Journal of Theoretical and Applied Electronic Commerce Research
ISSN 0718–1876 Electronic Version
VOL 10 / ISSUE 2 / MAY 2015 / 53-76
© 2015 Universidad de Talca - Chile

This paper is available online at
www.jtaer.com
DOI: 10.4067/S0718-18762015000200005

Managing Dynamic Identity Federations using Security Assertion Markup Language

Md. Sadek Ferdous[1] and Ron Poet[2]

[1] School of Computing Science, University of Glasgow, Glasgow, Scotland, m.ferdous.1@research.gla.ac.uk
[2] School of Computing Science, University of Glasgow, Glasgow, Scotland, Ron.Poet@glasgow.ac.uk

Received 23 January 2014; received in revised form 14 September 2014; accepted 22 September 2014

Abstract

Security Assertion Markup Language is one of the most widely used technologies to enable Identity Federations among different organisations. Despite its several advantages, one of its key disadvantages is that it does not allow creating a federation in a dynamic fashion to enable service provisioning (or de-provisioning) in real time. A few approaches have been proposed to rectify this problem. However, most of them require elaborate changes of the language and do not provide mechanisms to manage federations dynamically. This paper presents a better approach based on an already drafted Security Assertion Markup Language Profile and requires no change in its specification, rather it depends on the specific implementation. Our proposed approach covers all aspects regarding the management of dynamic Identity Federation. It will allow users to create federations dynamically between two prior unknown organisations and will allow them to manage such federations as long as it is required. Implicit in each identity federation is the issue of trust. Therefore, the trust issues involved in the management of dynamic federations are analysed in details. Finally, a proof of concept is discussed with a few use-cases 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 web-enabled (or online) services in the last decade or so, the way those services can be accessed has changed considerably. Many of those services require that the users must register themselves with the Service Provider (SP, in short; organisations that provide online services) to access its services. This is required to provide users with more personalised services as well as to ensure accountability at the SP. Users are issued with digital identities consisting of identifiers and related credentials which users need to present while accessing such services. As the number of web-enabled services as well as the user-base was expanding rapidly, more and more identities and credentials were issued, and soon their management became challenging, both for service providers 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, in short). Initially, these systems were not interoperable, meaning digital identities in one system was not recognised by others. As a matter of fact, interoperability was not considered to be important as different organisations managed their own user-bases by themselves in-house.

However, as the landscape for web and web-based services started to change, novel business scenarios started to emerge. One prominent example of such scenarios was the cooperation between disparate organisations to provide conglomerated services among business partners. To facilitate such collaborations, organisations felt the necessity to expand their service bases not only to an ever-growing number of users within the organisation but also to users from their business partners. This need gave rise to a novel model of Identity Management known as the Federated Identity Management (FIM, in short) [5], [6]. The FIM offers a flexible and secure way to establish Identity Federation (also known as Federated Identities or Federation of Identities) among organisations from different security domains. This allows organisations to reduce complexities in sharing their protected online resources with users of other organisations that are part of the same identity federation as well as offers much improved user experience by reducing the number of identifier/credential pairs that the users need to remember. There are several core technologies available that allow the creation of federations such as Security Assertion Markup Language (SAML) [23], OpenID [28], WS-Federation [27], etc.

Among them, OpenID is merely an authentication protocol where federations can be achieved in a limited way. With the use of its Attribute Exchange extension, attributes can be exchanged among different organisations [29]. Even though OpenID is a very popular authentication protocol for web-enabled services, it is not favoured by many organisations due to its trust everyone policy that virtually asks every organisation to trust other organisations [3]. A new extension called PAPE (Provider Authentication Policy Extension) [30] was proposed in 2008 for enforcing trust mechanisms in a very limited way. Therefore it is safe to say that trust, security and privacy issues have not been addressed thoroughly in the OpenID as of yet.

WS-Federation is a federation model with the aim to simplify the cross-domain service access and interaction among several parties. It is a part of the Web Service Protocol Stack (WS-*) which includes WS-Security, WS-SecurityPolicy, WS-MetadataExchange, WS-Trust, etc. Utilising these protocols, WS-Federation allows the creation of a federation that is based on strong trust assumptions and satisfies reasonable amount of security and privacy requirements [7]. Windows CardSpace was based on WS-Federation [36] and was thought to be the driving force for widespread adoption of the WS-Federation. With the demise of CardSpace, the widespread adoption of WS-Federation is uncertain and also in reality it has not been widely used to deploy a federation.

On the other hand, SAML has been the most widely used technology for deploying federations in scenarios when there is a need of a federation that requires strong trust assumptions as well as maintains good security and privacy properties. This is why it is favoured as the pick of the technologies by many educational institutes and Governmental web-enabled services all over the world. Despite its several advantages, one major constraint is the method by which trust is established and maintained in the SAML. To enable federations using SAML, trust among participating organisations has to be pre-configured by system administrators. 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 either require a considerable amount of change of the SAML protocol or only provide mechanisms for creating federations in a semi-automated fashion. And also the trust issues involved while creating federations in a semi-automated fashion are not thoroughly examined.

To rectify the stated problems a better mechanism was proposed to create federations in a fully automatic dynamic fashion in our previous work [8]. In that work, the key concepts behind dynamic federation were formally defined, trust issues in such scenarios were explored and a proof of concept to illustrate the applicability of the proposal was described.

This paper presents an extension of our previous work. Here, it has not only been considered how a dynamic federation can be created, but also have been investigated how such federations can be managed dynamically. The contributions of the current paper over our previous work are:

- A new Section to introduce the key concept of Identity and Federated Identity Management has been added. The term Identity Management is not well defined. Therefore, a more concrete definition of Identity Management has been provided in that section.

- The notion of the life-cycle of the traditional SAML federation encompassing the whole spectrum of the management of federations has been introduced. Then the notion with respect to the dynamic identity federation has been analysed.

- It has been discussed how the feature of the life-cycle of identity federation has been incorporated into our proposal using a few use-case scenarios.

- Novel features have been added to allow users to manage dynamic federations where one of the entities may reside in the local machine from where the user is accessing the service. This feature can be used to federate locally installed IdPs in dynamic fashion.

With this introduction, this paper is organised as follows. Section 0 provides a brief introduction on some key concepts of Identity and Federated Identity Management. Some existing works are discussed and heir strengths and weaknesses are analysed in Section 3. Then, some key concepts of Dynamic Federation are defined and the trust issues are explored in Section 4. The developed proof of concept is discussed for two different scenarios in Section 5 and in Section 6. The strengths and weaknesses of our implementation as well as possible future works are presented in Section 7. Finally, the paper is concluded in Section 9.

2 Background

In this section a brief introduction on Identity, Identity Management, Federated Identity Management and SAML is presented. In addition, the trust issues involved in a federation are discussed and how these issues are addressed in SAML are analysed.

2.1 Identity, Identity Management & Other Related Topics

A physical or logical object with a separate and distinctive existence, physically or logically, can be denoted as an Entity [10]. It has different interpretations in different disciplines. In the context of this paper it represents a person, an organisation or a machine (computer) operated by persons or organisations. Identity is the fundamental property of an entity that declares the uniqueness of that entity and helps to differentiate it from other entities [10]. It is by this property everything around us physically and virtually are identified. However, an entity can be identified by separate identities in separate environments or contexts. Each of these identities is usually known as a partial identity. Each partial identity may consist of a set of characteristics which are known as attributes. Among these attributes, the one which is used to uniquely identify an entity within a context is known as the Identifier. Social Security Number or national ID number in a country and a user-id in an application scenario, email addresses, etc. are examples of Identifiers.

Before looking at different aspects of Identity Management, a definition of the term *Identity Managemen*t would be useful. Surprisingly, a large variation of definitions of Identity Management exists and can be found in [9], [13], [17], [22], [31]. These definitions are simple to understand, unfortunately they only focus on the identification, representation and management of identities and do not cover all the aspects (which will be described shortly) of Identity Management. The definition provided in the ITU-T X.1250 Recommendation is the closest in spirit to the definition according to our understanding [14]. Unfortunately, it also fails to capture some other aspects of Identity Management that have been captured in the definitions mentioned in the beginning. It seems that none of these definitions can capture the full notion of identity management, however, a comprehensive definition that satisfies all aspects of Identity Management can be created by combining different previous definitions which is provided next:

Definition 1. Identity Management - Identity Management is a set of functions and capabilities (e.g., administration, management and maintenance, discovery, communication exchanges, correlation and binding, policy enforcement, authentication and assertions) used for:

- Creation and management of identity information (e.g., identifiers, partial identifiers, credentials, attributes);

- Assurance of identity information;

- Assurance of the identity of an entity (e.g., users/subscribers, groups, user devices, organisations, network and service providers, network elements and objects, and virtual objects);

- Selection of identity information to be used for authorisation and for service provisioning; and

- Supporting business and security applications.

A system that is used for managing the user identity is called an Identity Management System (IMS). Shibboleth [12], OpenID [28], Microsoft's CardSpace [36], etc. are examples of different Identity Management Systems which use SAML, OpenID and WS-Federation technologies respectively. Each of these IMS has three different types of actors which are described next:

- Identity Provider (IdP). An IdP is responsible for managing digital identities of users and providing identity related services to different Service Providers. IdPs are also known as Asserting Parties (AP).

- Service Provider (SP). A SP is responsible to provide web-enabled services to the users based on the identity information (identifiers and/or attributes) received from the IdP. SPs are also known as Relying Parties (RP).

- User (Client). A user/client receives services from a SP.

With these three parties, each IMS, generally, follows the following steps to allow any user to manage her identity which in turn is used to access online services. This set of steps is also known as the Life-cycle of an IMS.

- Registration. It is the initial step in which a user registers herself at the respective IdP (or at SP) by providing different personal information along with an identifier and a credential.

- Identification & Authentication. Before any service can be accessed, the user needs to be identified and then authenticated. The most common form of doing this is to use the identifier (username) and the credential (password).

- Assertion/Claim Exchange. This step is required when the IdP and the SP reside in different security domains which involves exchanging identity information between the IdP and the SP using a predefined protocol. Different protocols have different names for identity information, e.g. SAML calls it the Assertion while WS-Federation calls it a Claim [23], [27].

- Authorisation. Authorisation is a process to decide if a certain action is allowed by an entity based on her identifier or attributes. Once the user is authenticated at an IdP and the assertion/claim regarding the user is transferred to the SP, the SP has to check if a particular action is allowed by this user.

- Service Provisioning. Once the user is identified, authenticated and authorised to access a particular service, she can access the service. The phase of accessing a service is known as service provisioning.

- De-registration. The final step in the Identity Management is the De-registration process which allows any user to de-register from a service that the user does not wish to access any more. The de-registration process usually removes the association between an entity and the identifier as well as deletes any related personal information from the IdP.

2.2 Federated Identity Management

The Federated Identity Management model is based on the concept of Identity Federation. In the ITU-T X.1250 recommendation, a federation is defined simply as "An association of users, service providers and identity providers" [14]. 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/or technical contract [5], [6]. 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 [6]. Single Sign On (SSO) is the capability that allows users to log in to one system and then access other related but autonomous systems without further logins. It alleviates the need to log in every time a user needs to access those related systems. A good example is the Google Single Sign On service which allows users to log in a Google service, e.g., Gmail, and then allows them to access other Google services such as Calendar, Documents, YouTube, Blogs and so on.

