To be published in the IFIP IDMAN 2013, the 3rd IFIP WG 11.6 Working Conference on Policies & Research in Identity Management.

Dynamic Identity Federation using Security Assertion Markup Language (SAML)

Md. Sadek Ferdous and Ron Poet

School of Computing Science, University of Glasgow, Glasgow, G12 8QQ, Scotland
m.ferdous.1@research.gla.ac.uk, ron.poet@glasgow.ac.uk

Abstract. Security Assertion Markup Language (SAML, in short) is one of the most widely used technologies to enable Identity Federation among organisations from different trust domains. Despite its several advantages, one of the key disadvantages of SAML is the mechanism by which an identity federation is established. This mechanism lacks flexibility to create a federation in a dynamic fashion to enable service provisioning (or de-provisioning) in real time. Several different mechanisms to rectify this problem have been proposed. However, most of them require a more elaborate change at the core of the SAML. In this paper we present a simple approach based on an already drafted SAML Profile which requires no change of the SAML, rather it depends on the implementation of SAML. It will allow users to create federations using SAML between two prior unknown organisations in a dynamic fashion. Implicit in each identity federation is the issue of trust. Therefore, we also analyse in detail the trust issues of dynamic federations. Finally, we discuss our implemented proof of concept to elaborate the practicality of our approach.

Keywords: Identity Management, Federated Identity Management, Identity Federation, Security Assertion Markup Language (SAML), Trust.

1 Introduction

With the continuous evolution of the online landscape, the number of web-enabled services as well as the user-base expanded rapidly, more and more digital identifiers and credentials were issued to allow users to access those services, and soon their management became challenging, both for organisations and for users. Identity Management (IdM, in short) was introduced by the industry to facilitate online management of user identities, which resulted in various different Identity Management Systems (IMS or IdMS). Formally, Identity Management consists of technologies and policies for representing and recognising entities using digital identifiers within a specific context [14]. There are different types of Identity Management Models such as the SILO Model, Common Identity Domain Model, Federated Model, etc. [14]. Among these models, Federated Identity Management (FIM, in short) has gained considerable attention. It is based on the concept of Identity Federation (also known as Federated Identities or Federation

of Identities). In the ITU-T X.1250 recommendation, a federation is defined simply as "An association of users, service providers and identity providers" [13]. In other words, a federation with respect to the Identity Management is a business model in which a group of two or more trusted parties legally bind themselves with a business and technical contract [9, 17]. It allows a user to access restricted resources seamlessly and securely from other partners from different Identity Domains. An identity domain is the virtual boundary, context or environment in which a digital identifier is valid [17]. FIM offers a good number of advantages to different stakeholders such as separation of duties among different organisations, scalability, improved security and privacy, SSO for users, etc. [5, 9]. Single Sign On (SSO) is the capability that allows users to log in one system and then access other related but autonomous systems without further logins. It alleviates the need to log in to every time a user needs to access those related systems. Using a FIM System, users can take advantage of SSO and thus authenticate themselves in one identity domain and receive personalised services across multiple domains without any further authentication. A FIM System has three different actors:*Identity Provider (IdP)* - responsible for managing digital identities of users and providing identity related services to different Service Providers, *Service Provider (SP)* - responsible for providing web-enabled services to the users and *User (Client)* - receives services from an SP.

The issue of trust is a fundamental concept in FIM as different participating organisations need to trust each other inside the federation to a sufficient level to allow them to exchange user information and trust that information. A detailed analysis of trust requirements for FIM can be found in [15]. To summarise them, the SP needs to trust the IdP to authenticate the user using appropriate security mechanisms and release attributes to the SP as per the contractual agreement. Similarly, the IdP has to trust that the SP will not abuse the released attributes and use them only for the stated purpose as per the agreement. Based on this trust level, users will be granted access to a service or rejected. Thus, organisations inside a federation are said to form the so-called Circle of Trust (CoT).

There are several core technologies available that allow the creation of federations such as Security Assertion Markup Language (SAML) [19], OpenID [6], WS-Federation [22], etc. There are several FIM Systems such as Shibboleth [3], OpenID [6], Microsoft CardSpace [2] that respectively utilise these technologies. Among all of these technologies, SAML has been the most widely used technology for deploying federation that requires strong trust assumptions as well as maintains good security and privacy properties. SAML is an XML-based standard for exchanging authentication and authorisation information between different autonomous security domains. It is based on the request/response protocol in which one party (generally SPs) requests for particular identity information about a user and the other party (IdPs) then responds with the information.

