A Conceptual Model for Attribute Aggregation

David W. Chadwick[1], George Inman[1], Nate Klingenstein[2]
[1]University of Kent, UK. [2] Internet2 Consortium, USA

Abstract

This paper describes a conceptual model for attribute aggregation that allows a service provider (SP) to authorise a user's access request based on attributes asserted by multiple identity providers (IdPs), when the user is known by different identities at each of the IdPs. The user only needs to authenticate to one of the IdPs and the SP is given an overall level of assurance (LoA) about the authenticity of the user and his/her attributes. The model employs a new component called a Linking Service (LS), which is a trusted third party under the control of the user, whose purpose is to link together the different IdP accounts that hold a user's attributes, along with their respective LoAs. There are several possible interaction models for communications between the IdPs, the SP, LSs and the user, and each are described. The model is underpinned with a fully specified trust model, which also describes the implications when participants do not fully trust each other as required. Finally, the paper describes how the model has been implemented by mapping onto existing standard protocols based on SAMLv2.

1. Introduction

Many organisations are experimenting with virtual organisations (VOs) and federations. Practical examples abound, such as the Tera-Grid VO [1], the In-Common Federation [2] and the UK Access Management Federation for Education and Research [14]. Microsoft has added identity federation into its latest Vista operating system with Cardspace [3]. Whilst user authentication and authorisation was originally based on globally unique user identities, such as X.500 distinguished names held in X.509 public key certificates, more recently authentication is being based on federated identities [7], and authorisation is being built upon the Role or Attribute Based Access Control models (RBAC/ABAC), for example as exemplified in [15, 16]. The typical modus operandi of the latest federated systems is that the user authenticates to an Identity Provider (IdP), and the IdP sends an authentication statement and authorisation attributes to the Service Provider (SP). The SP then grants access based on the user's attributes. Note that only one IdP and one set of user identity attributes are typically involved in this exchange. However, most users have attributes assigned by a number of different authorities or IdPs, for example, the General Medical Council in the UK says who are doctors, organisations say who are their employees and what their roles are, VO managers say who are their VO members, whilst learned societies such as IEEE say who their members are, etc. Unfortunately most VO and federated systems currently suffer from a significant limitation, namely, the lack of a standard approach to aggregating user attributes, asserted by multiple authorities, for the SP to use in its access control decision making. Ad-hoc solutions are currently being experimented with, such as Grid-Shib [4], myVocs [5] and D-Grid [16]. Some of these solutions, such as Grid-Shib and D-Grid are only capable of aggregating attributes from two authorities, namely the VO manager and the user's organisation. myVocs is an alternative solution that places a myVocs IdP-SP server in-between the real IdP and the real SP. The myVocs server can hold a set of VO specific attributes which it can

aggregate with the IdP's attributes. Different IdPs can be involved. But myVocs has severe limitations in its trust model. It requires the SP to trust the myVocs server to both authenticate all users correctly, and to aggregate all users' attributes correctly. The SP has no assurance about the authentic source of any of the user's attributes since myVocs appears to be the authoritative source of all of them. In comparison, in this paper we propose a conceptual model and a standard protocols based solution to the problem of attribute aggregation, in which a user's attributes can be aggregated from any number of IdPs, whilst maintaining user privacy and giving the SP assurance about the authoritative sources of all the attributes.

A couple of use cases might help the reader to envisage why attribute aggregation is needed:
i) Accessing Electronic Medical Records (EMRs). Only qualified and registered health care professionals can access EMRs. In addition, these professionals have to be employed by a local health authority and be currently on duty. The EMR application needs to aggregate attribute assertions from the national professional database, the local health authorities and the duty roster system.
ii) Online Purchasing with Membership Discount. In order to purchase a mobile phone contract online and obtain a student discount, a user has to prove she is a registered student, has a good credit record, and has a credit card from an issuing bank. The online store needs to aggregate attribute assertions from the user's university and bank and a credit rating bureau.

Before we developed our conceptual model we gathered a set of user requirements for attribute aggregation, primarily from the academic networking community. The user requirements were obtained by widely circulating a structured questionnaire[1] to many email lists. The results were first presented in [6] and are summarised in section 2 below. Section 3 defines the conceptual model that satisfies most of the user requirements. Note that it is not possible to simultaneously satisfy all the user requirements since some of them are mutually exclusive, such as the desire to support multi-hop proxying without knowing who the ultimate end-entity is, and the requirement to have attribute assertions digitally signed by their authoritative sources. Section 4 describes the trust model that the conceptual model requires. Section 5 describes the mapping of the conceptual model onto existing standard protocols based on the Security Assertions Markup Language (SAML) version 2 from OASIS [8]. Section 6 concludes and indicates our next steps in this project.

2. User Requirements for Attribute Aggregation

The following requirements were seen to be important for any new multi-source attribute authorisation system by the majority of the questionnaire respondents:
General requirement

1. Attribute aggregation must be usable in a variety of ways: Humans via web browsers, Applications via APIs and Grid users via grid clients etc.

Privacy related requirements

[1] Questionnaire is available at http://sec.cs.kent.ac.uk/shintau/pages/requirements.html

2. Privacy protection of user attributes is of high importance and this should be through the use of technical controls, which are independent of legal means.
3. Service Providers should be able to track users between sessions if required
4. Service Providers should be able to learn the true identity of users in exceptional circumstances, but only by contacting the user's IdPs.

User consent requirements

5. IdPs and SPs should only be able to communicate with each other to link together the attributes of a user with the user's consent.
6. Service providers should only be able to query multiple IdPs, in order to pull additional attributes for authorisation purposes, with the user's consent.

Protocol Requirements

7. The protocols should be able to tunnel through firewalls using existing open ports (i.e. use http/https).
8. The system should use existing standard protocols and only extend them in a standard way if necessary. SAML is the most popular choice for the base protocol.
9. The proxying of information should be supported through multiple hops/proxies.

