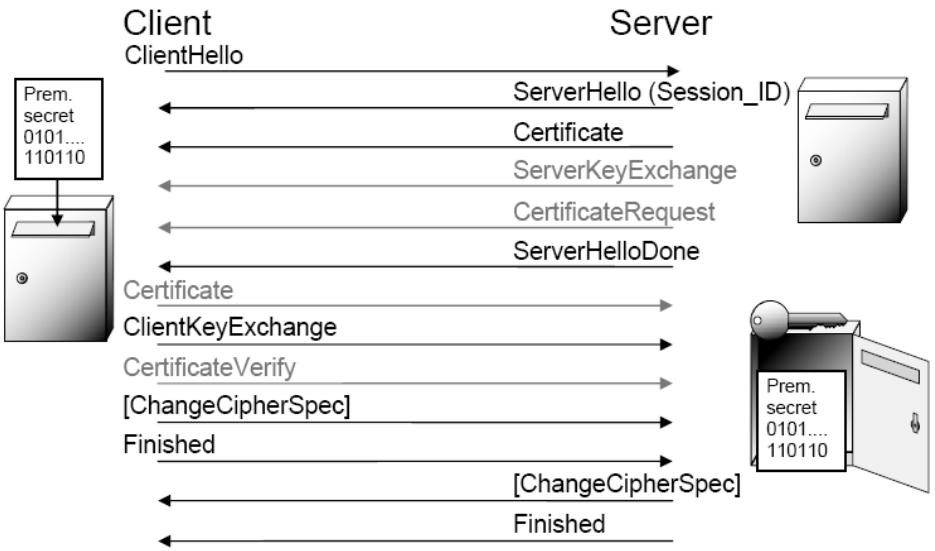

Fig. 2: In the TLS handshake, the client authenticates itself by signing a hash value of all previously exchanged messages in CertificateVerify. To be able to do so, the client must know the private key corresponding to the certificate sent in Certificate.

3 TLS-Federation

3.1 Overview

TLS-Federation is the application of the existing and ubiquitously implemented Transport Layer Security (TLS) protocol **[RFC2246]** with client-certificate authentication to a federated setting. Similarly to how Identity Authorities sign SAML-Assertions or WS-* Claims in the previously discussed approaches, the Identity Authorities in TLS-Federation use standard X.509 certificates to express their statements about user identities.

In the same way as the Liberty Alliance protocols and WS-* specifications are defined independently of the type of Source Credential to use, an Identity Authority in TLS-Federation is unlimited in its choice of authentication methods. Possibilities include the use of X.509 credentials (soft certificates or secure tokens) themselves, as well as username/password, one time passwords on paper or created by hardware tokens, non-X.509 tokens such as those used by Austrian Citizen Cards, etc. Really, like in Liberty Alliance and WS-*, any current or future authentication technology can be equally supported.

It is even possible that a TLS-Federation Identity Authority uses other federation solutions such as Liberty Alliance or WS-* to authenticate users. This means that a TLS-Federation Identity Authority can act as "translation point" that renders credentials from different federation solutions accessible to service providers who support solely the ubiquitous TLS client-cert authentication as has been included out of the box with every standard web server for years and is based on an exceptionally stable, secure (cf. **[MSW08]**), tested, and widely deployed IETF standard. TLS-Federation can also be seen as a very light-weight and simple meta-layer that integrates all possible national choices that span all possible token types both through direct access or use of existing federation solutions (Liberty, WS-*).

Also trust management in a TLS-Federation is comparable to that of Liberty Alliance and WS-*; in a "circle of trust", a Service Provider needs to know which Identity Authorities it can trust and needs to validate the certificate received as Session Credential. One possible difference is that a certificate, unlike a SAML-Assertion or a WS-* Claim can be revoked. This is motivated by the relatively long validity period of common X.509 certificates compared to the session-only life time of SAML assertions and WS-* Claims. In TLS-Federation, is is possible to either use very short-lived certificates without checking revocation status or the usual long-lived hardware-based credentials that require verification of revocation status.