Trust in SAML is established by exchanging metadata of the IdP and the SP and then storing them at the appropriate repositories which helps each party to build up the so-called Trust Anchor List (TAL). This exchange takes place in

out-of-bound fashion after a technical contract between the IdP and the SP is signed and has to be done before any interaction takes place between the said IdP and the SP. A metadata is an XML file in a specified format that plays the central role in SAML. It contains several pieces of information such as entity descriptor (id for each party), service endpoints (the locations of the appropriate endpoints for IdPs where the request will be sent to and for SPs where the response will be consumed), certificate(s) to be used for signing and encryption, expiration time of metadata, contact information, etc. and serves three purposes. Firstly, it allows each organisation to discover the required endpoint of another organisation for sending a SAML request/response. Secondly, the embedded certificate can be used by the SP to verify the authenticity of the SAML Assertion. Thirdly, a metadata behaves like an anchor of trust for each party. During the IdP discovery service at the SP, the list only contains those IdPs whose metadata can be found in its meta repositories (in other words in the TAL) and thus considered trusted. Similarly, IdP will respond only to those requests that are initiated from an SP whose metadata can be found in its metadata repositories (in its TAL). Exchanging and maintain the repositories of metadata and thus managing trust becomes extremely difficult as the number of the IdPs and SPs grows and is a well-known problem of SAML [8]. In addition, pre-configuring trust before any interaction hinders the establishment of a federation between two prior unknown parties in a dynamic fashion. Allowing federations to be created dynamically would open up the door for new business scenarios and novel web-enabled services. There have been several proposals and drafts to handle this situation. Most of these works would either require a considerable amount of change of the SAML protocol or only provide mechanisms for creating federations in a semi-automated fashion. Also the trust issues involved while creating federations in a semi-automated fashion are not thoroughly examined. In this paper, we build on some of the existing works and illustrate mechanisms that can be used to establish federations in a fully automatic fashion. Furthermore we thoroughly explore the complex trust issues involved during such scenarios. The contributions of this paper are:

- We have found that the idea of dynamic federation is still ill-defined. Therefore, we have taken the initiative to define formally some key concepts behind a dynamic federation.
- We extend some existing works so that federations can be created in a fully dynamic fashion.
- We explore the trust issues in such scenarios.
- Finally, we describe our developed proof of concept to illustrate the applicability of our proposal.

With this introduction, this paper is organised as follows. We discuss some existing works on dynamic federation and analyse their strengths and weaknesses in Section 2. Then, we define some key concepts of dynamic federation and explore the trust issues in Section 3. The developed proof of concept is discussed for two different use-cases in Section 4. We present the strengths and weaknesses of

our implementation as well highlight a few potential future works in Section 5. Finally, we conclude in Section 6.

2 Related Work

There have been several works to tackle the problem of scalability and dynamism of SAML. The most influential work, called Distributed Dynamic SAML, can be found in [12] where the authors prescribe that to trust any dynamically exchanged metadata, the metadata must be signed and the X.509 certificate that can be used to verify the signature must be included within the metadata. Assuming a trusted Certificate Authority (CA) issues the embedded certificate, each participating organisation will hold the root CA Certificates which can be used to validate the certificate chain in its TAL. Then, the trust in the metadata can be derived by just verifying the signature in the metadata using the embedded certificate with the traditional PKI. The result of their proposal is the formulation of a working draft of a novel SAML Profile called SAML Metadata Interoperability Profile which is a profile for distributed SAML metadata management system [20, 21]. Based on the proposal and the working draft, an implementation of dynamic SAML has been developed by UNINETT (*https://www.uninett.no/*) in their SimpleSAMLphp project [18, 4]. The SimpleSAMLphp is a native php-based implementation of the SAML protocol stack that allows a SAML IdP or SP to be deployed very quickly. The proposal and the working draft and the SimpleSAMLphp implementation would allow the establishment of federations more quickly than it would be possible previously. However, several crucial questions regarding trust assumptions at different parties have not been explored thoroughly. For example, each party (IdP and SP) validates the certificate of other parties using PKI to establish trust. However, is the established trust enough for any IdP to release sensitive user attributes to an SP which has been added dynamically since there may not be any legal contract between them? And also, since there may not be any legal binding, can the IdP trust that the SP would not abuse the released attribute in any way? Similarly, the SP will need to consider if it can trust any attributes that have been provided by a dynamically added IdP even though the SAML assertion containing those attributes are properly verified.

Furthermore, the SimpleSAMLphp implementation requires that the metadata of the IdP is already present at the SP so that the WAYF (Where Are You From) Service (and IdP discovery service) can display the list of the IdPs to the user. Once the user selects an IdP, a SAML authentication request will be sent to the IdP. If the IdP has the capability to add an SP dynamically (e.g. an IdP deployed with SimpleSAMLphp), it can retrieve the metadata of the SP dynamically and store it temporarily in case the entity ID of that SP is not found in its TAL. To make this possible, the SimpleSAMLphp requires that the entity ID of the SP has to be a URL from where the metadata can be fetched. In summary, the SimpleSAMLphp only allows IdPs to retrieve any SP metadata, not the other way around. We prefer to call it a semi-automatic federation where

the IdP has to be pre-configured at the SP and thus does not fully address the problem of dynamic federation.