Trust requirements

10. The optional ability to sign all assertions should be supported for all message exchanges.
11. The SP should be able to require that all assertions are signed by their authoritative sources.

Usability requirement

12. It should be easy to use by end-users and require the minimum amount of user interaction[2]

As we describe the conceptual model below we will show how most of these user requirements have been met.

3. The Conceptual Model

Before describing the attribute aggregation conceptual model, we initially need to describe the concept of level of assurance (or authentication).

3.1. Level of Assurance (LoA)

The level of assurance (or level of authentication) that is provided by an authentication service, indicates the amount of reliance that a relying party can have on the identity of the authenticated user. Identity Providers may indicate the LoA when issuing authentication assertions to Service Providers (SPs). NIST in [11] describes four levels of assurance, ranging from 1 to 4, with 4 being the strongest. Asserting an LoA of 4 requires that the user's identity has been verified physically during registration, using official documents such as a passport and birth certificate, and that online authentication is carried out using strong cryptography where the user's key is held in a tamperproof hardware device. Asserting an LoA of 1 does not require the user to have been physically identified during registration, but it does assert that it is the same online user each time

[2] This last requirement was not part of the user requirements questionnaire, but was mentioned by at least one respondent as additional requirements. In our opinion it should be a "given" for any system that is to gain wide acceptability.

(regardless of who this user actually is). An LoA of zero would indicate that no registration or authentication has been carried out whatsoever i.e. the online user could be any member of the public. The NIST guidelines specify the levels of assurance based on the combined strengths of the registration and authentication phases. If either are weak, then the asserted LoA must be correspondingly low.

Our model requires that IdPs use the LoA as follows. When each IdP initially registers a user, it will do so at a particular LoA value (which we call the registration LoA). This will be based on the level of identity checking that is performed by the IdP during registration. Self asserted attributes, where the user states his name, age, address, qualifications etc. and no checks are performed by the IdP, are given a registration LoA of 1.

The IdP may support a variety of authentication mechanisms which have different strengths (we call this strength the authentication LoA). When a user authenticates to an IdP during a service session, she will be authenticated using a particular authentication mechanism (which has an associated authentication LoA), and allocated a session LoA which is the lowest of the authentication and registration LoAs. Consequently, regardless of how strong an authentication mechanism might be, the registration LoA is the maximum value of LoA that can be assigned to a session LoA. In other words, a user can never claim attributes at a higher LoA in a user-SP session than that which was used during her initial registration of those attributes with the IdP. However, a session LoA can be lower than the registration LoA, depending upon the strength of the authentication mechanism that is used.

3.2. High Level Model Overview

Our attribute aggregation model is based on the following assumptions:

- the user has already registered with a number of IdPs, and has been assigned various attributes by each of them. The user will usually be known by a different identifier at each IdP, which either the IdP or the user may have chosen;
- only the user knows about all his IdP accounts. None of the IdPs know if the user has other accounts at the same or other IdPs.
- each SP and IdP has a bilateral trust relationships which allows them to communicate successfully with each other. The SP trusts the IdP to correctly authenticate the user to a particular session LoA and to correctly assert the attributes that belong to the user. The IdP trusts the SP not to misuse the attributes that it is given.

The first step in attribute aggregation is for the user to somehow explicitly link together his various accounts from his different IdPs. This satisfies requirement 5 (user's consent). We had a design choice as to whether the account linking process should be dynamic or static, and transient or permanent. Dynamic-transient mode means that the linking is done during each service request and is forgotten immediately after the service has been performed. Static-permanent mode means that the linking is performed prior to any service request and can then be used by any number of subsequent service requests. Dynamic-permanent mode means that the linking is established during a service request,

but is remembered for use in subsequent service requests. Static-transient mode is an oxymoron and wont be considered further.

Dynamic-transient mode can be implemented today by SPs using some, but not all[3], existing standard protocols and models, for example Shibboleth [7]. The SP simply has to ask the user to authenticate to multiple IdPs during the same session, and obtain attributes from each of them. Although one can argue that this is inconvenient to the user, since it requires the user to authenticate to multiple IdPs, it does not require a new conceptual model, or any new services, and so this mode won't be considered further.

If the linking process is modular and self contained within a new service, then it should be able to be performed either before a service request is made or partway through processing a service request, in which case we can implement both static-permanent mode and dynamic-permanent mode using the same model and protocols. This is the approach we have taken, by using a new Linking Service to permanently record the user's linked IdP accounts.

There is another dimension to consider for any permanently linked accounts and this is service activation. A user may link several identity accounts together before any service request is made, but then statically or dynamically determine which of these linked accounts should be used in any given service session. A user almost certainly will not want every SP to receive the same set of attributes from the same set of linked accounts. On the other hand, the user may want the same SP to receive the same set of linked attributes each time. Thus in order to satisfy requirement 6 the user should be asked at the start of a service request if he wishes to use linked accounts, and also have the ability to choose which of his linked accounts he wishes to use in this session, rather like a user choosing which credit card and frequent flyer card to present to a hotel at registration.

In order to have (semi-)permanently linked identity accounts which are under the control of the user, we define a new identity management component, which we call a **Linking Service** (LS)[4]. The LS is a trusted third party (TTP) that is used to link a user's IdP accounts together. Note that the LS does not link the actual identities or the user's attributes together, but rather links together the IdP accounts that hold the attributes. In this way trust in the LS is minimised since the LS has no knowledge of which identities and attributes each linked IdP holds, thereby maximising protection of the user's privacy. During the linking process the user logs in to the IdP accounts that he wishes to link together, indirectly informing the IdP that he wishes to link the account's attributes to those held by other IdPs. After linking has been established via the LS, the user contacts a

[3] The Information Card model and CardSpace protocols are currently not able to support dynamic and transient linking, so this model does need enhancing to support this mode.