A federated identity domain can be formed consisting of only one IdP in an identity domain and more than one SP with each SP residing in a separate identity domain (Type 1 in Figure 1). Several identity domains can be combined to form a larger identity domain where each smaller federated domain is of Type 1 (Type 2 in Figure 1). The issue of trust is a fundamental concept in FIM as different autonomous participating organisations need to trust each other inside the federation to allow them to exchange user information and trust that information. Such parties inside a federation are said to form the so-called Circle of Trust (CoT). The dashed boundary in each figure signifies the federated identity domain, that is a single Circle of Trust and each solid circle represents a separate identity domain.

Journal of Theoretical and Applied Electronic Commerce Research
ISSN 0718–1876 Electronic Version
VOL 10 / ISSUE 2 / MAY 2015 / 53-76
© 2015 Universidad de Talca - Chile

This paper is available online at
www.jtaer.com
DOI: 10.4067/S0718-18762015000200005

That is, IdP1 and other SPs in Figure 1(a) and IdP1, IdP2 and all SPs in Figure 1(b) are part of a single Circle of Trust respectively. Federated Identity Domain 1 and 2 in Figure 1(b) represent a combined Type 2 federated domain.

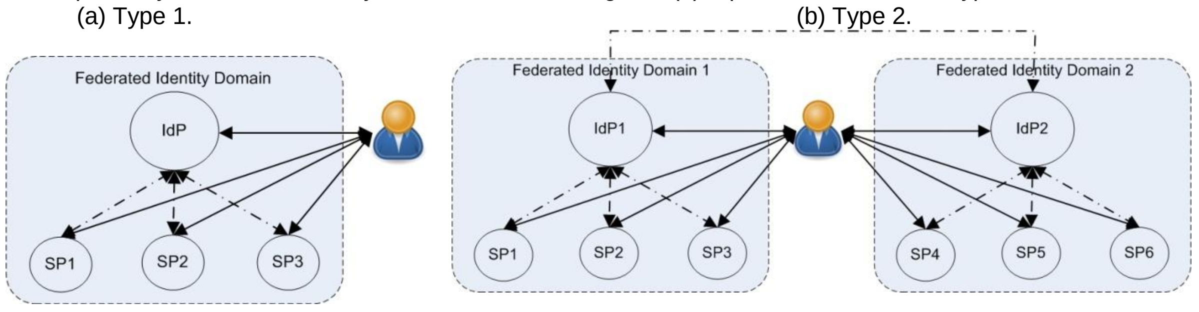

(a) Type 1.
(b) Type 2.
Federated Identity Domain
IdP
SP1
SP2
SP3
Federated Identity Domain 1
IdP1
SP1
SP2
SP3
Federated Identity Domain 2
IdP2
SP4
SP5
SP6

Figure 1: Federated identity domain

A detailed analysis of trust requirements for FIM can be found in [15]. From [15], the SP needs to trust that the IdP will 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 purpose as outlined in the agreement. Based on this trust level, users will be granted to access a service or rejected.

The FIM offers a good number of advantages to different stakeholders [5], [18]:

- The separation of duties among different organisations is the main advantage of FIM. IdPs only concentrate on managing identities and providing identity information to the SPs and SPs concentrate only on providing services based on the identity information to authenticated users. Such separation allows SPs to offload the associated cost of managing and maintaining user identities to trusted IdPs.

- FIM provides scalability in the sense that it allows SPs to offer their services not only to their user-bases, but also to users from other SPs and thus maximising the number of consumers using the same infrastructure.

- It is also attractive from the IdP’s perspective as it allows IdPs to maintain an association with end users and enables them to access an array of diversified services with the same identity.

- FIM utilises standardised approach to ensure the improved security and privacy of users and make sure that the personally identifiable information is minimally propagated and thereby reduces the risk of identity theft.

- The circle of trust can easily be expanded by integrating new application and service partners into the federation. It takes a minimal effort for any new partner to be a part of that circle as they need not to worry about managing user-base.

- It gives users the single sign on (SSO) facility which allows them to avail services from different providers without any further logins. This also reduces the need to maintain personal profiles on different locations. Thus SSO improves efficiency and enhances user experience.

- FIM also alleviates users from remembering different user-ids and their related credentials as only a few accounts at different IdPs might be enough to access different services from the SPs. In this sense, it also increases security imperatively as only a very few accounts are managed by a user; a strong password can be chosen and easily remembered.

2.3 Life-cycle of an Identity Federation

Once an entity joins a federation, it will remain there as long as the contract allows. During this time it might be required to update corresponding information of that entity inside the federation. Finally, the entity must be removed from the federation once the contract ends or the entity wants to leave the federation. These tasks are carried out at the admin level by the respective administrators and are integrated functionalities of a federation. These tasks can be combined to introduce the notion of the Life-cycle for an identity federation.

The life-cycle of an identity federation implies the steps that are required to create, revoke, maintain and to utilise a federation. The steps are:

Journal of Theoretical and Applied Electronic Commerce Research
ISSN 0718–1876 Electronic Version
VOL 10 / ISSUE 2 / MAY 2015 / 53-76
© 2015 Universidad de Talca - Chile

This paper is available online at
www.jtaer.com
DOI: 10.4067/S0718-18762015000200005

- Association (Federation). The association is the process by which a SP is added to a new or existing federation. At this step, the metadata of the SP and the IdP are exchanged and then their metadata are stored in the respective TAL (Trust Anchor List, see below) and thus the trust between these two entities are established.

- Provisioning. This is the intermediary step in which the entities (the IdP or SP) utilise the federation to offer services to the users.

- Maintenance. This intermediary step allows the admin to update any information inside the metadata of the respective entities.

- Revocation (Defederation). In this step, the SP is defederated (removed) from the federation. To do this, the IdP simply removes the metadata of the respective SP from its TAL and the SP removes the metadata of the IdP from its TAL.

These four steps essentially define the way a federation can be managed and will be used to demonstrate the applicability of our proposal for managing a dynamic identity federation in the subsequent section.

2.4 Security Assertion Markup Language (SAML)

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 (usually, IdPs) then responds with the information, so that the user can be identified and authenticated at its end. The whole language comprises of four key concepts: assertions, protocols, bindings and profiles. A SAML assertion is the declaration about a user asserted by the IdP for a service provider. In its most general form an assertion states the following [23], [35]: “The assertion A issued at time T by issuer (IdP) I about the user U as long as conditions C are valid”.

An assertion can contain three different types of statements which the SP uses for providing service to the user: i) Authentication statement is used to assert that the user has been authenticated by the asserted IdP at the specified time, ii) Attribute statement is used to specify the attributes of the user and iii) Authorisation statement is used to assert that the user is permitted a certain action on the specified resource under some conditions. The SAML protocol is used to define the rules on how to pack the SAML elements inside the request/response packet and on how to process them on receipt.

SAML binding is used to map the SAML protocol into the communication protocol such as HTTP (Hypertext Transfer Protocol) or SOAP (Simple Object Access Protocol) by which the SAML elements are transported. Simply, bindings define the mechanisms by which SAML elements are placed inside any HTTP or SOAP packet. A SAML profile combines all of the previously mentioned SAML elements and describes how they can be used to implement a use case. The most important use case is the Web Browser SSO (Single Sign On) profile [24]. A SAML protocol flow based on the Web Browser SSO Profile where both the IdP and the SP use the HTTP Post binding is illustrated next [35]:

1. A user wants to access a service provided by a SP. The user uses her browser to request to access the resource.

2. The SAML interface in the SP traps the request and redirects the user to the Discovery Service, also known as the Where Are You From (WAYF) Service, where a pre-configured list of trusted IdPs are shown to the user. The user selects one of the IdPs.

3. The SAML interface at the SP builds a SAML Authentication Request element. This element is deflated, base64-encoded, URL-encoded and then inserted into an XHTML Form and is sent back to the user in response to that previous request.

4. The browser retrieves a few values from the form and submits a POST request to the chosen IdP using those values.

5. The SAML interface (in this case the SSO service) in the IdP traps the request and checks using cookies if there is any security context (Identity information, meaning if the user is already authenticated) regarding the user at the IdP. If so, Step 6 is ignored.

6. The user authentication takes place. There are many authentication mechanisms supported by the SAML and the one used depends on the respective IdP.

7. The SSO Service returns an XHTML form that contains the SAML Response.

8. The browser extracts the SAML Response and a few parameters from the form and issues an HTTP Post Request to the Assertion Consumer Service of the SP with these retrieved values.

9. The Assertion Consumer Service at the SP extracts the Response, validates it and if everything is okay, creates a security context for the user at the SP. The user is re-directed to the requested resource.

10. The browser again submits an access request to the SP.

11. As there exists a security context this time, the SP checks if the user is authorised to access the resource. If so, the resource is returned to the user. If not, the resource is not returned and an error message is displayed to the user.

The concept of trust and trust management on the setting of online services is a widely studied topic. There exists a numerous amount of work on these topics and the way the concept of trust has been defined or considered vary considerably on these works. For the purpose of this paper, we prefer the following definition quoted from [16] which ultimately was inspired from [21] is preferred: *Trust is the extent to which one party is willing to depend on something or somebody in a given situation with a feeling of relative security, even though negative consequences are possible*. In the setting of identity federation, this definition outlines how much one entity in the federation can rely upon another entity inside the same federation. This notion essentially exemplifies trust as confidence as illustrated in [20].

The issue of such trust plays a central role in SAML. 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. 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 information such as entity descriptor (known as the entity ID, an identifier 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, 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 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 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, an IdP will respond only to those requests that are initiated from a SP whose metadata can be found in its metadata repositories (in its TAL). Exchanging and maintaining the repositories of metadata and thus managing trust becomes extremely difficult as the number of the IdPs and SPs grows up and is a well-known problem of the SAML [1]. In addition, pre-configuring trust before any interaction hinders to establish a federation between two prior unknown parties in a dynamic fashion. Allowing to create federations in a dynamic fashion would open up the door for new business scenarios and novel web-enabled services. In this paper, this issue is addressed.

3 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 [11] 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 on the metadata can be derived by just verifying the signature in the metadata using the embedded certificate with the traditional Public Key Infrastructure 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 a distributed SAML metadata management system [25], [26]. Based on the proposal and the working draft, an implementation of dynamic SAML has been developed by UNINETT (Site 1) (in their SimpleSAMLphp project [32], [33]. The SimpleSAMLphp is a native php-based implementation of the SAML protocol stack that can be used to quickly deploy a SAML IdP or SP. 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 a 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 a 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. It can be regarded as 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 other existing works that also provide proposals for dynamic federations. In [3], 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 respond with reputation data. The authors in [39] 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 a 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 [38] 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 our work has been based on the Dynamic SAML. The existing SimpleSAMLphp implementation will be extended so that a federation can be established and managed fully dynamically considering different trust issues.