There are some other works that also provide proposals for dynamic federations. In [7], the authors propose a SAML extension to accommodate reputation data in XML format. According to their proposal, trust has to be negotiated based on that reputation data before any entity can join the federation. Each entity will maintain a dynamic list called Dynamic Trust List (DTL), instead of the static TAL, which will contain the list of joined entities in the same federation with their reputation data and will be updated dynamically as the federation evolves. To realise their proposal, a novel exchange protocol has to be developed to request and response reputation data. The authors in [25] proposed a dynamic federation architecture, called DFed, based on SAML and Automatic Trust Negotiation (ATN) to establish trust between participating parties in run time. Each DFed party, known as Dynamic Federation Service Providers (DFSP), can act as an IdP and SP. Each DFed consists of different components such as Gate Keeper (GK), Directory Services, Trusted DFSP Repository, SAML Agent and ATN Agent. GK is responsible for the SSO Protocol, Directory Service is responsible for storing attributes and policies, DFSP repository stores the information of federated SPs, SAML Agent is responsible for carrying out the SAML Protocol and ATN agent is responsible for ATN protocol and trust negotiation. All of them function together to realise the Dynamic Federation protocol. It is also clear that DFed also requires that SAML are changed extensively to accommodate the DFed protocols. There is another solution proposed in [24] where the authors propose calculating trust values based on the modified Dijkstra algorithm and to calculate a distributed reputation based on the PageRank algorithm from Google and use the trust and reputation value to create dynamic federations. And like before, this also requires a major change of the SAML Protocol.

Our focus in this work is not to change anything in the core SAML Protocol and therefore we have based our work on the Dynamic SAML. We will extend the existing SimpleSAMLphp implementation so that a federation can be established fully dynamically considering different trust issues.

3 Dynamic Federation

All previously mentioned works have used the term *Dynamic Federation* literally without defining them formally. The lack of any formal definition for Dynamic Federation means that there are scopes for misunderstanding and multiple interpretations. Before we proceed any further, it is therefore essential to define the term *Dynamic Federation* formally which is presented below:

Definition 1. *A Dynamic Federation with respect to the Identity Management is a business model in which a group of two or more previously unknown parties federate together dynamically without any prior business and technical contract with the aim to allow users to access services under certain conditions.*

This definition is a stark contrast with the traditional definition of the identity federation based on SAML in which there must be a legally binding technical

and business contract between participating organisations before they can join any federation. The primary advantage here is the ability to join the federation instantly in real time. However, the lack of any legally binding contract means that organisations must consider that there might be negative consequences involved since no party is bound to behave as it should and therefore take proper precautions. This leads us to the topic of trust which is explored below.

3.1 Trust Issues

According to our proposed definition, participating organisations may not trust each other entirely since they are previously unknown and there is no contract to make them accountable in case there is any unintended incident. The IdPs may not want to release a few sensitive attributes to the SP that has been added dynamically and the SPs may not trust all attributes released by the IdP that has been added dynamically.

Such trust issues have not been considered while drafting the Dynamic SAML which allows any SP to communicate with the IdP and any IdP to communicate with the SP without any sort of verification. Remember that, technically speaking, joining a federation in the Dynamic SAML will just require the parties to exchange and store the respective metadata with each other. The SimpleSAMLphp implementation based on the Dynamic SAML only allows an SP to join with an IdP since it requires the pre-configuration of the IdP in the SP TAL to allow any user to choose that IdP. However, to harness the true potential of dynamic federation, we need to ensure that both parties can be added dynamically. Moreover, the IdP in SimpleSAMLphp does not distinguish between statically and dynamically added SPs. This allows the IdP to release the same level of (sensitive) attributes to both types of SPs. In summary, we have two requirements to fulfil: i) Fully automate the joining procedure in a federation for both parties and ii) Ensure that some sort of trust is established in a dynamic federation. With these two goals in minds we introduce the notion of fully trusted, semi-trusted and untrusted entities.

Definition 2. ***Fully Trusted Entities****. Fully trusted entities are the IdP and SP in the traditional SAML federation in which there is a legal contract between the IdP and the SP. They are so called since each IdP (or SP) inside a federation trusts any SP (or IdP) in the same federation to behave as intended and can be made accountable in case the other party behaves maliciously.*

Definition 3. ***Semi-trusted Entities****. Semi-trusted entities are the SPs in a dynamic federation that have been added dynamically to an IdP inside the federation under* ***some conditions*** *without the presence of any contract between them and to whom any user(or users) of the IdP has(have) agreed to release a subset of her(their) attributes. They are so called since the user wants to release a subset of their attributes to these SPs inside the dynamic federation even though the IdP in the same federation may not fully trust such SPs to behave as intended. Therefore, such SPs might not be made accountable by the IdP in cases the they behave maliciously with the absence of any contract between them.*

Definition 4. ***Untrusted Entities****. Untrusted entities are the IdP and SP in a dynamic federation in which they have been added dynamically under* ***some conditions*** *without the presence of any contract between them. They are so called since each IdP (or SP) inside a dynamic federation may not trust at all any other dynamically added SP (or IdP) in the same federation to behave as intended.*

It is important to understand that a dynamic federation may accommodate as many fully trusted entities as possible. As such, a dynamic federation is an extension of the traditional federation.