[4] Whilst the LS is conceived as an independent trusted third party, implementations may choose to integrate the functionality of the LS in their IdP or SP software, in which case the communications between the LS and the integrated component would be internal to the application (and trusted). This would simplify both the trust model and network communications that are needed, but would increase the trust that others would need to place in the combined entity. Alternatively each user could operate their own LS on their desktop, in which case the LS would only link together the user's own accounts. The trust that IdPs and SPs would need to have in this LS would be correspondingly lower.

SP with a service request. The SP redirects the user to his chosen IdP for authentication as now (e.g. by using a Where Are You From service, or by proprietary means). The redirection may be direct to the chosen IdP (static-permanent mode), or indirectly via the LS (dynamic-permanent mode).

After authentication, the various components (IdPs, LSs and SP) communicate directly with each other to aggregate the user's attributes. Aggregation can be performed by either the SP (SP aggregation) or the LS (LS aggregation). Either way, the SP ends up with an authentication statement about the user from the chosen IdP, plus a set of attribute assertions, each signed by the authoritative IdP, in which the user is identified by a random identifier. This maximises privacy protection of the user, and ensures that a user need not be tracked between sessions[5]. This satisfies requirements 2, 3, 4, 10, 11 and 12.

3.3. The IdP Account Linking Procedure

The IdP account linking procedure takes place between the user, a LS and one or more IdPs. The user contacts a trusted LS with his web browser, and views a welcome page which invites the user to login. The login page displays a predefined picking list of trusted IdPs (see Figure 1). After choosing an IdP, the user is redirected to his chosen IdP, the IdP authenticates the user by one of its usual mechanisms, and then redirects the user back to the LS, returning an authentication assertion in which the user is identified with a permanent ID (PId) chosen by the IdP. The PId is any attribute type and value with the property that the IdP will always use this PId to refer to this user each time it communicates with the LS about this user, and also the IdP will not generate and use the same PId for identifying a different user. The PId, when concatenated with the URL of the IdP, is thus a globally unique identifier for the user. Note that the conceptual model does not suppose that different IdPs will collaborate and generate the same PId for a given user, since this will weaken the user's privacy, but neither does (or can) the model forbid it. Some organisations e.g. government departments, may prefer to use the same PId for the same user at each IdP. Either way, the PId is a secret between the LS and the IdP and therefore must be encrypted with the public key of the recipient when being transferred between them. The authentication assertion must also be digitally signed by the IdP to guarantee its authenticity.

[5] If any of the user's attributes are uniquely personally identifying attributes, such as an ID, then the user can always be tracked between sessions and uniquely identified. The model therefore supports optional SP tracking.

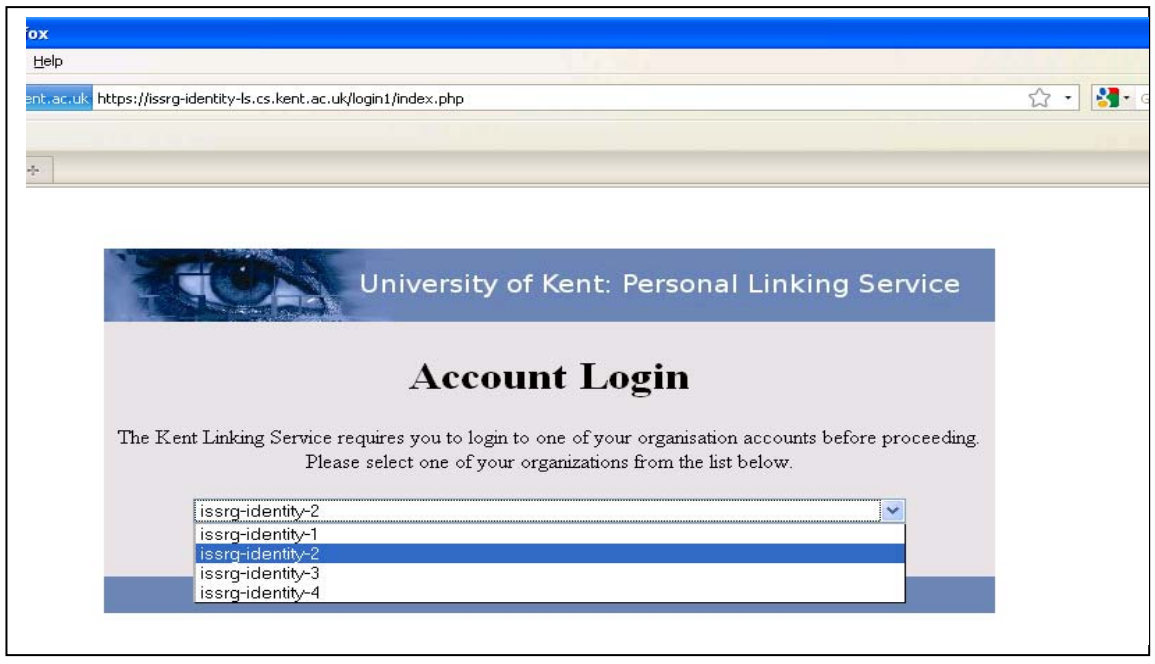

Figure 1. The Linking Service Login Page

The authentication assertion may optionally contain the LoA of the user. This may be stored as the user's registration LoA by the LS in order to improve the performance of subsequent user-SP sessions (see Section 3.6).