TLS client-certificate authentication is usually not associated with the concept of Single Sign-On, as every time a User navigates from one Service Provider to another, its identity has to be newly established by a TLS-handshake and proven through the signature of a different challenge. We nevertheless consider TLS as being a Single Sign-On solution as it provides the same user experience: Repeated execution of a TLS-handshake for maximum security is indeed transparent for Users since they need to log in to the "Identity-providing entity"[4] only once at the beginning of a Single Sign-On session.

For a better understanding of how TLS can be applied to a federated setting, the various steps of the abstract model of federation provided in section 2.1 are described in the following:

1. **UA → SP:** The User Agent sends an HTTP GET request to the Service Provider to access a page protected by TLS. Users recognize such protected areas as HTTPS URLs. If a clear-text HTTP GET request is used, TLS can be enabled by sending a HTTP REDIRECT status code pointing to a HTTPS URL. The Service Provider has enabled the TLS option to request a client-certificate for authentication of the User.

[4] This statement not only applies to Identity Authorities but also for authentication with secure tokens like smartcard-based eIDs where a single login with PIN may be valid for an unlimited number of signatures of challenges and is interrupted only when the token is removed from the card terminal.

2. **SP → UA:** Using the standard TLS protocol as implemented in the major browsers, the Service Provider informs the browser on which Identity Authorities it considers trustworthy. This happens in the transport layer, even before the Service Provider receives the user's HTTP request. Service Providers typically configure this simply by populating their trust store with certificates of trusted IAs.

3. **UA → IA:** All major browsers ask users in a dialogue which of the available certificates shall be presented to the Service Provider. The user decision determines which Identity Authority is to be used. Note that different implementation options of TLS-Federation differ in their timing of when the Identity Authority is contacted (see Section 3.2 for details on a possible implementation). In particular, as a first option, it can be contacted before contacting the Service Provider such that the browser already has a key pair and certificate ready for use; or as a second option, the browser can interact with the Identity Authority using some middleware[5] that creates key pairs and requests certificates on the fly. The difference is solely in timing and not in essence.

4. **IA:** The Identity Authority authenticates the User in this step if this has not already happened before. Note that for this purpose a standard X.509 certificate according to **[RFC3280]** or any other token type could be used.

5. **IA → UA:** On successful authentication, the Identity Authority returns the Session Credential, an X.509 certificate, to the User Agent. The certificate may contain extensions (e.g. those defined in **[RFC3281]**) which transport identity and/or role/authorization information to the Relying-Party. In this case the functionality requested from the Identity Authority is just that of an ordinary Certification Authority that replies to a certification request. Alternatively, if Relying-Parties would prefer to receive the identity information in form of XML-based SAML-Assertions, the Identity Authority could simply include the SAML-Assertion into a certificate-extension in order to realize a secure SAML-binding in the spirit of **[GLS08]**.

6. **UA → SP:** Through the standard mechanisms of the TLS-handshake, the User Agent sends the received certificate to the Service Provider.

---

[5] Browsers have their standard APIs for such middleware. Currently PKCS#11 is supported by the Mozilla family of browsers, the Microsoft CSP-interface is supported by the Internet Explorer and emerging standards such as ISO/IEC 24727-3 and CEN 15480-3 may provide a common interface in the long term. Such middleware is commonly used to interact with locally connected hardware tokens but it is conceptually equivalent if the communications used to receive the Session Credential reaches a remote Identity Authority instead of a local token and if a local PIN entry is replaced by the authentication to a remote Identity Authority.