Now, the term "some conditions" in the definition of the semi-trusted and untrusted entities require further explanations. It can be the combinations of several different conditions by which an SP can be added dynamically to the IdP and vice versa, the conditions for establishing individual trust with each other in such a federation, the condition by which attributes are released to a semi-trusted SP and the condition by which an SP treats attributes of a user from an untrusted IdP. Semi-trusted and untrusted entities of different dynamic federations should have different sets of conditions suitable for their business models and service provisioning scenarios. Here, we present a set of conditions that we have assumed for developing our proof of concept of dynamic federation using SAML.

- Only a valid user of an IdP is allowed to add an SP to that IdP dynamically. This is to ensure that only those SPs that the users want to access for service provisioning are added in a dynamic federation. This is missing in the current implementation of SimpleSAMLphp.
- Once the SP is added to the IdP, the SP must add the IdP to its TAL to ensure that the user can select the IdP next time. This nullifies the need to pre-configure the IdP in the SP.
- Dynamically added SPs must be tagged as untrusted entities in the IdP at the initial stage. Only when a user, after being authenticated at the IdP, has agreed to release a subset of her attributes to the SP, the SP should be re-tagged as a semi-trusted entity.
- A dynamically added IdP should always be tagged as an untrusted entity for the SP.
- IdPs should ensure that it does not release any attributes to an untrusted entity.
- IdPs should ensure that some crucial and sensitive attributes are not released to any semi-trusted entity since there is no guarantee that such attributes will be handled as intended. Therefore, it should allow their administrators to configure what attributes should be released to a semi-trusted entity.
- It is up to the discretion of each SP how they want to treat released attributes from an untrusted IdP. They could use the NIST LoA (Level of Assurance or Level of Authentication) guidance of 1 to 4 where Level 1 conforms to the lowest assurance and 4 conforms to the highest assurance [16]. Usually, the LoA level comes from the IdP and is embedded inside a SAML assertion to provide the level of assurance for a certain authentication mechanism at the

IdP. However, the SP should consider implicitly that LoA 1 is the maximum that can ever accompany the SAML authentication and attribute statements from any untrusted IdP. Since, how the released attributes will be handled depends on the individual SP, it can vary from one SP to another even inside the same federation.

To summarise, we propose that entities have to be federated in a fully automatic fashion without any human intervention to harness the full potential of dynamic federation and while doing so all entities must consider trust issues involved. The conditions outlined before are one of the many ways to fulfil our proposals.

4 Proof of Concept

In this section we will discuss the proof of concept that we have developed to illustrate the applicability of our proposals. We have used SimpleSAMLphp and modified it to meet our requirements. The following subsections will consider two different use-cases: i) IdP-SP Scenario and ii) IdP-IdP-SP Scenario.

4.1 IdP-SP Scenario

The architectural setup for this scenario is that there are one IdP and one SP deployed with the modified version of the SimpleSAMLphp. At the beginning, the IdP and SP are not part of a common federation (they individually may be part of separate federations) and they have no prior knowledge of the other party whatsoever. The IdP is configured to use a MySQL database at its end to store user attributes including username and password in a table called users. In addition, the IdP uses two new tables called *semitrusted and untrusted* to store the entity IDs of semi-trusted and untrusted SPs respectively. Similarly, the SP is also configured to use a MySQL database at its end where it uses a new table called *untrusted* to store the entity IDs of IdPs that have been federated dynamically.

In addition, we also need to provide a mechanism by which an admin of the IdP can configure which attributes to release to a semi-trusted SP. We have opted in for a configuration parameter called *semitrusted.sp* which can be added to the configuration file (called config.php) in the SimpleSAMLphp. A sample configuration parameter could be *'semitrusted.sp'=> array ('username', 'name', 'telephone', 'age', 'position', 'org')* which will configure the IdP to release only these attributes by excluding all other attributes such as *salarygrade & email* attributes, which the IdP normally releases to all trusted SPs but assumes to be too crucial and sensitive, to a semi-trusted SP. The admin can add as many or as less attributes as needed as per the requirements. This configuration parameter works like an attribute release policy. SimpleSAMLphp does not have the concept of an Attribute Release Policy, however, it has something similar called an Authentication Processing Filter (AuthProc) [1]. An AuthProc allows the system to do something extra once the user authentication is done. For example,

among other things, it can be used for filtering out attributes once the authentication is done. There are several authentication processing filters bundled with the SimpleSAMLphp implementation. One of them is the Consent module that is used to display the list of attributes to the user just before they are released. We have modified this module to allow the IdP to show only those attributes that can be found on the *semitrusted.sp* parameter. From the displayed list, the user can choose which attributes she wants to release to the SP.

With this setup, the protocol flow for this scenario is given below.

1. A user visits the SP for the first time to access one of its service. Since there is no security context (e.g. no cookie) of the user at the SP, the user needs to be authenticated at an IdP and therefore she is redirected to the WAYF service and a list with federated IdPs with the SP is shown.
2. Since the IdP and SP are not part of a common federation, the IdP list at the SP does not contain the IdP. However, since the SP supports (more precisely the SimpleSAMLphp that has been used to deploy the SP supports) our proposal of dynamic federation, it contains two additional text fields (Figure 1) which allow the user to enter the entity ID of her IdP and a code to ensure that only the valid users of the IdP have the ability of federating an SP with the IdP.