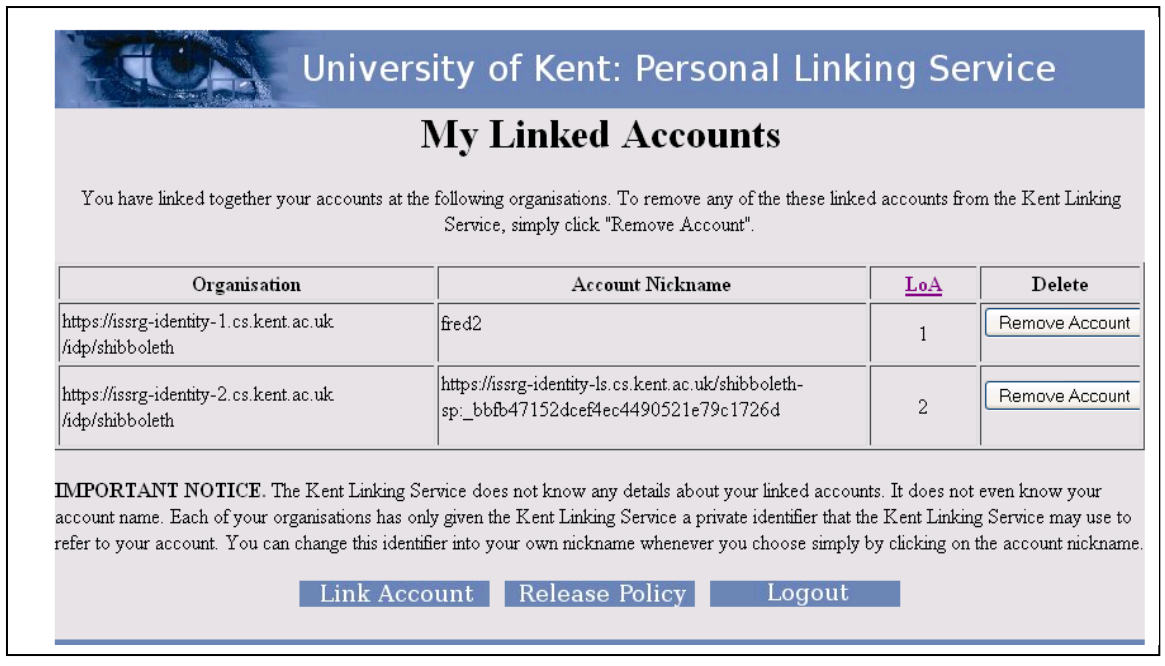

Figure 2. The Linking Service Account Linking Page

When the LS receives the authentication assertion it validates the signature and decrypts the PId. It stores the PId-IdP tuple, and optionally the registration LoA, in its database

entry for this user. In order to make the system more user friendly, the LS should allow the user to change the PId into an easy to remember nickname. Figure 2 shows two linked accounts; one is identified by the PId generated by the IdP, in the other the user has replaced the PId with his own nickname.

The user may then be invited to choose another IdP, in which case the above procedure is repeated and the LS will now have two PId-IdP tuples in its database entry for this user (as in Figure 2). This procedure is repeated as often as the user wishes to link more IdP accounts together. If the user terminates his session with the LS, and then wishes to link further IdP accounts at a later stage, the user must first pick an existing linked IdP from the LS's picking list (see Figure 1), authenticate to it, and this will allow the LS to locate the user's entry in its internal database. The user may then add a further IdP account to its linked set, by selecting one, being redirected to it by the LS, authenticating to it, and the IdP returning its PId to the LS.

It is important to note that the account linking process only links the user's IdP accounts together at the LS, but does not give the LS permission to release any of this information to any SP. In order to do this, the user must first complete an Account Release Policy (ARP), which is permanently stored by the LS. The ARP indicates which IdP accounts can be released to which SPs (see Figure 3). In the simplest case, the user might indicate that all linked accounts can be released to all SPs. The default ARP for each user is that no linked accounts can be released to any SP without first obtaining the user's consent. This is to satisfy requirement 6. In the most complex case, the user might require a different set of linked IdP accounts to be used with each SP. Figure 3 shows the account release policy for a user who has at least 3 linked accounts. All his linked accounts can be released to the SP *dual.openathens.net*, two accounts can be released to the SP *kantarainitiative*, whilst all other SPs are only given access to a single account (the user's fred2 pseudonym account).

An SP may allow a user to dynamically activate account linking at the start of a session (i.e. the dynamic-permanent mode of operation), by allowing the user to choose a linking service instead of an IdP for initial authentication. In this case the user will be redirected to the LS's welcome page and can then go through the procedure described above, before returning to the SP. The user will authenticate to various IdPs, establish the linked accounts, then set up an account release policy, which will include the current SP, before clicking on a button "Return to Service" (not shown in Figures 2 and 3). Because this linking is permanently remembered by the LS, it will then be available for the user to use in all subsequent sessions with the same SP.