In addition, any discussions on dynamic federations only seem to concentrate on how an entity can join in a federation and discussions on other aspects (maintenance, provisioning and revocation) are rare. Hence, it is essential to address all aspects covering the full life-cycle of a dynamic identity federation.

4 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 proceeding any further, it is, therefore, essential to define the term Dynamic Federation formally which is presented next:

Definition 2. 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 exists a legally binding technical and/or 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 possible since no party is bound to behave as it should and therefore take proper precautions. This leads us to the topic of trust which will be explored in the following subsection.

The life-cycle of a dynamic federation maintains the same steps involved in the life-cycle of a traditional identity federation: association, provisioning, maintenance and revocation, however, all such steps must be carried out in a fully dynamic fashion without any administrator to intervene. Providing the capability to create federations dynamically in real time will require to enable users to carry out some of these steps such as association, provisioning and revocation dynamically. This contrasts with the traditional federations where all such steps must be carried out by the administrators. However, the other task of Provisioning & Maintenance must be carried out by the respective administrator to ensure that any malicious user cannot update the metadata of the entities in the federation. Such mechanisms have not been considered in any previous work.

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 anyone accountable in case of disputes. 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. This takes us back to the open trust paradigm of the OpenID where every organisation (IdP/SP) is assumed to be trusted even though it might behave maliciously.

Journal of Theoretical and Applied Electronic Commerce Research
ISSN 0718–1876 Electronic Version
VOL 10 / ISSUE 2 / MAY 2015 / 53-76
© 2015 Universidad de Talca - Chile

This paper is available online at
www.jtaer.com
DOI: 10.4067/S0718-18762015000200005

Such trust issues have not been considered while drafting the Dynamic SAML which allows any SP and any IdP to be federated with each other without any sort of verification. This is a serious flaw of the Dynamic SAML and a mechanism must be introduced so that trust can be re-instated for dynamic federation. 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 a 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, it must be ensured 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, there are two requirements to fulfil: i) Fully automate the life-cycle in a federation for both parties and ii) Consider trust issues in a dynamic federation. With these two goals in mind the notion of fully trusted, semi-trusted and untrusted entities based on how different entities federate with each other are introduced.

Definition 3. 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 held accountable in case the other party behaves maliciously.

Definition 4. 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 they behave maliciously with the absence of any contract between them.

Definition 5. 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. Even though, different entities have been categorised based only on how they federate, these three types of entities have their own effect on other parts of the life-cycle of the federation which will be explored later on.

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 a federation between two entities can be created, updated or removed dynamically, the conditions for establishing and removing 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 a 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, a set of conditions that have been assumed for developing our proof of concept of managing a dynamic federation using SAML is presented next.

- Only a valid user of an IdP is allowed to add a 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.

- Only a valid user of an IdP who has added a SP to that IdP dynamically is allowed to initiate the procedure to revoke the SP from the federation with that IdP. In this case, the IdP will be removed from the TAL of the SP and the SP will be removed from the TAL of the IdP. This is to ensure that another user does not accidentally defederate entities which have been federated by other users and hence might still be used by them.

- Only the administrator of each entity is allowed to dynamically update any information in the metadata stored in the TAL of other entities. This is to ensure that users with malicious intent cannot update or corrupt such information of the metadata.

- 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.

Journal of Theoretical and Applied Electronic Commerce Research
ISSN 0718–1876 Electronic Version
VOL 10 / ISSUE 2 / MAY 2015 / 53-76
© 2015 Universidad de Talca - Chile

This paper is available online at
www.jtaer.com
DOI: 10.4067/S0718-18762015000200005

- A dynamically added IdP should always be tagged as an untrusted entity for the SP.

- IdPs should ensure that they do 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 not 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 (National Institute of Standards and Technology) 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 [2]. 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, it is proposed that entities have to be federated/defederated in a fully automatic fashion without any administrative intervention to harness the full potential of dynamic federation and while doing so all entities must consider trust issues involved. Moreover, our proposal also outlines the conditions for managing such federations in a fully automated manner. It should be noted that the conditions outlined before are one of the many ways to fulfil our proposals, other implementations may require different sets of conditions suitable for their use-case scenarios.

5 Proof of Concept: Dynamic Management

In this section the proof of concept that has been developed to illustrate the applicability of our proposals for managing the dynamic identity federations will be discussed. The SimpleSAMLphp has been used and modified to meet our requirements. The IdP and one SP deployed with this modified version of the SimpleSAMLphp. 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 tables called *semitrusted* and *untrusted* to store the entity IDs of semi-trusted and untrusted SPs respectively. Moreover, the IdP uses two additional tables called *idpadmin* and *spadmin*. The *idpadmin* stores the SP entity ID and the IdP-generated admin code that can be used to update the metadata of the IdP stored at the TAL of the SP and the *spadmin* stores the entity ID of the SP with the SP-generated admin code to update the metadata of the SP stored at the TAL of the IdP. The purpose of the admin codes and how they are generated are explained in the appropriate place. Similarly, the SP is also configured to use a MySQL database at its end where it uses a table called *untrusted* to store the entity IDs of IdPs that have been federated dynamically. Likewise, the SP uses two additional tables called *idpadmin* and *spadmin*. The *idpadmin* table at the SP stores the entity ID of the IdP along with an IdP-generated admin code to update the metadata of the IdP stored at the TAL of the SP and the *spadmin* table at the SP stores the entity ID of the IdP along with the SP-generated admin code to update the metadata of the SP stored at the TAL of the IdP. Note that only admins of the IdP or the SP has access to the *spadmin* and *idpadmin* tables since the general users are not allowed to make any changes to the metadata.

The following subsections will consider three different scenarios: i) dynamic federation, ii) dynamic defederation and iii) dynamic update of identity federations.

5.1 Use-case: Dynamic Federation

The architectural setup for this scenario is that there are one IdP and one SP. 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. With this setup, the protocol flow, illustrated in Figure 2, for federating the IdP and the SP dynamically while trying to access a service from the SP is given next.

1. A user visits the SP for the first time to access one of its services. 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 to discover her IdP 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 (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 3) 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 a SP with the IdP.

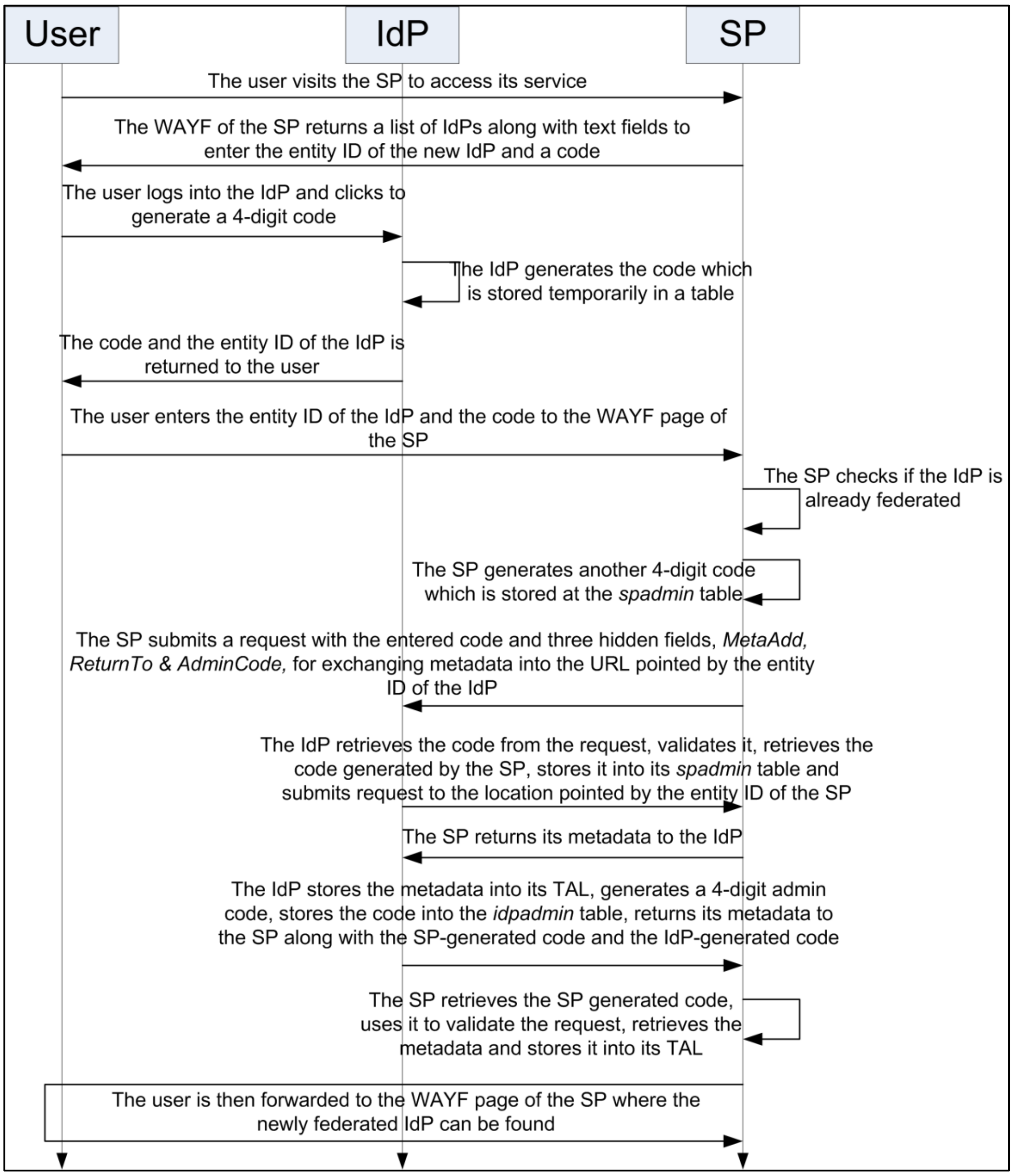
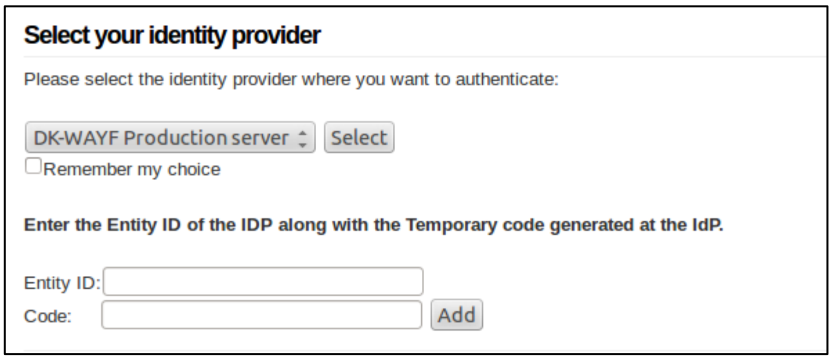

User
IdP
SP
The user visits the SP to access its service
The WAYF of the SP returns a list of IdPs along with text fields to enter the entity ID of the new IdP and a code
The user logs into the IdP and clicks to generate a 4-digit code
The IdP generates the code which is stored temporarily in a table
The code and the entity ID of the IdP is returned to the user
The user enters the entity ID of the IdP and the code to the WAYF page of the SP
The SP checks if the IdP is already federated
The SP generates another 4-digit code which is stored at the *spadmin* table
The SP submits a request with the entered code and three hidden fields, *MetaAdd, ReturnTo* & *AdminCode,* for exchanging metadata into the URL pointed by the entity ID of the IdP
The IdP retrieves the code from the request, validates it, retrieves the code generated by the SP, stores it into its *spadmin* table and submits request to the location pointed by the entity ID of the SP
The SP returns its metadata to the IdP
The IdP stores the metadata into its TAL, generates a 4-digit admin code, stores the code into the *idpadmin* table, returns its metadata to the SP along with the SP-generated code and the IdP-generated code
The SP retrieves the SP generated code, uses it to validate the request, retrieves the metadata and stores it into its TAL
The user is then forwarded to the WAYF page of the SP where the newly federated IdP can be found

Figure 2: Protocol flow for dynamic federation

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 (Figure 4). This page also shows the entity ID of the IdP which the user needs to enter at the WAYF page (Figure 4). Once the Generate button is clicked, a 4 digit random number is generated and displayed (Figure 5). This random number is also stored temporarily in a database table called code along with the user identifier. The code is used during metadata exchange for verification (see the following steps). Here, a 4-digit code has been used in our implementation, other implementations may opt for other type of codes according to their own requirements..