7. **SP:** The Service Provider validates the presented Credential. For this purpose, the Service Provider sends a challenge to the UA to request a proof that the certificate has been presented by its legitimate owner. This is comparable to the “proof key” in WS-* and is a mandatory and integral part of the TLS handshake. The Service Provider further validates the IA's signature, the validity period, the revocation status of the IA-certificate and optionally (see discussion above) the revocation status of the session credential. The necessary functionality is available in all major web servers.

8. **SP → UA:** The Service Provider, now that the authentication is completed, determines whether the user has access to the requested resource/service (authorization) and either provides the requested resource or presents a page informing about the rejection.

3.2 A Possible Implementation of TLS-Federation

While in the long run, an implementation of TLS-Federation as an eID-middleware component, possibly with a standard API as defined by **[CEN15480-3],** would possibly be preferable, the following describes a straight-forward implementation option that uses the browser's native capability of generating and storing key pairs, issuing certification requests, and inserting certificates received in response in the native certificate store. While this functionality is available in proprietary form in all major browsers, for simplicity, the following description is limited to the methods used by the Mozilla family of browsers; Microsoft’s Internet Explorer can easily be supported in a similar fashion.

The basic work flow is that users activate their eID by logging in to the Identity Authority with the national eID credential of arbitrary type as provided by their member state. Once activated, the user accesses the services of one or several Service Providers. For security, the eID is then deactivated at the end of the work-session even before the short-lived session credential expires.

This user experience is quite comparable to that of a smartcard-based eID in a non-federated setting. Here, users activate their eID by inserting the card into the reader and typing in their PIN at the first use. They deactivate the eID by removing the card from the reader.

The following describes the detailed steps of a user session:

1. **UA → IA:** The User navigates to the secure login page (i.e., an HTTPS URL) of the Identity Authority that requests authentication with the Source Credential (national eID of arbitrary type). Since users usually use their national eID Identity Authority, they typically keep a bookmark on this login page for ease of use.

2. **IA → UA:** The Identity Authority identifies the browser type and responds with a login page containing the browser's proprietary elements necessary for key generation and certificate signature request. In the case of Mozilla browsers, the login page contains an HTML form that contains a `<keygen>` tag.

3. **UA → IA:** On form submission, this tag instructs the browser to generate a key pair, insert the private key in its native key store, and submit a certificate signature request with the public key in SPKAC format (cf. **[SPKAC]** and **[RuSc02]**, A.5) as part of the form submission.

4. **IA → UA:** The Identity Authority receives this certificate signature request, determines the necessary content of the certificate to issue, signs the certificate, and returns it with the according MIME type to the browser. In this step, the Identity Authority has total freedom to determine what "assertions" to make on the user as part of the certificate. For example, based on a requested choice of sector of use or pseudonym by the user in one of the form fields, the Identity Authority may derive a sectorial or a pseudonym Subject Common Name. Similarly, the certificate could contain information of specific roles granted to the user or assertions about an age class or citizenship that may complement a pseudonym Subject CN.

5. **UA → IA:** The browser receives the signed certificate with the appropriate MIME type. This instructs the browser to verify whether its key store contains an according key pair and on success automatically inserts the certificate in its native certificate store. This step completes the activation of the eID by the user who can now access the resources of one or several service providers.

6. **UA → SP:** The User navigates to the page of a protected resource offered by a Service Provider. This page is under an HTTPS URL and the server requests the optional client-certificate-authentication. The TLS handshake automatically indicates to the browser which certificates, and thus Identity Authorities, are trusted by the Service.

7. **SP → UA:** As part of performing the TLS-handshake, the browser, in its standard functionality, now presents the user with a dialogue to choose the certificate (identity) to present to the server. If the Service trusts the Identity Authority to which the user has logged in previously during the activation of her eID, the User selects the previously issued certificate. To conclude the TLS-handshake, the browser also uses the previously generated private key to sign the challenge sent by the Service Provider.

8. **SP → UA:** The Server validates the presented certificate and on success delivers the requested resource.