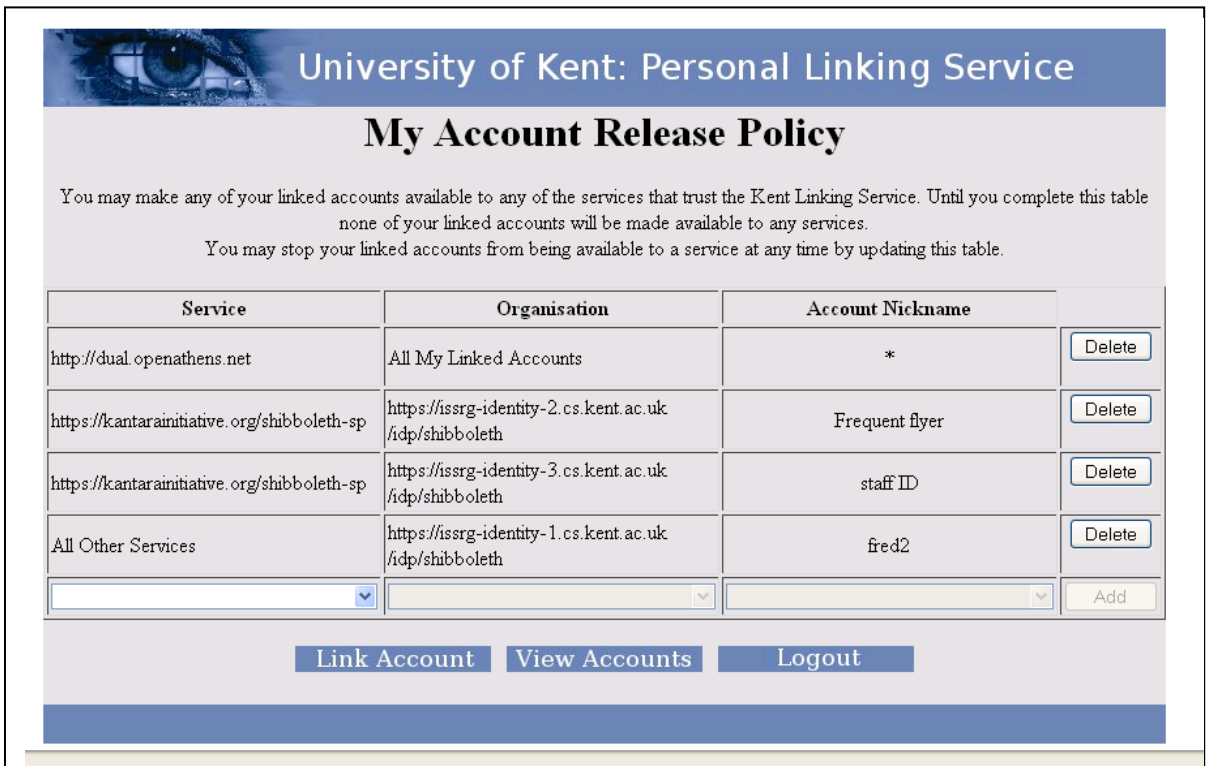

Figure 3. Creating an Account Release Policy at the Linking Service

3.4. Attribute Aggregation Procedure (Static-Permanent Mode)

In this mode, the user has already performed account linking and created his account release policy as described above. The user contacts a SP and is redirected to his chosen authenticating IdP. The SP sends a combined Authentication Request and Attribute Query to the authenticating IdP. The IdP authenticates the user in its normal way, and in addition will ask the user if she wants to use attribute aggregation in this session. The IdP knows to ask this question if it already has established links with one or more LSs for this user. If no links have been established, the authenticating IdP and SP behave as now without attribute aggregation and nothing further needs to be described. If one or more sets of linked accounts do exist, the authenticating IdP will ask the user which of the linking services she wishes to use, and assuming the user chooses one or more of them, attribute aggregation will take place as follows.

The authenticating IdP prepares an authentication assertion containing the session LoA, and uses a newly generated random identifier to refer to the user. The authenticating IdP obtains the user’s consent to release her attributes to the SP, and produces an attribute assertion commensurate with this. In CardSpace the user is allowed to dynamically choose which attributes from her information card are released to the SP. In Shibboleth the user will have pre-established an Attribute Release Policy at her IdP. An attribute release policy rule, as specified by Shibboleth, consists of the following:

- A destination SP ID e.g. https://engineering.example.edu/blackboard/shibboleth-sp
- A list of attribute types (and optionally specific values) that should be released to this SP
- Other optional conditions such as time of day or location of the user etc. as may be implemented by the IdP (typically none are implemented at present, but this allows for more sophisticated privacy controls to be added in the future).

The IdP will only return user attributes to the SP that are present in both the Attribute Query (from the SP) and the Attribute Release Policy rule for this destination SP.

Finally, the authenticating IdP produces one or more Referrals to its linked LSs (one Referral for each LS and set of linked accounts that the user has indicated she wishes to use). The Referral is a new conceptual message that points to another entity (IdP or LS) that may hold additional attributes for the identified user. The user is identity in a Referral by using the PId of the user which is shared between the LS and the IdP. It is encrypted by the sender, to the public key of the recipient, so that only the recipient can decrypt it. All linked IdPs will map the user's PId into the random identifier present in the authentication assertion in order to protect the confidentiality of the PId. All the linked IdPs will return their attribute assertions about the user, to the SP, by using the random identifier allocated by the authenticating IdP in the authentication assertion. This proves to the SP that all the attributes belong to the same authenticated user, without revealing the user's PIds to the SP. All the attribute assertions will be signed by their authoritative sources (this solves requirement 11).

The authentication and attribute assertions and the set of Referrals are returned to the SP by the authenticating IdP. If the attribute assertion is sufficient to grant the user access, then no attribute aggregation is needed and the procedure stops here. If however more attributes are required, the SP forwards the authentication assertion, the appropriate Referral, and an Attribute Query to each LS. It sets a flag in this message to tell the LS whether it should perform the attribute aggregation (LS aggregation) or it will do the aggregation itself (SP aggregation). The Attribute Query informs the LS which set of attributes it still requires for the user. The user is identified in the Attribute Query by the same random identifier as in the authentication assertion. The LS validates all three data constructs, and after decrypting the user's PId from the Referral, looks it up in its internal database and finds the other IdP accounts that are linked to the account from the authenticating IdP. The LS consults the user's Account Release Policy which says which of the linked accounts can be used with this SP, and creates Referrals to each linked account in the release policy. The LS may optionally further filter the Referrals based on the session LoA and the registration LoAs of the linked IdPs (see section 3.6 for more details). The LS uses the public keys of the linked IdPs, obtained from its meta-data, to perform the encryption of the PIds. Each Referral is then digitally signed by the LS.