This paper is available online at
www.jtaer.com
DOI: 10.4067/S0718-18762015000200005



Select your identity provider

Please select the identity provider where you want to authenticate:

DK-WAYF Production server | Select

☐ Remember my choice

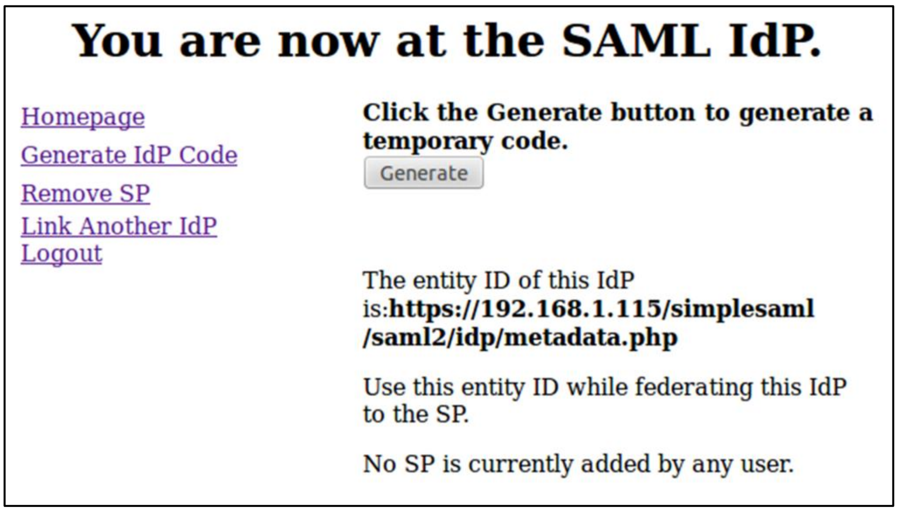
Enter the Entity ID of the IDP along with the Temporary code generated at the IdP.

Entity ID:

Code: | Add

Figure 3: Additional text fields at the WAYF

4. 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 (Figure 4). This page also shows the entity ID of the IdP which the user needs to enter at the WAYF page (Figure 4). Once the Generate button is clicked, a 4 digit random number is generated and displayed (Figure 5). This random number is also stored temporarily in a database table called code along with the user identifier. The code is used during metadata exchange for verification (see the following steps). Here, a 4-digit code has been used in our implementation, other implementations may opt for other type of codes according to their own requirements.



You are now at the SAML IdP.

Homepage
Generate IdP Code
Remove SP
Link Another IdP
Logout

Click the Generate button to generate a temporary code.

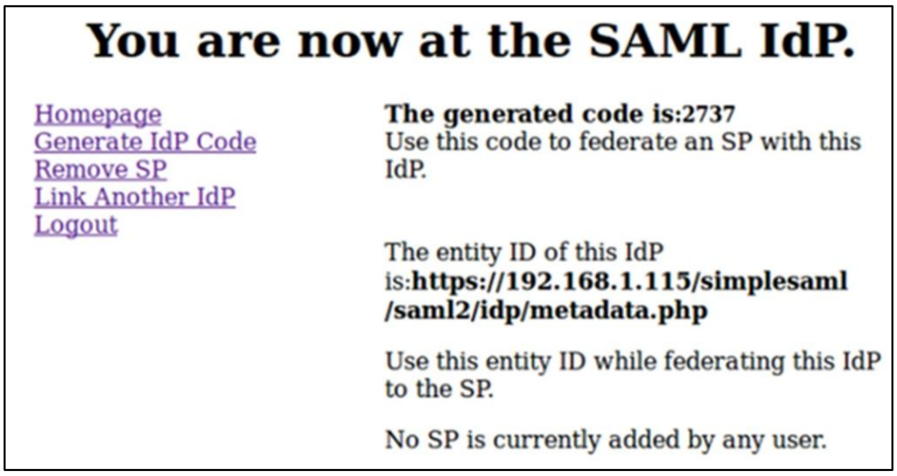
Generate

The entity ID of this IdP is:**https://192.168.1.115/simplesaml/saml2/idp/metadata.php**

Use this entity ID while federating this IdP to the SP.

No SP is currently added by any user.

Figure 4: Code generate page at the IdP



You are now at the SAML IdP.

Homepage
Generate IdP Code
Remove SP
Link Another IdP
Logout

The generated code is:2737
Use this code to federate an SP with this IdP.

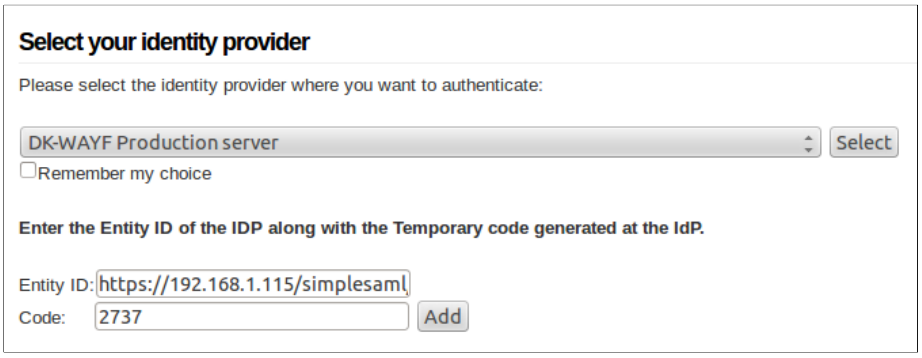
The entity ID of this IdP is:**https://192.168.1.115/simplesaml/saml2/idp/metadata.php**

Use this entity ID while federating this IdP to the SP.

No SP is currently added by any user.

Figure 5: Generated code at the IdP

5. After generating the code, the user notes the entity ID of the IdP and the generated code. Then she inserts the entity ID and the code to the WAYF page of the SP (Figure 6) and clicks the *Add* button.


Select your identity provider

Please select the identity provider where you want to authenticate:

DK-WAYF Production server

Remember my choice

Enter the Entity ID of the IDP along with the Temporary code generated at the IdP.

Entity ID: https://192.168.1.115/simplesaml

Code: 2737

Figure 6: Entering values at the WAYF

6. 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.

7. Assuming there is no error during verification, the SP generates a unique 4 digit random code which is then stored at the spadmin along with the entity ID of the IdP. Then 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 three hidden fields called MetaAdd, ReturnTo & AdminCode. The values of the MetaAdd & ReturnTo fields contain the entity ID of the SP and the URL of the services that the user has requested at the first place which initiated the SAML flow whereas the AdminCode field contains the SP generated random admin code. Remember that SimpleSAMLphp implementation of dynamic SAML requires that entity ID be the endpoints from where the metadata can be fetched. This approach has been extended by adding a verification mechanism.

8. 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.

9. Assuming that the code field contains a valid code, the IdP retrieves the value of the MetaAdd & AdminCode fields which contain the entity ID of SP and the 4 digit SP-generated admin code. The IdP stores this admin code at the spadmin table at the IdP along with the entity ID of the 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.

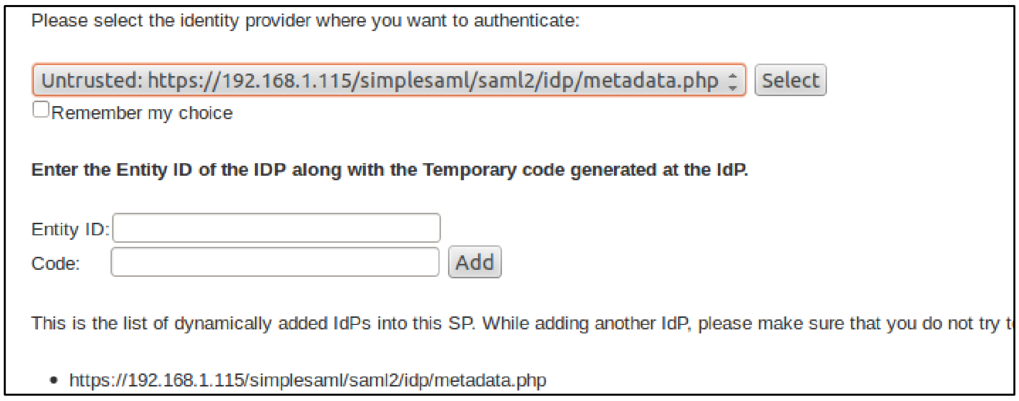
10. When the SP receives this request, it returns its metadata.

11. 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 user identifier for that code is retrieved from the code table and the entity ID of the SP is added into the untrusted table of the database along with the code and the user identifier. Then the used code is removed from the code table to ensure that it cannot be reused again. If the metadata verification is not correct, the respective entry from the spadmin is removed and an error message is returned to the SP.

12. If everything goes right at the IdP, it generates a 4-digit random admin code which is then stored at the idpadmin table of the IdP along with the entity ID of the SP. Then the metadata of the IdP along with the ReturnTo field and its value, a code field containing the previously generated value and the AdminCode containing the recently IdP-generated 4 digit admin code are returned to the requesting endpoint of the SP, where the metadata, ReturnTo, code and AdminCode values are separated. Like before, the SP goes through 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 along with the code is added into the untrusted table of its database and thus the IdP is tagged as an untrusted IdP.

Moreover, the entity ID of the IdP along with the value of the AdminCode parameter are stored at the idpadmin table.

13. 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 7). Moreover, it is shown to the user that the IdP has already been added as an untrusted IdP so that no other user tries to add it once again which will result in an error.


Please select the identity provider where you want to authenticate:

Untrusted: https://192.168.1.115/simplesaml/saml2/idp/metadata.php
Select
Remember my choice

Enter the Entity ID of the IDP along with the Temporary code generated at the IdP.