9. **UA:** After completing the work session, for security, the user may execute a local utility program that deletes everything issued by her Identity Authority from the key and certificate store, even before their short validity period expires.

3.3 Discussion of advantages and disadvantages

As illustrated using the abstract model of federation in Section 2.1, the various federation solutions are basically equivalent from a conceptual point of view; all solutions use a set of statements that are signed by a trusted Identity Authority and compose the Session Credential. The most significant conceptual difference is whether the credential incorporates a "proof key"; i.e., a public key that is then used in a challenge-response-protocol to determine whether the presenter of the credential is its legitimate owner. TLS/X.509 always uses such "proof keys", in WS-* the use of them is optional (depending on the type of Security Token) and in SAML/Liberty "proof keys" do not seem to be supported at all. The remaining differences are limited to the choice of data formats and protocols to exchange credentials. The TLS standard **[RFC2246]** defines these protocols at a transport layer level in a way specific to the needs of highly-secure authentication. The protocols are highly stable, ubiquitously implemented and tested. Also, in TLS, there is a single protocol without major options or profiles to choose from and to support. These characteristics are crucial for high-levels of security and ease of large-scale deployment.

In Liberty Alliance/SAML, the protocols used to interact between the various parties of the federation scenario are based on HTTP (redirect of the Browser Artifact Profile), HTTP/HTML (in the Browser Post Profile), or a custom protocol (Liberty-Enabled Client), respectively. The former two are difficult to deploy in certain settings, for example that of eID interoperability in Europe. There problem here is the lack of a User-centric approach in which Users may select the identity and thus Identity Authority to use. Instead, these Liberty Alliance profiles make the assumption that Servers already know the Identity Authority chosen by a user (since otherwise they couldn't issue the redirect or form POST). Distributing up to date lists of Identity Authorities in Europe to *all* relevant Service Providers of both private and public sector at a European scale is surely a deployment issue; but beyond that, a hard-coded association of Identity Authority to User surely prohibits the application of privacy-friendly and User-centric solutions where Users need to make choices on the identity and personal data to present to a specific service.

These shortcomings apply only to the first two Liberty Profiles and it seems that the third profile was conceived precisely with the objective to overcome them. While it surely succeeds in resolving both the security and User-centricity issues, we see a major problem in the fact that all current browsers lack support for that profile.

We believe that the objective of ubiquitous browser support is rather difficult to achieve considering that one of the major browser vendor supports a competing federation solution and is not likely to help the marketing/deployment of Liberty-Enabled Client solution. Ubiquitous support of the Liberty Protocols is further hindered by the fact that there is still a rapid evolution of standards of Liberty and SAML and that the lowest common denominator among all players may well be behind the latest version.

Another issue of the Liberty Alliance solution is that the choice of three profiles makes it less likely that all Service Providers and Identity Authorities actually support the only profile option that provides the required security and User-centric properties.

The approach of a User-controlled identity agent like Microsoft CardSpace or Higgins together with WS-* seems to be a more promising approach in the long term, because it has a User-centric design and allows to use “proof keys” to reach a high level of security.

But the success of this approach requires a large-scale deployment of new components both on the User's desktop, the Service Provider, and the Identity Authority. Currently, on the desktop, the significant roll-out seem to be limited to recent Microsoft platforms and support by Service Providers still seems to be rather sparse. It is to be seen whether the recently launched Identity Card Foundation **[ICF]** that is supported by some major players and has the objective of deploying the approach at large scale will prove to be successful.

On the side of government Identity Authorities in Europe, judging for example from the recent Common Specification for interoperable eID **[IDABC-CS]** or the STORK Project **[STORK]**, it does not seem that the adoption of WS-*/CardSpace is currently considered as one of the valid options. Much rather it seems that the choice of federation solution (Liberty, SAML 2.0, WS-*, no federation at all) is left to each Member State and that the interoperability of eIDs in Europe is planned to be achieved by a “neutral” layer above the national choices. Indeed, TLS-Federation may well be the most light-weight and straight-forward solution to realize this “pan-European Identity Layer”.