If the LS supports attribute aggregation, and the SP has asked for LS aggregation, the LS forwards each newly minted Referral, the Authentication Assertion and the Attribute Query to each linked IdP. If it does not support LS aggregation, or the SP has not asked for it, the LS returns the set of Referrals to the SP. The SP combines each Referral with

the Authentication Assertion it received from the authenticating IdP, and an Attribute Query, and forwards these to each linked IdP.

Either way, each linked IdP now receives: a Referral, the Authentication Assertion and an Attribute Query. The contents are the same regardless of the delivery path that was taken. The response will also be the same in both cases. The IdP extracts the PId of the user from the Referral, decrypts the PId, and validates that the Referral was signed by the LS. The IdP uses the PId/LS pair to locate the user's account and corresponding set of user attributes in its database. These attributes are filtered according to the user's consent as provided by her Attribute Release Policy. Using the public key of the recipient SP, the IdP generates an encrypted and signed attribute assertion comprising:

- the user ID copied from the authentication assertion;
- a subset of the attributes that were requested in the Attribute Query. The subset is the intersection of the set that was requested, the set that is held by the IdP, and the set that passes the Attribute Release Policies of both the IdP and the user;
- the attributes are encrypted to the public key of the recipient SP. This ensures that only the recipient SP can read the user's attributes thereby protecting the privacy of the user, and ensuring that the LS or any other intermediary cannot read the contents if the message is relayed.

If attribute aggregation is being performed by the LS, it will receive a set of responses from the linked IdPs, merge these together and forward them to the SP. Otherwise the SP will receive a set of responses directly from the linked IdPs. In both cases, the SP will receive a set of attribute assertions encrypted to itself and digitally signed by their authoritative IdP sources. They will all contain the same subject identifier from the authentication assertion initially received from the authenticating IdP. Based on the session LoA and the set of attribute assertions, the SP is now able to make an authorisation decision about the user's service request.

3.5. Attribute Aggregation Procedure (Dynamic-Permanent Mode)

The user contacts a SP and this time chooses a Linking Service (LS) rather than an IdP for authentication. The advantage to the user of this interaction mode is that she can dynamically determine which linked IdP accounts to use for each service session, whereas in the static-permanent mode she had to rely on a previously established Account Release Policy at the LS. The user is redirected to her chosen LS, and the SP sends it an Authentication Request and Attribute Query. The LS displays its list of trusted IdPs (Figure 1) and the user chooses one of them. The user is then redirected to that IdP to authenticate, and the IdP is sent the combined Authentication and Attribute Query.

The IdP authenticates the user, and this time does not need to ask the user if attribute aggregation is to be used, since the IdP knows that it is, since the user has been redirected to it from a LS. The IdP consults its Attribute Release Policies to determine which attributes may be sent to the SP and/or asks the user for consent, and returns to the LS:

- an Authentication Assertion containing a random session ID and session LoA

- a Referral holding the encrypted PId of the user that is shared with the LS
- an Attribute Assertion containing attributes encrypted with the SP's public key.

The LS temporarily stores the first and third messages whilst processing the Referral. The LS decrypts the PId, looks it up in its database and finds the other PId/IdP pairs that are linked to this one. The LS displays the list of linked accounts to the user (Figure 2), allowing the user to add further accounts if desired. The user is given the option of creating an Account Release Policy to cover the current SP or updating the existing policy (Figure 3). Once the user has finished dynamically linking her accounts and consenting to their release, she tells the LS to return to the SP. The list of linked IdP accounts is filtered by the LS according to the user's Account Release Policy and the session LoA (as described in section 3.6). The LS creates Referrals to each of the linked IdP accounts remaining in the filtered list. If the LS supports LS aggregation and the SP has requested it, the LS will forward each of the Referrals to the linked IdPs, along with the Attribute Query and Authentication Assertion. If SP aggregation is to be employed, the LS will return the set of Referrals to the SP along with attribute and authentication assertions from the authenticating IdP. The SP will forward these to the linked IdPs and processing will continue as in section 3.4 above.