This is the list of dynamically added IdPs into this SP. While adding another IdP, please make sure that you do not try t

- https://192.168.1.115/simplesaml/saml2/idp/metadata.php

Figure 7: Added IdP at the WAYF

The usage of codes while creating dynamic federations requires further explanations. Allowing arbitrary users to create federations, in general, does not impose any threats. However, if an arbitrary user can create federations between an IdP and SP in which she does not have any account, she will not be able to use that IdP, or the federation, for any use-case scenario. It means that the federations between that said IdP and SP will be of no use and the storage of metadata in the respective TAL will be wasted for nothing. That is why this particular choice has been opted. And also if a particular setup requires that such a federation can only be created by the admin, it can be easily adopted using our preferred option where only the admin can generate the required code at the IdP and hence only the admin is allowed to create such federations.

In the traditional SAML scenario, it is assumed that both IdP and SP are visible to each other as they are expected to be online. This assumption does not hold for SAML IdPs which might be installed locally inside the user's computer from where she is trying to access a service. One example of such an IdP is the Portable Personal Identity Provider (PPIdP in short) as introduced in [9]. The PPIdP is a special type of SAML IdP, based on the SAML HTTP Post binding, that is hosted in a mobile device owned and/or used by the user and allows the user to use the PPIdP while accessing services using a browser in that mobile phone. Firstly, the user has to federate the PPIdP with the SP using the steps described previously. However, the problem occurs during the metadata exchange period since the SP at the SimpleSAMLphp is designed to communicate with the IdP directly. The implementation of the SP at the SimpleSAMLphp has been modified so that the SP can communicate with the local IdP. This has been done by embedding a JavaScript into an XHTML form which is submitted automatically at the entity ID of the IdP. Since the JavaScript runs inside the browser and the local IdP (PPIdP) is visible to the browser, it can properly submit these requests. The SP, on the other hand, is assumed to be online. Therefore it is visible to the IdP and no special treatment is required.

5.2 Use-case: Dynamic Defederation

The setup for this scenario is that an IdP and a SP have been federated dynamically using the steps described previously. Now, the user wants to defederate them dynamically. A user is only allowed to defederate those entities that she federated previously. This is to ensure that another user does not accidentally defederate entities which have been federated by other users and hence might still be used by them. Unlike the federation, the protocol for defederating starts at the IdP.

With this setup, the protocol flow, illustrated in Figure 8, for defederating the IdP and the SP dynamically is given next:

1. The user logs in to the IdP and then clicks the Remove SP link at the IdP.

2. If the user has not federated the IdP and a SP dynamically, a message will be shown to the user. Assuming, the user has already federated the IdP and the SP(s), a list of SPs are retrieved from the untrusted/semitrusted tables using the user identifier to ensure that only the SP that has been federated

dynamically by the user is retrieved. The reason for checking both the tables will be explained in the Section 6. Then a list of check boxes containing the entity ID of these dynamically added SPs is shown (Figure 9).

3. The user selects one of the SPs and clicks the Remove button.

4. The IdP uses the entity ID of the selected SP to retrieve the code from the untrusted/semitrusted table.

5. Once the code is retrieved, the corresponding entry is removed from the respective table. The IdP also removes the corresponding SP entry from the idpadmin & spadmin tables.

6. Then, a request to remove the IdP entry from the TAL of the SP is sent to the entity ID of the SP. The request contains three parameters: i) the remove parameter containing the entity ID of the IdP, ii) the code parameter containing the code retrieved in Step 4 and iii) the ReturnTo parameter containing the link of the Remove page at the IdP where the user will return later.

7. The SP validates if there exists the entity ID of an IdP along with the code containing the respective value in its untrusted table. If such an entry is found, the entry is removed from the table, otherwise, an error message is shown. Moreover, the SP also removes the respective entry from the idpadmin & spadmin tables.

8. Then the user is redirected to the location retrieved from the ReturnTo parameter and the user is informed about the successful defederation.

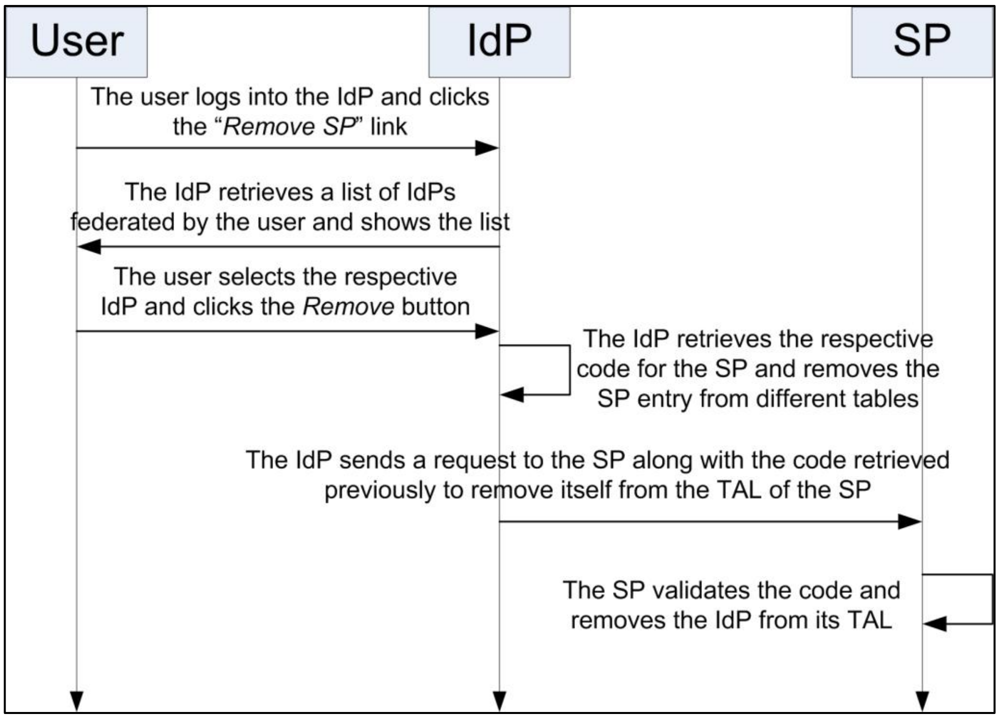
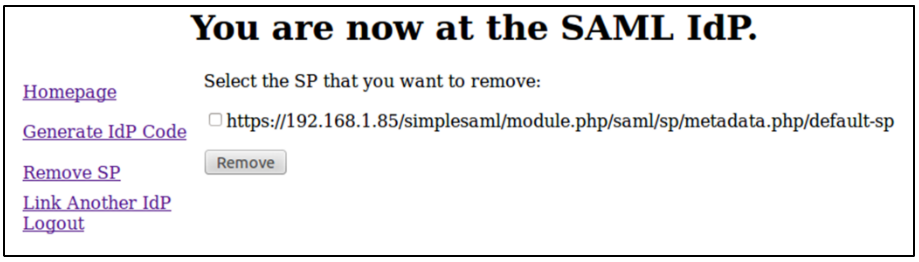

User
IdP
SP
The user logs into the IdP and clicks the "*Remove SP*" link
The IdP retrieves a list of IdPs federated by the user and shows the list
The user selects the respective IdP and clicks the *Remove* button
The IdP retrieves the respective code for the SP and removes the SP entry from different tables
The IdP sends a request to the SP along with the code retrieved previously to remove itself from the TAL of the SP
The SP validates the code and removes the IdP from its TAL

Figure 8: Protocol flow for dynamic defederation



You are now at the SAML IdP.

Homepage
Generate IdP Code
Remove SP
Link Another IdP
Logout

Select the SP that you want to remove:

☐ https://192.168.1.85/simplesaml/module.php/saml/sp/metadata.php/default-sp

Figure 9: Removing a SP at the IdP

As described in the previous use-case, the same approach has been adopted to defederate any locally installed IdP using a JavaScript embedded into an XHTML form which is submitted automatically to the entity ID of the IdP and like before the SP is assumed to be online.

Please note that general users might not be motivated to defederate entities since they do not gain anything doing this. However, this use-case will be useful in the setup where the admin has federated the entities and there is no need to maintain that federation.

5.3 Use-case: Dynamic Update

The setup for this scenario is that an IdP and a SP have been federated dynamically using the steps described in Section 5.1. The *idpadmin* table at the IdP contains the entity ID of the SP along with the IdP-generated 4 digit *AdminCode* and the *idpadmin* table at the SP contains the entity ID of the IdP with the same IdP-generated 4 digit AdminCode. Now, the admin of the IdP has made some changes (let's assume the changed entries are the Binding & Location entries of the *SingleSignOnService* entry of the *IDPSSODescriptor* field of the metadata) into the IdP's metadata locally and the admin wants to propagate these changes into the metadata stored at the SP in a dynamic fashion. It is assumed that the IdP has provided an interface to allow the admin to propagate these changes.

With this setup, the protocol flow, illustrated in Figure 10, for updating metadata dynamically is given next.

1. The admin of the IdP visits the admin interface of the IdP and logs in there.

2. The admin clicks the Update Metadata link at the interface and the update page is shown to the user (Figure 11).

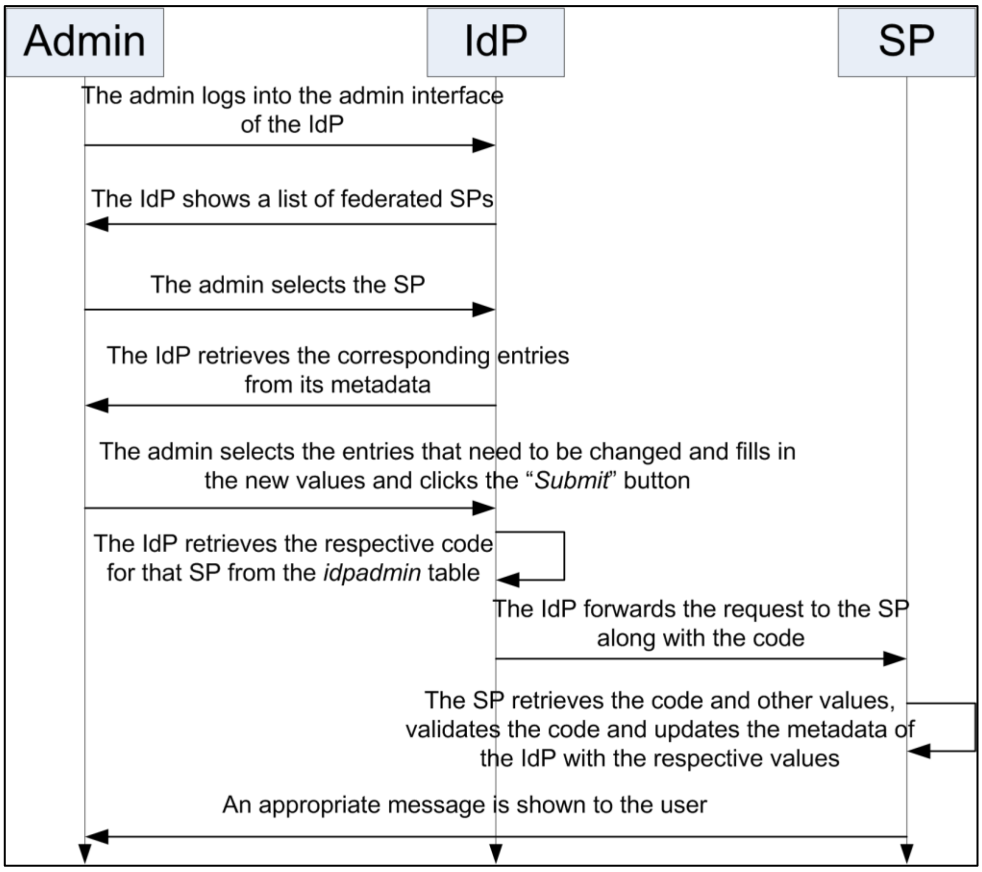
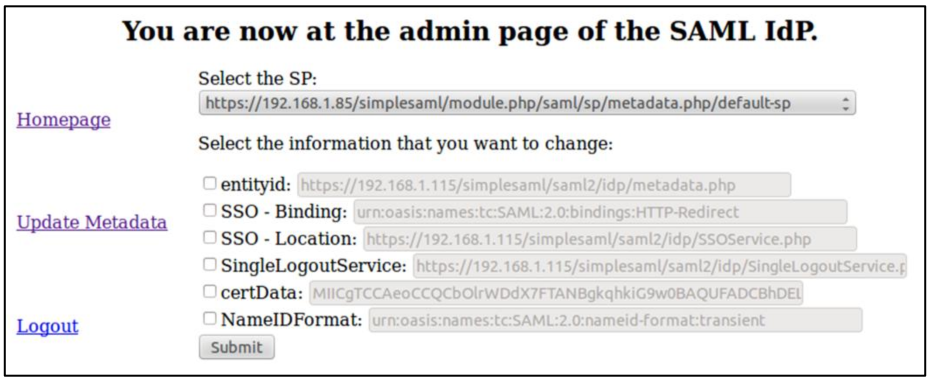