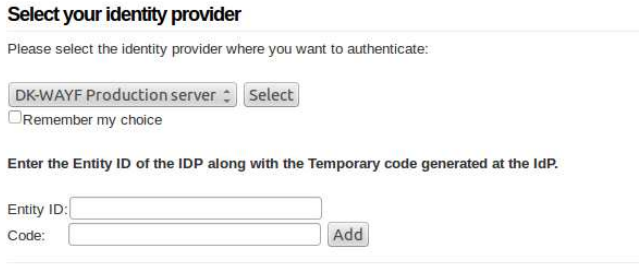

Fig. 1. Additional Text Fields at the WAYF.

3. Since the user does not have the code, she logs in to her IdP and generates a code using the Generate button at the Generate IdP Code page. This page also checks the *semitrusted* table to see if there is any dynamically added SP. Since there is none now, it says so. Once the Generate button is clicked, a 4 digit random number is generated and displayed. This random number is also stored temporarily in a database table called *code* which is used during metadata exchange for verification (see below). Here, we have opted to generate a 4 digit code, other implementations may opt for other type of codes according to their own requirements.
4. After generating the code, the user inserts the entity ID of the IdP and the generated code to the WAYF page of the SP and clicks the Add button.
5. Once the Add button is clicked, it is verified that entity ID field or code is not null and that the inserted entity ID is not already part of the federation (dynamically or statically). If any part of the verification process fails,

an appropriate error message is displayed and the user is redirected to the WAYF page where she can insert valid values again.

6. Assuming there is no error during verification, a request to retrieve metadata from that entity ID with some specific values is posted. The request contains the entity ID and the code that the user has entered and two hidden fields called *MetaAdd & ReturnTo*. The value of the *MetaAdd & ReturnTo* fields contain the entity ID of the SP and the URL of the services that the user requested at the first place which initiated the SAML flow. Remember that SimpleSAMLphp implementation of dynamic SAML requires that entity ID be the endpoints from where the metadata can be fetched. We have extended its approach to add verification.
7. Once the Appropriate end point of the IdP receives this request, it checks if there is any field called *MetaAdd*. If found, it knows that this is a special request for exchanging metadata with the requested SP. If not found, it assumes that it is normal metadata fetch request and returns its metadata. Since the request contain a *MetaAdd* field, it, then, checks for a code field and retrieves its value and verifies if the same value can be found at the code table of its MySQL database. If found, it indicates that this request for exchanging metadata is valid. If not found, an error message is returned to the SP.
8. Assuming that the code field contains a valid code, the IdP retrieves the value of the *MetaAdd* field which contain the entity ID of SP and sends an HTTP GET request to that location. GET has been used since there is no other parameters to pass during the metadata fetch process from the SP.
9. Once the metadata is retrieved, the IdP goes thorough the specified verification process for a dynamic SAML (verifying the embedded certificate, verifying the signature on the metadata using that certificate, etc.). If the verification is correct, the metadata is stored in its repository and the SP is added to its TAL list. In addition, the entity ID of the SP is added into the *untrusted* table of the database along with the code and the used code is removed from the *code* table to ensure that it cannot be reused again. If the metadata verification is not correct, an error message is returned to the SP.
10. If everything goes right at the IdP, the metadata of the IdP along with the *(*ReturnTo) field and its value are returned to the requesting endpoint of the SP, where the metadata and *ReturnTo* values are separated. As before, the SP goes thorough the specified verification process for a dynamic SAML. If verification is correct, the metadata is stored in its repository and the IdP is added to its TAL list. In addition, the entity ID of the IdP is added into the *untrusted* table of its database and thus the IdP is tagged as an untrusted IdP.
11. Then, the user is redirected to the URL retrieved from the *ReturnTo* field (the URL of the service requested initially) which in turn takes the user back to the IdP selection page of the WAYF. However, as the IdP has already been added, the list contains the list of the IdP and it is tagged as an untrusted IdP (Figure 2). Moreover, it is shown to the user that the IdP has already

been added as an untrusted IdP so that no other users tries to add it once again which will result in an error.

12. Now, if the user selects the IdP, the usual SAML flow will take place. A SAML authentication request will be sent to the selected IdP and if the user is not already authenticated at the IdP, she will be prompted for login.

Fig. 2. Added IdP at the WAYF.

13. Once the user is logged in, the Consent module is called internally. It reads the *semitrusted.sp* configuration parameter and displays the attributes automatically. Figure 3 shows the consent form. The consent page also states which attributes are filtered out from the full set and why. At this point, the user can choose which attributes she wants to release to the SP.