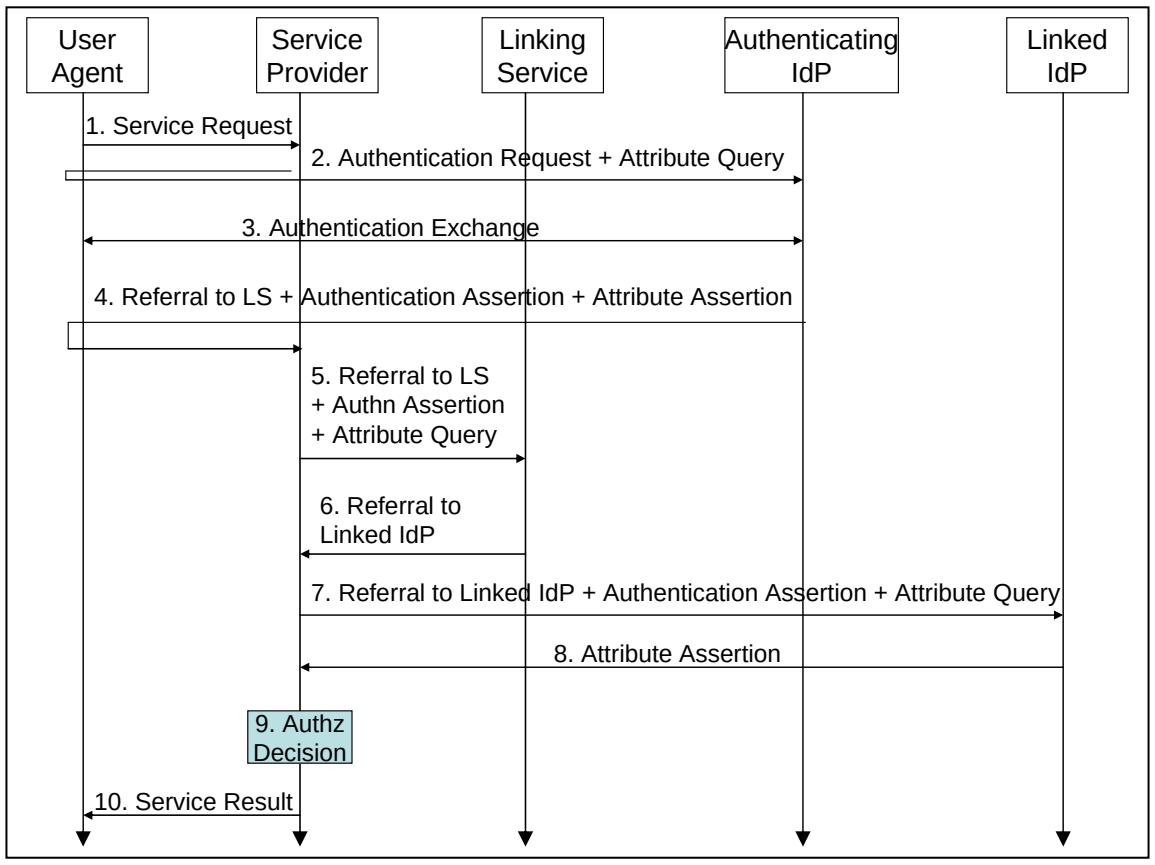

Figure 4. Conceptual protocol flows for static-permanent mode with SP aggregation

3.6 Use of Level of Assurance (LoA)

The LS may use the LoA as follows. During initial account linking, each IdP should tell the LS the session LoA for the user in the returned authentication assertion. The LS may store this value as the account's registration LoA, in order to improve the performance of attribute aggregation in subsequent user-SP service sessions. During a user's service session, the LS will only contact linked IdPs that have account registration LoAs that are higher than or equal to the current session LoA. This prevents a user from creating links with low levels of assurance, and subsequently using them at higher session LoAs. A user is allowed to be assigned attributes at high registration LoAs, and create linked accounts with high registration LoAs, and then subsequently use them on lower session LoAs, since the SP will know that the attributes can only be trusted up to the level of the current session LoA.

If the LS has stored the account registration LoAs, then if a subsequent user-SP session is authenticated by another IdP at a lower session LoA than these, the LS should forward Referrals to these linked IdPs. If the subsequent user-SP session is authenticated by another IdP at a higher session LoA than the account registration LoAs, the LS should not forward Referrals to these linked IdPs. (There is no point, since the linked IdPs must always refuse to return any user attributes in this higher session LoA, since their attributes have not been assured to such a high level.)

If the LS has not stored the user's account registration LoAs, the LS will need to forward Referrals to the linked IdPs in all subsequent user-SP sessions regardless of the value of the session LoA. When an IdP receives an Attribute Query and Authentication Assertion containing a session LoA, the IdP must only return an attribute assertion if the session LoA is less than or equal to the user's account registration LoA (and it trusts the authenticating IdP to authenticate to the stated session LoA). Otherwise the IdP must return an error message such as User assurance not high enough, or Authenticating IdP not trusted.

4. Trust Model

Trust forms a core building block of all federations and VOs. It is important that each party knows what trust it is expected to have in all the other parties that it is interacting with. It is also important to know what the implications are when a communicating party decides that it does not trust the other communicating party. The trust requirements of the entities involved in attribute aggregation and the implications of a lack of trust between them are presented below.

4.1 Trust Requirements during the Linking Process

The LS and IdPs must have trust relationships during the linking process as specified below.

1) All IdPs must trust the LS to:
   - hold their PIds securely and confidentially,
   - hold the links between the PId-IdP tuples securely,
   - not release their PIds to any entity (except if legally required to do so),

- only release the linked PId-IdP tuples to a SP with the user's consent and only then if the PIds remain confidential.

These trust relationships give the IdPs confidence that the user's identity (PIds) will remain private and not be divulged to any third party by the LS.

2) The LS must trust each linked IdP to:
   - authenticate the user correctly to the stated session LoA,
   - assign a unique PId to the user and not re-use the PId for a different user,
   - always use the same PId in subsequent interactions about the user.

These trust relationships give the LS confidence that each IdP will behave responsibly when authenticating the user.

4.2 Trust Requirements during Service Invocation

The LS, linked IdPs and SP must have trust relationships with each other during the service invocation as specified below.

3) The SP must trust the LS to:
   - hold the user's linked accounts securely and with integrity,
   - in the case of SP aggregation, only release the linked accounts to it (as Referrals) with the user's consent,
   - in the case of LS aggregation, only aggregate attribute assertions from the linked accounts with the user's consent,
   - correctly relay authentication requests in the dynamic-permanent mode.

The above will give the SP confidence that the LS will not mix up user accounts and wrongly link different user accounts together.

4) The SP must trust the authenticating IdP to:
   - authenticate the user correctly to the stated session LoA,
   - only return the user's attributes to it (as Attribute Assertions) with the user's consent, and
   - only generate a Referral to the user's LS with the user's consent.

The only difference in the trust relationship between an IdP and a SP with or without attribute aggregation is the last item concerning Referrals.

5) The SP must trust each IdP to:
   - correctly process incoming Referrals, and only return attributes that are valid for the authenticated user at the stated session LoA, for which they are authoritative and for which the user has provided consent.

This is an additional item of trust that the SP must have in an IdP since the processing of Referrals is a new action that previously was not required when there was no attribute aggregation.

6) The LS must trust the authenticating IdP to:
   - authenticate the user correctly to the stated session LoA,
   - only send a Referral to it with the user's consent.

Note that the LS does not need to trust the SP nor any of the linked IdPs during service invocation, since it does not provide any of them with confidential information and it will not suffer any loss if any of them behave incorrectly.