While Liberty/SAML federation solutions have been successfully deployed in corporate settings, we believe that different federation settings such as eID interoperability in Europe may impose different requirements and possibly favour different solutions. In particular, in the very complex environment of a large number of mostly governmental Identity Authorities and a very large number of Service Providers from all sectors and many countries, practical deployment issues may be one of the factors that determine between success and failure.

To achieve a consensus on the protocol, protocol version, and protocol profile to use for interoperable services among so many players who act in so many jurisdictions is a major undertaking that illustrates that the organizational/political complexity of federation may significantly outweigh the technical complexity. Surely this is a significant difference to corporate settings.

From this point of view, we believe that the more conservative the choice of protocol, the easier it is to successfully deploy a federation solution at this unprecedented scale. We therefore see great merit in TLS-Federation that uses mature, long-standing and stable protocols that are already ubiquitously implemented and tested in all major browsers and web servers. Beyond the fact that unlike the other solutions, TLS-Federation does not require the roll-out of new software components at the various players, it can out of the box integrate all possible national choices of eIDs, ranging from the use of X.509 tokens used directly without federation, over privacy-enhanced solutions like the Austrian Citizen Card, to national deployments of Liberty Alliance, SAML 2.0, or WS-* federations.

A second major difference between federation solutions is their choice of session credentials. In Liberty/SAML and WS-*, the choice is largely different from the technology of user-credentials used to authenticate to the Identity Authority. In contrast, in TLS-Federation the technology chosen for the session credential coincides with the technical choice made for a large majority of national eIDs; both are X.509.

The consequence of using the same credential format is that in countries that use X.509 credentials (soft certificates like Malta or smartcards like Belgium, Estonia, Finland, Spain, Italy, etc.), users can participate in TLS-Federation *without* the need of a national Identity Authority. The eID token can be seen as a local, autonomous Identity Authority and not surprisingly, the functionality and interfaces to the browser can be identical between a remote IA and a local hardware token.

4. Conclusions and outlook

In the current paper we briefly analysed different existing approaches for Federated Identity Management and introduced a novel approach called “TLS-Federation” which allows easy deployment both, by typical Relying Parties and X.509-affine communities and member states that already have adopted one of the other federation solutions. Furthermore it is advantageous from a security perspective, because it is immune against important classes of web-attacks, such as XSS or Pharming and hence it seems to be a very interesting alternative to the other approaches for Federated Identity Management based on SAML or WS-*. This is particularly true in the context of the forthcoming large scale pilot project **[STORK]**, which aims at the cross-border recognition of national eID-cards in Europe.

References

[AlMi08] W. A. Alrodhan, C. J. Mitchell: *A Client-side CardSpace-Liberty Integration Architecture*, IDtrust 2008, 7th Symposium on Identity and Trust on the Internet, 04.-06.03.2008, http://www.isg.rhul.ac.uk/cjm/acscsl.pdf

[BSB08] V. Bertocci, G. Serack, C. Baker: *Understanding Windows CardSpace – An Introduction to the Concepts and Challenges of Digital Identities*, AddisonWesley, 2008

[CEN15480-3] Comité Européen de Normalisation (CEN): *Identification card systems - European Citizen Card - Part 3: European Citizen Card Interoperability using an application interface*, CEN 15480-3 (Working Draft), 2008

[Groß03] T. Groß: *Security analysis of the SAML single sign-on browser/artifact profile*. In Annual Computer Security Applications Conference. IEEE Computer Society, 2003, http://www.acsac.org/2003/papers/73.pdf

[GHP07] J. Grossman, R. Hansen, P. D. Petkov: *Cross Site Scripting Attacks: XSS Exploits and Defense*, Syngress Media, 2007