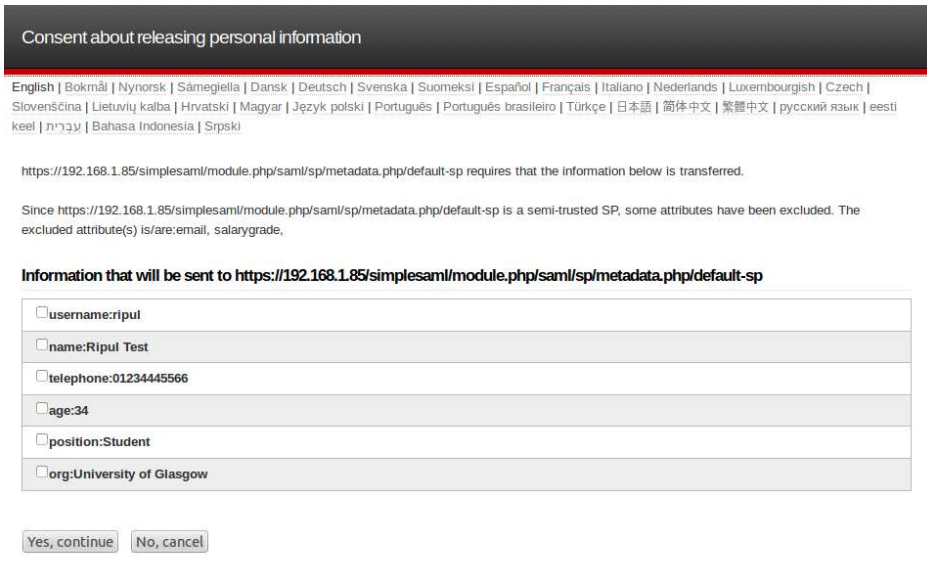

Fig. 3. Attributes to be released at the Consent page.

14. If the user has chosen to release any attribute(s) by clicking the 'Yes, continue' button, the entity ID of the SP is removed from the *untrusted* table

and inserted into the *semitrusted* indicating that the SP will be tagged as a semi-trusted entity hereafter. If the user chooses not to release any attribute, the entity ID of the SP will remain in the *untrusted* table. At this point, the chosen attributes would be released to the SP.

At this point, the SP knows from the *untrusted* table of its database that these attributes have been released by an untrusted IdP. Therefore, the respective SP will treat these attributes coming from an IdP having LoA level of maximum 1 and authorise the user accordingly.

4.2 IdP-IdP-SP Scenario

The protocol flows described above will allow the creation of federations in a fullly dynamic fashion and also consider the trust issued involved at both ends. However, the main problem is that the SP may not trust all attributes coming from the untrusted IdP even though the IdP is honest. The problem can be resolved if it is possible to link the untrusted IdP with an IdP which is fully trusted by the SP. In such a case, the fully trusted IdP would act like a Proxy IdP as described in [10]. The SP would think that it is talking to the fully trusted IdP while in fact the proxy IdP would delegate the authentication service to the untrusted IdP which is hidden from the SP. The untrusted IdP would release the user attributes to the fully trusted IdP once the user is authenticated which will be then retrieved and returned to the SP in such a way that the SP will think that the attributes have been released by the fully trusted IdP. Another advantage is that the SP no longer needs to support dynamic federations, since the proxy IDP can, in a trustworthy manner, add new IDPs indirectly. As before, we have used SimpleSAMLphp to demonstrate this scenario. The SimpleSAMLphp allows multiple authentication sources (including another SAML IdP) for authenticating a user. This can be enabled by the *MultiAuth* module of SimpleSAMLphp. We have used this feature to add the untrusted IdP as one of the authentication sources for the fully trusted IdP. To the untrusted IdP, the fully trusted IdP would be treated as a normal SAML SP, however, the fully trusted IdP would treat the untrusted IdP as the SAML IdP. To enable this configuration, it needs the authentication source to be pre-configured by exchanging metadata just like a federation. Like the previous scenario, we have modified the SimpleSAMLphp to automate this procedure so that two prior unknown SAML IdPs can be linked (in other words federated) in a fully dynamic fashion. However, this approach introduces an inconsistency which a malicious user might abuse for elevation of privileges. Since the SP would think that the attributes from the linked IdP have come from a trusted source (the proxy IdP), they might be tricked in trusting them. To ensure that it does not happen, the trusted IdP must add a LoA value of maximum 1 into the assertion in cases it has received any attributes from a dynamically linked IdP to them and return the assertion with that lower LoA. This would help the SP to decide how much trust they can put in those attributes.

With this setup, the protocol flow for the scenario is given below.

1. The user logs in to the untrusted IdP and generate a code just like the previous one. This code will be used to link the proxy IdP with this IdP.
2. Now, the user needs to log in to the proxy IdP. Since the IdP has enabled the *MultiAuth* module, the IdP shows all authentication sources (Figure 4). As there is no other authentication sources, it only shows the Userpass source which allows the user to log in to the IdP by using Username/password. The user selects this source and logs in using her username and password. After login, the user clicks the 'Link Another IdP' option.

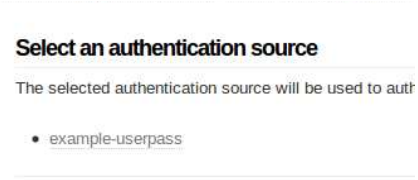

Fig. 4. Only Username/password source at the IdP.

Fig. 5. Linked untrusted IdP as the authentication source in the proxy IdP.