Admin
IdP
SP
The admin logs into the admin interface of the IdP
The IdP shows a list of federated SPs
The admin selects the SP
The IdP retrieves the corresponding entries from its metadata
The admin selects the entries that need to be changed and fills in the new values and clicks the "*Submit*" button
The IdP retrieves the respective code for that SP from the *idpadmin* table
The IdP forwards the request to the SP along with the code
The SP retrieves the code and other values, validates the code and updates the metadata of the IdP with the respective values
An appropriate message is shown to the user

Figure 10: Protocol flow for dynamic update


You are now at the admin page of the SAML IdP.

Homepage
Update Metadata
Logout

Select the SP: https://192.168.1.85/simplesaml/module.php/saml/sp/metadata.php/default-sp

Select the information that you want to change:

- ☐ entityid: https://192.168.1.115/simplesaml/saml2/idp/metadata.php
- ☐ SSO - Binding: urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect
- ☐ SSO - Location: https://192.168.1.115/simplesaml/saml2/idp/SSOService.php
- ☐ SingleLogoutService: https://192.168.1.115/simplesaml/saml2/idp/SingleLogoutService.p
- ☐ certData: MIICgTCCAeoCCQCbOlrWDdX7FTANBgkqhkiG9w0BAQUFADCBhDEL
- ☐ NameIDFormat: urn:oasis:names:tc:SAML:2.0:nameid-format:transient

Submit

Figure 11: Metadata update page

3. The update page contains a dropdown list containing the entity IDs of all (dynamically or statically) SPs federated with the IdP (Figure 8).

4. The admin selects the entity ID of the SP to which she wants to propagate these changes.

5. The selected entity ID of the SP is used to retrieve the respective entries of the metadata and those entries are displayed using check boxes to the admin. The admin selects the Binding & Location (SSO-Binding and SSO-Location entries in Figure 8) entries and clicks the Submit button. The IdP restores the AdminCode from the idpadmin table using the selected entity ID of the SP.

6. Then a request is sent to the entity ID of the SP. The request contains several parameters: the update parameter containing the entity ID of the IdP, the AdminCode parameter containing the code, the name of the entries of the metadata that need to be updated and their respective updated values and the ReturnTo parameter containing the location of the Update page at the IdP where the admin will return. The following steps take place only if the SP is online and is able to receive the request. In case the selected SP is offline, the request times out after a period and an appropriate error message is displayed to the admin.

7. Once the SP receives this request, all values of the parameters are retrieved. The update parameter informs the SP that the IdP with the entity ID in that parameter wants to update its metadata. The AdminCode parameter is used to authorise the request by matching them with the values stored at the idpadmin table at the SP.

8. If they match, the metadata is updated with the requested values and an appropriate message is returned to the location retrieved from the ReturnTo parameter. If no match is found, an error message is returned to that IdP location.

This similar approach can be used to update the metadata of the SP stored at the TAL of the IdP. As described in the previous use-case, the same approach has been adopted to update metadata of any locally installed IdP using a JavaScript embedded into an XHTML form which is submitted automatically to the entity ID of the IdP and like before the SP is assumed to be online.

6 Proof of Concept: Service Provisioning

In this section the proof of concept that has been developed to illustrate how users can access services using the federations created in dynamic fashion will be discussed. The following subsections will consider two different scenarios: i) IdP-SP Scenario to illustrate our concept for Type 1 Federation and IdP-IdP-SP Scenario to illustrate the concept for Type 2 Federation.

For the first use-case, it is assumed that the IdP and the SP have already been federated dynamically where the IdP is tagged as an untrusted entity at the SP and the SP is tagged as an untrusted entity at the IdP. Before discussing the protocol flow, the effect of trust issues discussed previously needs to be analysed.

A mechanism has been provided to ensure that an admin of the IdP can configure which attributes to release to a semi-trusted SP. A configuration parameter called *semitrusted.sp* has been used 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 release 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) SSPHPAuthProc [34]. An AuthProc allows the system to do additional activities 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. This module has been modified to allow the IdP to show only those attributes that can be found in the *semitrusted.sp* parameter. From the displayed list, the user can choose which attributes she wants to release to the SP. Note that the illustrated use-cases are applicable for any locally installed IdPs as well as for any traditional IdPs using the mechanisms discussed previously.

6.1 Use-case: IdP-SP Scenario

With this setup, the protocol flow, illustrated in Figure 12, for the scenario is given next.

1. A user visits the SP for the first time to access one of its services. 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 of federated IdPs with the SP is shown. The list also contains the untrusted IdP that has been federated in the previous use-case.

2. 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.

3. 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 13 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.

4. 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 and the chosen attributes would be released to the SP. 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 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 of maximum 1 and authorise the user accordingly.

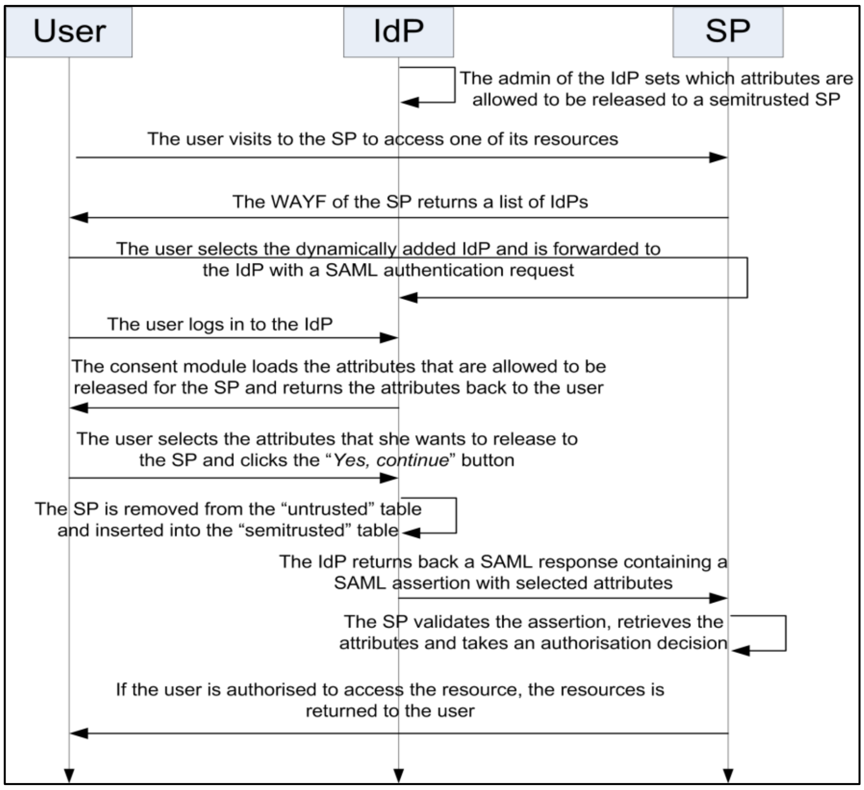
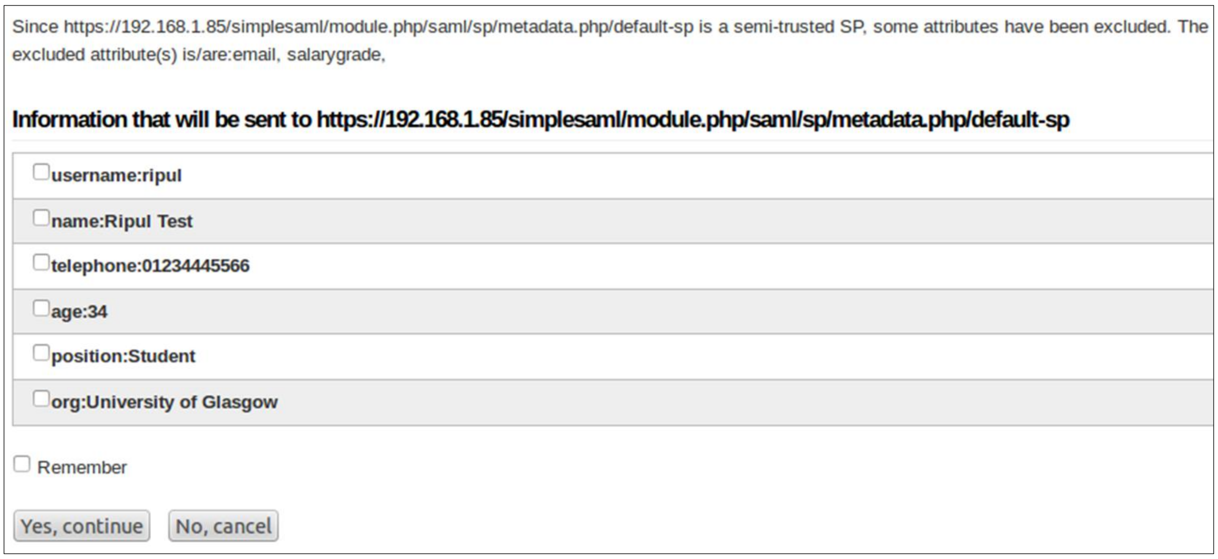

User
IdP
SP
The admin of the IdP sets which attributes are allowed to be released to a semitrusted SP
The user visits to the SP to access one of its resources
The WAYF of the SP returns a list of IdPs
The user selects the dynamically added IdP and is forwarded to the IdP with a SAML authentication request
The user logs in to the IdP
The consent module loads the attributes that are allowed to be released for the SP and returns the attributes back to the user
The user selects the attributes that she wants to release to the SP and clicks the "*Yes, continue*" button
The SP is removed from the "untrusted" table and inserted into the "semitrusted" table
The IdP returns back a SAML response containing a SAML assertion with selected attributes
The SP validates the assertion, retrieves the attributes and takes an authorisation decision
If the user is authorised to access the resource, the resources is returned to the user

Figure 12: Protocol for IdP-SP service provisioning


Since https://192.168.1.85/simplesaml/module.php/saml/sp/metadata.php/default-sp is a semi-trusted SP, some attributes have been excluded. The excluded attribute(s) is/are:email, salarygrade,

Information that will be sent to https://192.168.1.85/simplesaml/module.php/saml/sp/metadata.php/default-sp

- ☐ **username:ripul**
- ☐ **name:Ripul Test**
- ☐ **telephone:01234445566**
- ☐ **age:34**
- ☐ **position:Student**
- ☐ **org:University of Glasgow**

☐ Remember

Figure 13: Attributes to be released at the consent page

6.2 Use-case: IdP-IdP-SP Scenario

The described protocol flows will allow the users to access services using a dynamically federated (untrusted) IdP. 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 [4]. The SP would think that it is interacting with 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, SimpleSAMLphp has been used 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. This feature has been used 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, the SimpleSAMLphp has been modified 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 escalation 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 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, illustrated in Figure 14, for the scenario is given next.