[GLS08] S. Gajek, L. Liao, J. Schwenk: *Two New TLS Bindings for SAML and SAML Artifacts*, to appear, 2008

[GSX08] S. Gajek, J. Schwenk, C. Xuan: *On the Insecurity of Microsoft's Identity Metasystem*, Ruhr University Bochum, Technical Report TR-HGI-2008-003, http://www.nds.rub.de/gajek/papers/GaScXu08_CardSpaceTR.pdf

[Herr07] W. Herrmann: *Gartner's most important IT-Trends*, (in German), http://www.computerwelt.at/detailArticle.asp?a=110421&n=2

[Higgins] Eclipse Foundation: *Higgins Open Source Identity Framework*, http://www.eclipse.org/higgins/

[ICF] Information Card Foundation: *Information Card Foundation Website*, http://informationcard.net/

[IDABC-CS] IDABC: *Common specifications for eID interoperability in the eGovernment context*, http://ec.europa.eu/idabc/servlets/Doc?id=30989

[ISO7816-8] ISO/IEC 7816-8: *Identification cards – Integrated circuit cards – Part 8: Commands for security operations*. International Standard, 2004

[KNT91] John T. Kohl, B. Clifford Neuman, Theodore Y. Ts'o, *The Evolution of the Kerberos Authentication Service*. Proc. 1991 EurOpen Conference, Tromsø, Norway, 1991

[LA-Specs] Liberty Alliance Project: *Specifications*, via http://www.projectliberty.org/specifications__1

[LA-ID-FF-B&P] Liberty Alliance: *Liberty Alliance ID-FF Bindings and Profiles Specification*, Version 1.2, Errata v2.0, within http://www.projectliberty.org/liberty/content/download/1266/8160/file/liberty-idff-1.2-20050520.zip

[LA-ID-FF-P&S] Liberty Alliance: *Liberty Alliance ID-FF Protocols and Schema Specification*, Version 1.2, Errata v3.0, within http://www.projectliberty.org/liberty/content/download/1266/8160/file/liberty-idff-1.2-20050520.zip

[MS07] Microsoft Inc.: *Mitigating Cross-Site Scripting with HTTP-only Cookies*, http://msdn.microsoft.com/en-us/library/ms533046.aspx, 2007

[MS-CardSpace] Microsoft Inc.: *Windows CardSpace*, via http://netfx3.com/content/WindowsCardspaceHome.aspx

[MS-CS-TechRef] Microsoft Inc.: *A Technical Reference for InfoCard v1.0 in Windows*, August, 2005, http://download.microsoft.com/download/5/4/0/54091e0b-464c-4961-a934-d47f91b66228/infocard-techref-beta2-published.pdf

[MSW08] P. Morrissey, N.P. Smart, und B. Warinschi: *A Modular Security Analysis of the TLS Handshake Protocol*, Cryptology ePrint Archive, Report 2008/236, http://eprint.iacr.org/2008/236 , 2008.

[ModTerm] Modinis IDM Study Team: *Common Terminological Framework for Interoperable Electronic Identity Management*, Modinis Study on Identity Management in eGovernment – Consultation Paper, Version 2.01, via https://www.cosic.esat.kuleuven.be/modinis-idm/twiki/pub/Main/GlossaryDoc/modinis.terminology.paper.v2.01.2005-11-23.pdf

[PfWa03] B. Pfitzmann, M. Waidner: *Analysis of liberty single-sign-on with enabled clients*, Internet Computing, IEEE, Vol. 7, Issue 6, Nov.-Dec. 2003, pp. 38- 44

[RFC1510] J. Kohl: *The Kerberos Network Authentication Service (V5)*, IETF RFC 1510, via http://www.ietf.org/rfc/rfc1510.txt

[RFC2246] T. Dierks, C. Allen: *The TLS Protocol*, Version 1.0, IETF RFC 2246, via http://www.ietf.org/rfc/rfc2246.txt