3. The user is presented with a page which has three fields: IdP ID field for entering the entity ID of the untrusted IdP, Code field to enter the generated code from step 1 and a Name field to enter a *Nick Name* for the untrusted IdP. This *Nick Name* will help the user to remember the IdP once it is added as one of the authentication sources. This field is not needed at the SP because each IdP is listed using its entity ID, whereas at the proxyIDP the IdPs are listed as authentication sources using a user friendly name. After entering all this information, the user clicks the Submit button.
4. Once the Submit button is clicked, it is checked to make sure that all information has been entered in the three fields. If not, appropriate error messages are displayed. If yes, a request to retrieve metadata from that entity ID is submitted with some specific values just like the way discussed in the previous scenario.
5. At this point, code is verified, metadata between two IdPs are exchanged, verified and stored just like the previous scenario. At this point, the two IdPs are linked.
6. Now, the user visits the SP to access one of the services. Assuming there is no security context, the user is redirected to the WAYF Service where the list of federated IdPs are shown. The user selects the fully trusted IdP (proxy IdP) and a SAML Authentication request is submitted to that IdP.
7. The IdP displays the list of authentication sources. As the untrusted IdP is already linked, the user can see the nick name (My IdP) of the untrusted IdP which was given during the linking phase (Figure 5).
8. Once the user selects the untrusted IdP, she is redirected to the IdP where the user logs in and the consent page with attributes are shown. As the proxy IdP is not tagged as the semi-trusted SP, all attributes that the user chooses will be released to the Proxy IdP.
9. Once the user clicks the 'Yes, continue' button, a SAML assertion with all attributes is sent back to the proxy IdP where all attributes are retrieved.

The proxy IdP builds a SAML assertion with these attributes with a LoA value of 1 and it is sent back to the SP.

What the SP will do with all these attributes is not further explored here.

5 Discussion & Future Work

Creating dynamic federations using our approach has several advantages:

- Users can create federations just in time and whenever required. Even though it was not considered in our implementation, any IdP or SP can decide on how long it would allow the other party that has been added dynamically to remain in the federation by using a time threshold. Once the threshold is reached, the respective entry can be removed automatically from the TAL list, thus de-federating the entity.
- By using separate trust domains inside the same federation for fully trusted, semi-trusted and untrusted entities, a federation can host all types and leverage the advantages of all in the same configuration. However, one must keep in mind that the types of treatments semi-trusted entities will receive will fully depend on a particular implementation since the behaviour is not standardised.
- One of the requirements for an ideal IMS is the Segregation of power which is required to ensure that no single entity will have dominant position over other entities and users have the ability to choose a specific entity for a specific scenario [23, 11]. Since the traditional SAML enforces users to use only those IdPs that can be in the TAL, segregation of power is not fully exercised [11]. Allowing the dynamic creation of federations would allow the users to choose a specific IdP for a specific scenario and hence ensuring the segregation of power.
- Allowing users to link two of their IdPs would enable any users to use their own IdPs (e.g. a locally installed OpenID Provider) via a trusted IdP. This would help to aggregate attributes from different sources. For example, the local IdP may provide some dynamically created attributes (e.g. location data etc.) which would be difficult for the fully trusted IdP to provide.

However, it should be kept in mind that the IDP-IDP linking (the second use case) means that a highly trusted (Proxy) IDP which can normally issue assertions at high LOAs, has to issue all its assertions at LOA 1 when the user authenticated via a linked IDP. This is not ideal, and may thus lessen the value of the service.

There are a few directions to take from here. The current implementation requires that both SAML IdPs are online for exchanging metadata during the linking procedure. To enable a user to link her local SAML IdP (IdP residing in the user's PC) with the trusted IdP, we need to find a way for the trusted IdP to communicate with the local IdP for exchanging metadata. It could be another interesting topic to investigate the ways different dynamic attributes could be aggregated from different dynamically added IdPs.

6 Conclusion

In this paper, we have explored the avenue of the dynamic SAML. We have provided a proposal for creating dynamic federations in a fully automatic fashion. Our approach is simple in nature and can be easily adopted by any SAML implementation and requires no modification of the SAML Protocol. We have also examined the trust issues involved in such scenarios and illustrated two use-cases with detailed protocol flows. However, it should be noted that the issues of trust are very complex. The way we have outlined the trust issues may not be suitable for all scenarios. For example, an IdP may be reluctant to trust any SP which is not pre-configured in the traditional static way and thereby hesitant to release any attributes to it. In such cases, it will be very difficult to create federations in a dynamic way. We believe that IdPs and SPs will need to relax the trust requirements if they want to allow their users to take advantage of the dynamic federation. How much relaxation it will be required depends entirely on a specific use-case and will dictate the positive effect a dynamic federation can bring to their users.