1. The user logs in to the untrusted IdP and generates a code just like the previous one. This code will be used to link the proxy IdP with this IdP. The page also shows the entity ID of the untrusted 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 15). As there is no other authentication sources, it only shows the example-userpass source (an authentication source based on locally stored username/password in a database) 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.

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 proxy IdP 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, an appropriate error message is 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 is 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 16).

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 chosen 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.

One might question the incentive for the fully trusted IdP to act as a proxy IdP for the untrusted IdP. The incentive here for the trusted IdP is that this service will allow it to attract more users since the user will be able to choose attributes from different linked IdPs via a single interface provided by the trusted IdP. With a little effort this single interface can be used to aggregate attributes from multiple IdPs in a single SP session which is not possible in the current settings of FIM.

7 Discussion & Future Work

Our approach offers several key advantages:

- Users can create federations just in time and whenever required. Even though it has not been 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 defederating 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 Identity Management System 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 [7], [37]. Since the traditional SAML enforces users to use only those IdPs that can be in the TAL, segregation of power is not fully exercised [7]. Allowing to create federations dynamically would allow the users to choose a specific IdP for a specific scenario and thus segregation of power can be ensured.

- Our proposal illustrates how the life-cycle of any identity federation can be managed dynamically in a fully automatic fashion. This approach can be extremely useful to automate the task of federation management which now has to be done manually.

- Allowing users to link two of their IdPs would enable them to use their own IdPs (e.g. a locally installed IdP) 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 to provide for the fully trusted IdP. However, this should be treated with cautions as it might escalate privileges as discussed previously. In such cases, the LoA value should used as suggested previously.

Currently, the implementation only deals with one level of trust for linking IdPs: IdP-IdP. It will be interesting to see if it can be extended for multi-levels so that the trust hierarchy looks something like this: IdP-IdP-IdP-...-IdP. Technical challenges for exchanging metadata might be not very difficult, however, establishing the transitive trust between all these IdPs could be challenging. Also the current model is currently based only on the SimpleSAMLphp implementation of the SAML. It is necessary to investigate how our proposals can be implemented in other SAML implementations as well to ensure its widespread adoption. The best way to achieve this is by converting our proposals into specifications and then integrating them with the official specifications to ensure consistent implementations. Also it will be beneficial for the users if they can utilise the IdP-IdP-SP setting to aggregate attributes from multiple IdPs in a single SP session. It is needed to explore how it can be achieved using this setting. The developed mechanism and the proof of concept are not currently available online and there are plans to make them available online soon.

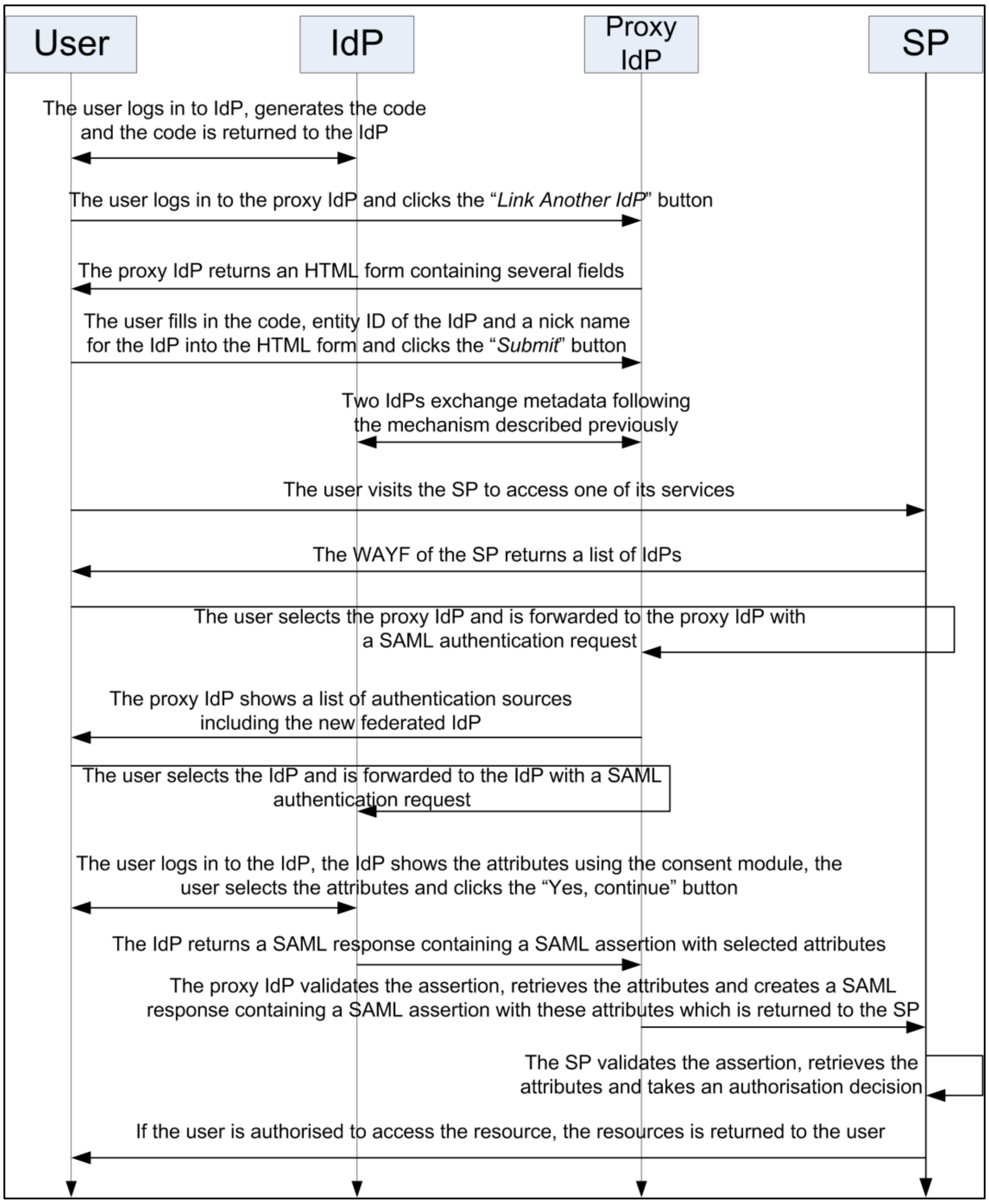
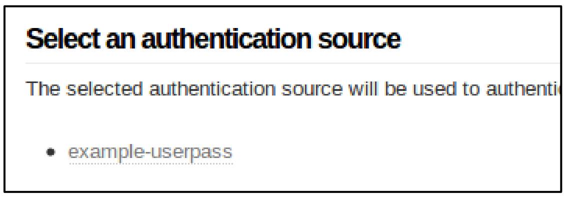

User
IdP
Proxy IdP
SP
The user logs in to IdP, generates the code and the code is returned to the IdP
The user logs in to the proxy IdP and clicks the "Link Another IdP" button
The proxy IdP returns an HTML form containing several fields
The user fills in the code, entity ID of the IdP and a nick name for the IdP into the HTML form and clicks the "Submit" button
Two IdPs exchange metadata following the mechanism described previously
The user visits the SP to access one of its services
The WAYF of the SP returns a list of IdPs
The user selects the proxy IdP and is forwarded to the proxy IdP with a SAML authentication request
The proxy IdP shows a list of authentication sources including the new federated IdP
The user selects the IdP and is forwarded to the IdP with a SAML authentication request
The user logs in to the IdP, the IdP shows the attributes using the consent module, the user selects the attributes and clicks the "Yes, continue" button
The IdP returns a SAML response containing a SAML assertion with selected attributes
The proxy IdP validates the assertion, retrieves the attributes and creates a SAML response containing a SAML assertion with these attributes which is returned to the SP
The SP validates the assertion, retrieves the attributes and takes an authorisation decision
If the user is authorised to access the resource, the resources is returned to the user

Figure 14: Protocol for IdP-IdP-SP service provisioning

Journal of Theoretical and Applied Electronic Commerce Research
ISSN 0718–1876 Electronic Version
VOL 10 / ISSUE 2 / MAY 2015 / 53-76
© 2015 Universidad de Talca - Chile

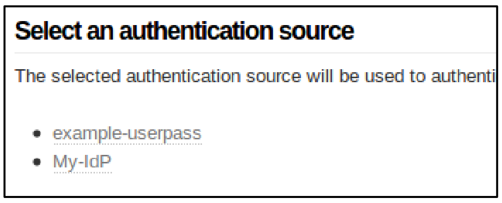
This paper is available online at
www.jtaer.com
DOI: 10.4067/S0718-18762015000200005


Select an authentication source

The selected authentication source will be used to authenti

- example-userpass

Figure 15: Initially, only one authentication source (Username/password) at the IdP


Select an authentication source

The selected authentication source will be used to authenti

- example-userpass
- My-IdP

Figure 16: Linked untrusted IdP as the authentication source in the proxy IdP

8 Discussion & Future Work

Our approach offers several key advantages:

- Users can create federations just in time and whenever required. Even though it has not been 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 defederating 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 Identity Management System 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 [7], [37] Since the traditional SAML enforces users to use only those IdPs that can be in the TAL, segregation of power is not fully exercised [7]. Allowing to create federations dynamically would allow the users to choose a specific IdP for a specific scenario and thus segregation of power can be ensured.

- Our proposal illustrates how the life-cycle of any identity federation can be managed dynamically in a fully automatic fashion. This approach can be extremely useful to automate the task of federation management which currently is done manually.

- Allowing users to link two of their IdPs would enable them to use their own IdPs (e.g. a locally installed IdP) 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 to provide for the fully trusted IdP. However, this should be treated with cautions as it might escalate privileges as discussed previously. In such cases, the LoA value should used as suggested previously.