[RFC3280] R. Housley, W. Polk, W. Ford, D. Solo: *Internet X.509 Public Key Infrastructure - Certificate and Certificate Revocation List (CRL) Profile*, IETF RFC 3280, via http://www.ietf.org/rfc/rfc3280

[RFC3281] S. Farrell, R. Housley: *An Internet Attribute Certificate Profile for Authorization*, IETF RFC 3281, via http://www.ietf.org/rfc/rfc3281

[RuSc02] U. Rutishauser, A. Schäfer: *Open reference implementation of a SCEP v2 client*, http://www.urut.ch/scep/scepclient.pdf

[SAML-v1.1] S. Cantor, J. Kemp, R. Philpott, E. Maler: *Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V1.1*, OASIS Standard, 02.09.2003, http://www.oasis-open.org/committees/download.php/3406/oasis-sstc-saml-core-1.1.pdf, 2003

[SAML-v1.1-B&P] E. Maler, P. Mishra, R. Philpott: *Bindings and Profiles for the OASIS Security Assertion Markup Language (SAML) V1.1*, OASIS Standard, 02.09.2003, http://www.oasis-open.org/committees/download.php/3405/oasis-sstc-saml-bindings-1.1.pdf , 2003

[SAML-v2.0] S. Cantor, J. Kemp, R. Philpott, E. Maler: *Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V2.0. OASIS Standard*, 15.03.2005. http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf, 2005.

[SAML-v2.0-Bd] S. Cantor, F. Hirsch, J. Kemp, R. Philpott, E. Maler: *Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0*, OASIS Standard, 15.03.2005, http://docs.oasis-open.org/security/saml/v2.0/saml-bindings-2.0-os.pdf

[Slem01] M. Slemko: *Microsoft passport to trouble*, 2001. http://alive.znep.com/marcs/passport/

[SOAP-v1.1] D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. F. Nielsen, S. Thatte, D. Winer: *Simple Object Access Protocol (SOAP) 1.1*, W3C Note, 08 May 2000, via http://www.w3.org/TR/2000/NOTE-SOAP-20000508

[SPKAC] OpenSSL Project: *SPKAC printing and generating utility documentation*, http://www.openssl.org/docs/apps/spkac.html

[SRJ06] Sid Stamm, Zulfikar Ramzan, and Markus Jakobsson. *Drive-by pharming*. Technical report, Indiana University Computer Science Technical Report number 641, December 13, 2006

[STORK] STORK-Consortium: *STORK project website*, http://www.eid-stork.eu/

[WS-Federation] C. Kaler, A. Nadalin:: *Web Services Federation Language (WS-Federation)*, Version 1.1 December 2006, http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-fed/WS-Federation-V1-1B.pdf

[WS-Mex(v1.1)] F. Curbera, S. Parastatidis, J. Schlimmer. *Web Services Metadata Exchange (WS-MetadataExchange)*, Version 1.1, August 2006. http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-mex/metadataexchange.pdf, 2006

[WS-SecPol(v1.1)] C. Kaler, A. Nadalin: *Web Services Security Policy Language (WS-SecurityPolicy)*, July 2005, Version 1.1, http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-secpol/ws-secpol.pdf , 2005.

[WS-Trust(v1.3)] A. Nadalin, M. Goodner, M. Gudgin, A. Barbir, H. Granqvist: *WS-Trust 1.3*. OASIS Standard, 19.03.2007. http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-os.pdf , 2007

[WS-Security-1.1] A. Nadalin, C. Kaler, R. Monzillo, P. Hallam-Baker: *Web Services Security: SOAP Message Security 1.1. OASIS Standard*, 01.02.2006. http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf, 2006.

[X.509] ITU-T: ITU-T Recommendation X.509:2000 - ISO-IEC 9594-8:2000. *Information technology Open Systems Interconnection The Directory: Public-key and attribute certificate frameworks*, March 2000