References

1. Authentication processing filters in simplesamlphp. `http://simplesamlphp.org/docs/stable/simplesamlphp-authproc`.
2. Microsoft Windows CardSpace. `http://www.microsoft.com/windows/products/winfamily/cardspace/default.mspx`.
3. Shibboleth. `http://shibboleth.internet2.edu/`.
4. SimpleSAMLphp. `http://simplesamlphp.org/`.
5. Liberty Alliance Whitepaper: Benefits of Federated Identity to Government, March 2004. `http://projectliberty.org/liberty/content/download/388/2723/file/Liberty\Government\Business\Benefits.pdf`.
6. OpenID Authentication 2.0 - Final. 5 December, 2007. `http://openid.net/specs/openid-authentication-2\0.html`.
7. Patricia Arias Cabarcos, Florina Almenrez Mendoza, Andrs Marn-Lpez, and Daniel Daz-Snchez. Enabling saml for dynamic identity federation management. In Jozef Wozniak, Jerzy Konorski, Ryszard Katulski, and Andrzej Pach, editors, *Wireless and Mobile Networking*, volume 308 of *IFIP Advances in Information and Communication Technology*, pages 173–184. Springer Boston, 2009.
8. Mortaza S. Bargh, Bob Hulsebosch, and Hans Zandbelt. Scalability of trust and metadata exchange across federations, December 2010. `https://tnc2011.terena.org/getfile/693`.
9. David W Chadwick. Federated Identity Management. In A. Aldini, G. Barthe, and R. Gorrieri, editors, *FOSAD 2008/2009*, number 5705 in LNCS, pages 96–120. Springer-Verlag, Berlin, January 2009.
10. David W. Chadwick, George L. Inman, Kristy W.S. Siu, and Mohammad Sadek Ferdous. Leveraging social networks to gain access to organisational resources. In *Proceedings of the 7th ACM workshop on Digital identity management*, DIM '11, pages 43–52, New York, NY, USA, 2011. ACM.
11. M.S. Ferdous and R. Poet. A comparative analysis of Identity Management Systems. In *High Performance Computing and Simulation (HPCS), 2012 International Conference on*, pages 454 –461, july 2012.

12. P. Harding, L. Johansson, and N. Klingenstein. Dynamic security assertion markup language: Simplifying single sign-on. *Security Privacy, IEEE*, 6(2):83 –85, march-april 2008.
13. ITU-T. Baseline capabilities for enhanced global identity management and interoperability, September 2009. `http://www.itu.int/ITU-T/recommendations/rec.aspx?rec=X.1250`.
14. Audun Jøsang, Muhammed Al, and Zomai Suriadi Suriadi. Usability and privacy in identity management architectures. In *ACSW '07: Proceedings of the fifth Australasian symposium on ACSW frontiers'*, pages 143–152, 2007.
15. Audun Jøsang, John Fabre, Brian Hay, James Dalziel, and Simon Pope. Trust requirements in identity management. In *Proceedings of the 2005 Australasian workshop on Grid computing and e-research - Volume 44*, ACSW Frontiers '05, pages 99–108, Darlinghurst, Australia, Australia, 2005. Australian Computer Society, Inc.
16. NIST. Electronic authentication guideline: Information security, April 2006. `http://csrc.nist.gov/publications/nistpubs/800-63/SP800-63V1_0_2.pdf`.
17. Md. Sadek Ferdous, Mohammad Jabed, Morshed Chowdhury, Md. Moniruzzaman, and Farida Chowdhury. Identity federations: A new perspective for bangladesh. In *Informatics, Electronics Vision (ICIEV), 2012 International Conference on*, pages 219 –224, may 2012.
18. Andreas Solberg. Dynamic SAML. 18 February, 2010. `https://rnd.feide.no/2010/02/18/dynamic_saml/`.
19. OASIS Standard. Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0. 15 March, 2005. `http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf`.
20. OASIS Standard. A profile for distributed SAML metadata management. 22 October, 2007. `https://spaces.internet2.edu/display/dsaml/A+profile+for+distributed+SAML+metadata+management`.
21. OASIS Standard. SAML V2.0 Metadata Interoperability Profile, Working Draft 01. 1 August, 2008. `https://spaces.internet2.edu/download/attachments/11275/draft-sstc-metadata-iop-01.pdf?version=2&modificationDate=1217876016355`.
22. OASIS Standard. Web Services Federation Language (WSFederation) Version 1.2. 22 May, 2009. `http://docs.oasis-open.org/wsfed/federation/v1.2/os/ws-federation-1.2-spec-os.pdf`.
23. Future of Identity in the Information Society WP3. Study on Mobile Identity Management, May 2005. `http://www.fidis.net/fileadmin/fidis/deliverables/fidis-wp3-del3.3.study_on_mobile_identity_management.pdf`.
24. Y. Xiang and J. Kennedy amd M. Egger amd H. Richter. Network and trust model for dynamic federation. In *Proceedings of the Fourth International Conference on Advanced Engineering Computing and Applications in Sciences*, pages 1–6, 2010.
25. Yicun Zuo, Xiling Luo, and Feng Zeng. Towards a dynamic federation framework based on saml and automated trust negotiation. In Fu Wang, Zhiguo Gong, Xiangfeng Luo, and Jingsheng Lei, editors, *Web Information Systems and Mining*, volume 6318 of *Lecture Notes in Computer Science*, pages 254–262. Springer Berlin / Heidelberg, 2010. 10.1007/978-3-642-16515-3_32.