Currently, the implementation only deals with one level of trust for linking IdPs: IdP-IdP. It will be interesting to see if it can be extended for multi-levels so that the trust hierarchy looks something like this: IdP-IdP-IdP-...-IdP. Technical challenges for exchanging metadata might be not very difficult, however, establishing the transitive trust between all these IdPs could be challenging. Also the current model is currently based only on the SimpleSAMLphp implementation of the SAML. It is necessary to investigate how our proposals can be implemented in other SAML implementations as well to ensure its widespread adoption. The best way to achieve this is by converting our proposals into specifications and then integrating them with the official specifications to ensure consistent implementations. Also it will be beneficial for the users if they can utilise the IdP-IdP-SP setting to aggregate attributes from multiple IdPs in a single SP session. It is needed to explore how it can be achieved using this setting. The developed mechanism and the proof of concept are not currently available online and there are plans to make them available online soon.

9 Conclusion

In this paper, the avenue of the dynamic federation has been explored: how the life-cycle of such a federation can be fully managed in an automatic fashion and how such federations can be used to provide services to the user. Managing an identity federation, especially when it is very large, can be a laborious job if it is done manually. Our proposed approach can be beneficial in such aspects as our proposal can be used for creating, removing and updating federations dynamically and in a fully automatic fashion. Also, our approach offers a better solution and can be easily adopted to any SAML implementation and requires no modification of the SAML Protocol.

The trust issues involved in such scenarios have been examined and several use-cases with detailed protocol flow have been illustrated. Using our use-cases as the foundation, different other use-cases can be created which can be

Journal of Theoretical and Applied Electronic Commerce Research
ISSN 0718–1876 Electronic Version
VOL 10 / ISSUE 2 / MAY 2015 / 53-76
© 2015 Universidad de Talca - Chile

This paper is available online at
www.jtaer.com
DOI: 10.4067/S0718-18762015000200005

applied for enterprise scenarios as well as for general online service provisioning scenarios. However, it should be noted that the issues of trust are very complex. The way the trust issues have been outlined 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 way and thereby hesitant to release any attributes to it. In such cases, it will be difficult to create federations in a dynamic way. It is our understanding that IdPs and SPs will need to relax the trust requirements if they want to allow their users to take advantage of dynamic federations. However, how much relaxation it will require will depend entirely on a specific use-case.

Websites List

Site 1: UNINETT
https://www.uninett.no/

References

[1] M. S. Bargh, B. Hulsebosch and H. Zandbelt. (2010, December) Scalability of trust and metadata exchange across federations. Terena. [Online]. Available: https://tnc2011.terena.org.
[2] W. E. Burr, D. F. Dodson and W. T. Polk. (2006, April) Electronic authentication guideline: Information security. NIST. [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-63/SP800-63V1_0_2.pdf.
[3] P. A. Cabarcos, F. A. Mendoza, A. Marín-López, and D. Díaz-Sánchez, Enabling SAML for dynamic identity federation management, in Wireless and Mobile Networking, volume 308 of IFIP Advances in Information and Communication Technology (J. Wozniak, J. Konorski, R. Katulski, and A. Pach, Eds.). Gdansk, Poland, Springer Boston, 2009, pp. 173-184.
[4] D. W. Chadwick, G. L. Inman, K. W. S. Siu, and M. S. Ferdous, Leveraging social networks to gain access to organisational resources, in Proceedings of the 7th ACM workshop on Digital identity management, DIM ’11, New York, NY, USA, 2011, pp. 43-52.
[5] D. W. Chadwick, Federated Identity Management, in FOSAD 2008/2009, LNCS 5705 (A. Aldini, G. Barthe and R. Gorrieri, Eds.). Berlin: Springer-Verlag, 2009, pp. 96-120.
[6] M. S. Ferdous, M. J. M. Chowdhury, M. Moniruzzaman, and F. Chowdhury, Identity federations: A new perspective for Bangladesh, in Proceedings Informatics, Electronics Vision (ICIEV), 2012 International Conference on, Dhaka, Bangladesh, 2012, pp. 219-224.
[7] M.S. Ferdous and R. Poet, A comparative analysis of Identity Management Systems, in Proceedings International Conference on High Performance Computing and Simulation (HPCS), Madrid, Spain, 2012, pp. 454-461.
[8] M. S. Ferdous and R. Poet, Dynamic identity federation using security assertion markup language (SAML), in Policies and Research in Identity Management, volume 396 of IFIP Advances in Information and Communication Technology (S. Fischer-Hübner, E. Leeuw and C. Mitchell, Eds.). London, UK, Springer Berlin Heidelberg, 2013, pp. 131-146.
[9] M. S. Ferdous and R. Poet, Portable personal identity provider in mobile phones. In to appear, in Proceedings of the 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (IEEE TrustCom-13), Melbourne, Victoria, Australia, 2013, pp. 736-745.
[10] M. S. Ferdous, Identity management with petname systems, M. S. thesis, Norwegian University of Science and Technology, Department of Telematics, Kjeller, Norway, 2009.
[11] P. Harding, L. Johansson and N. Klingenstein, Dynamic security assertion markup language: Simplifying single sign-on, Security Privacy, IEEE, vol. 6, no. 2, pp. 83-85, 2008.
[12] Internet2. (2005, March) The Shibboleth software. Internet2. [Online]. Available: http://shibboleth.internet2.edu/.
[13] ISO/IEC International Standard. (2011, December) Information technology - Security techniques - A framework for identity management. ISO. [Online]. Available: http://standards.iso.org/ittf/PubliclyAvailableStandards/-c057914_ISO_IEC_24760-1_2011.zip.
[14] ITU-T. (2009, September) Baseline capabilities for enhanced global identity management and interoperability. Committed to connecting the world. ITU-T. [Online]. Available: http://www.itu.int/ITU-T/recommendations/rec.aspx?rec=9456.
[15] A. Jøsang, J. Fabre, B. Hay, J. Dalziel, and S. Pope, Trust requirements in identity management, in Proceedings of the 2005 Australasian workshop on grid computing and e-research. Australian Computer Society, Inc., Darlinghurst, Australia, 2005, pp. 99-108.
[16] A. Jøsang, E. Gray and M. Kinateder, Simplification and analysis of transitive trust networks, Web Intelligence and Agent Systems, vol. 4, no. 2, pp. 139-161, 2006.
[17] A. Jøsang, M. A. Zomai and S. Suriadi, Usability and privacy in identity management architectures, in Proceedings of the fifth Australasian Symposium on ACSW frontiers’, Ballarat, Australia, 2007, pp. 143-152.
[18] F. Kaupa, L. Mathan, P. Sibieta, P. Cole, A. Shikiar and S. Deadman. (2004, March) Whitepaper: Benefits of federated identity to government. Liberty Alliance. [Online]. Available: http://projectliberty.org/liberty/content/-download/388/2723/file/Liberty_Government_Business_Benefits.pdf.
[19] J. Lewis, Enterprise identity management: It’s about the business, The Burton Group Directory and Security Strategies, Technical Report v1, London, UK, 2003.

[20] S. Marsh, Formalising trust as a computational concept, Ph.D. Thesis, Department of Computing Sciencie and Mathematics, University of Stirling, Scotland, 1994.
[21] D. H. McKnight and N. L. Chervany, The meanings of trust, University of Minnesota, Minnesota, USA, Technical Report 96-04, 1996.
[22] Modinis. (2011, November) Common terminological framework for interoperable electronic identity management. European Comission. [Online]. Available: http://ec.europa.eu/information_society/activities/ict_psp/documents/eid_terminology_paper.pdf
[23] OASIS Standard. (2005, March) Assertions and protocols for the OASIS security assertion markup language (SAML) V2.0. OASIS. [Online]. Available: http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf.
[24] OASIS Standard. (2005, March) Profiles for the OASIS security assertion markup language (SAML) V2.0. Oasis. [Online]. Available: http://docs.oasis-open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf.
[25] OASIS Standard. (2007, October) A profile for distributed SAML metadata management. Internet 2. [Online]. Available: https://spaces.internet2.edu/display/dsaml/A+profile+for+distributed+SAML+metadata+management.
[26] OASIS Standard. (2008, August) SAML V2.0 Metadata interoperability profile, working draft 01. Internet 2. [Online]. Available: https://spaces.internet2.edu/download/attachments/11275/draft-sstc-metadata-iop-01.pdf?version=2/RK=0/RS=SN7wI_PbaP91AgXcZJOioDILO6w-.
[27] OASIS Standard. (2009, May) Web services federation language (WSFederation) Version 1.2. Oasis. [Online]. Available: http://docs.oasis-open.org/wsfed/federation/v1.2/os/ws-federation-1.2-spec-os.pdf.
[28] OpenID. (2007, December) Authentication 2.0 - Final. Openid. [Online]. Available: http://openid.net/specs/openid-authentication-2_0.html.
[29] OpenID. (2007, December) Attribute exchange 1.0 - Final. Openid. [Online]. Available: http://openid.net/specs/-openid-attribute-exchange-1_0.html.
[30] OpenID. (2008, December) Provider authentication policy extension 1.0. Openid. [Online]. Available: http://-openid.net/specs/openid-provider-authentication-policy-extension-1_0.html.
[31] A. Pfitzmann and M. Hansen. (2010, August) A terminology for talking about privacy by data minimization: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management. Dud.Inf. [Online]. Available: http://dud.inf.tu-dresden.de/literatur/Anon_Terminology_v0.34.pdf.
[32] A. Solberg. (2010, February) Dynamic SAML. Feide RnD. [Online]. Available: https://rnd.feide.no/2010/02/18/-dynamic_saml/.
[33] UNINETT, (2008, September) SimpleSAMLphp. [Online]. Available: http://simplesamlphp.org/.
[34] UNINETT, (2012, October) Authentication processing filters. SimpleSAMLphp. [Online]. Available: http://-simplesamlphp.org/docs/stable/simplesamlphp-authproc.
[35] Wikipedia. (2011, September) Security assertion markup language. Wikipedia. [Online] Available: http://-en.wikipedia.org/wiki/Security_Assertion_Markup_Language.
[36] Windows. (2011, September) Life after windows XP. Windows. [Online]. Available: http://www.microsoft.com/-windows/products/winfamily/cardspace/default.mspx.
[37] WP3. (2005, May) Study on mobile identity management. Fidis. [Online]. Available: http://www.fidis.net/-fileadmin/fidis/deliverables/fidis-wp3-del3.3.study_on_mobile_identity_management.pdf.
[38] Y. Xiang, J. Kennedy, M. Egger, and H. Richter, Network and trust model for dynamic federation, in Proceedings of the Fourth International Conference on Advanced Engineering Computing and Applications in Sciences, Florence, Italy, 2010, pp. 1-6.
[39] Y. Zuo, X. Luo and F. Zeng, Towards a dynamic federation framework based on SAML and automated trust negotiation, in Web Information Systems and Mining (Fu Wang, Zhiguo Gong, Xiangfeng Luo, and Jingsheng Lei, Eds). volume 6318 of Lecture Notes in Computer Science, Berlin / Heidelberg: Springer-Verlag, 2010, pp. 254-262.