7) All linked IdPs must trust the authenticating IdP to:
   - authenticate the user correctly to the stated session LoA, and
   - only create a Referral to a LS with the user's consent.

8) All linked IdPs must trust the LS to:
   - in the case of SP aggregation, to only create Referrals that point to them and give these to the SP with the user's consent,
   - in the case of LS aggregation, only contact them on behalf of the SP (with an Attribute Query, Referral and Authentication Assertion) with the user's consent.

9) All linked IdPs must trust the SP to:
   - keep the user's attributes private and
   - only use the attributes according to its and the user's wishes.
   These trust requirements are no different to those that operate when there is no attribute aggregation.

4.3 Implications of Lack of Trust

In general, if there is a lack of trust between the various entities, then attribute aggregation wont work. If there is sufficient level of trust between the SPs and IdPs as in current systems, then the system will revert to its current state of operation without attribute aggregation. If even this level of trust is not present, then the user will not be able to obtain a federated service.

We use the same paragraph numbering system as in 4.1 and 4.2 above to show what happens if the required level of trust is not there.

1) and 8) If an IdP does not trust the LS as described above, then it will not return a PId to it at Linking time, and no link will be stored by the LS. No Referrals will ever be generated by the LS to the IdP during service invocation.

2) If the LS does not trust an IdP as described above then no link will be stored to this IdP and attribute aggregation cannot work with this IdP.

3) If the SP does not trust the LS as described above, it will simply discard any Referrals to the LS that it receives from the authenticating IdP, and it will grant or deny the user access to its resources based solely on the attributes that it has obtained from the authenticating IdP. The SP will not redirect the user to an untrusted LS for authentication at service invocation time, neither will it accept an authentication response that has been returned via an untrusted LS.

4) If the SP does not trust the authenticating IdP at all then the SP will not redirect the user to this IdP for authentication, nor will it accept authentication statements from this IdP. The user will need to choose a different IdP in order to obtain any service from the SP. If the SP trusts the IdP for authenticating users but does not trust it to generate Referrals, then the SP will discard any Referrals that it receives from the IdP and the user will be authorised based solely on the attributes received from the authenticating IdP.

5) If the SP does not trust an IdP to correctly process incoming Referrals it can either simply discard the Referral and not forward it to the IdP (SP aggregation) or discard any Attribute Assertions it receives from this IdP via the LS (LS aggregation).

6) If the LS does not trust the authenticating IdP then no link will be stored to that IdP by the LS. The authenticating IdP will not be able to generate a Referral to the LS and so attribute aggregation via this LS will not be able to take place.

7) If an IdP does not trust the authenticating IdP then it will not return any attributes to the SP (SP aggregation) or LS (LS aggregation) in response to the Referral that it receives. (Remember that the Referral is accompanied by the Authentication Statement from the authenticating IdP.)

9) If the authenticating IdP does not trust the SP, then it will not authenticate the user on behalf of the SP. If any linked IdP does not trust the SP, then it will not return any attributes in response to an Attribute Query.
Note that if the authenticating IdP does not trust one of the other linked IdPs, this does not matter, since the authenticating IdP does not have to rely on the other linked IdP for anything, and cannot be damaged by it.

5. Mapping to Standard Protocols

The message exchanges of a conceptual model can be mapped onto any number of protocols, standard or proprietary. We have chosen to map the conceptual message exchanges onto standard protocols based on SAML V2.0. A summary of each is presented below.

5.1 IdP Linking Protocol

The IdP linking protocol is used to link user accounts at different IdPs into a single user account at the linking service. We use the standard SAML v2.0 authentication request [8] for this. The LS acts as a service provider and issues the SAML authentication request to query an IdP for authentication details. The LS sets the <NameIDPolicy> Format attribute to “persistent”, the AllowCreate attribute to “true” and the SPNameQualifier attribute to the name of the LS. These tell the IdP to create a persistent ID for the user if one does not already exist for her with the LS. The name identifier returned in the AuthnStatement is then used by the LS as the PId to either map the user to a pre-existing user account or to create a new user account if the IdP/PId tuple does not exist. Since the combination of IdP name and PId is globally unique, the LS will know if this

combination already exists in its database or not. After the response from the first IdP has been processed, if the user wishes to link to another IdP, then a new authentication request is issued to the second IdP. The name identifier returned in the second AuthnStatement is added to the user's LS account that contains the IdP/PId tuple returned in the first authentication response. The user can repeat this process as often as required, thereby linking an arbitrary number of IdP accounts together in one LS account.

5.2 Managing the Account Release Policy

Since this protocol is local to the LS and the user client software, we do not standardise it. In our implementation we use the http protocol, JavaScript and PHP to create the active web pages.

5.3 Encoding the Referral

The Referral is conceptually a data structure created by one linked partner (the IdP or the LS), that points to the user's account at the other linked partner. It is sent in the response to both a Discovery Query to the LS and an Attribute Query to an IdP, and means "further attributes of this subject may be found here".

We can use a Liberty Alliance ID-WSF Endpoint Reference (EPR) [12] to encode the fields of a Referral as follows:

- we use a SAMLv2 assertion as the <sec:Token> of the EPR (as described below) to identify the user's account at the receiving partner's service,
- a service endpoint address which identifies the partner's service where the security token can be used, and
- a service type definition that says which protocol type to use when querying the partner's service (this is always set to the LA Discovery Service).

The EPR can then be returned in either a response to a Discovery Query to a LS or as an attribute in response to a SAMLv2 Attribute Query to an IdP. It is used to contact the Discovery Service of either an IdP or a LS respectively, to ask for service instances of either the Attribute Authority or linked IdPs respectively.

The fields of the SAMLv2 assertion used as the <sec:Token> are set as follows:

- the Issuer is set to the partner that issues the Referral,
- the Subject is set to <encryptedID> and contains the PId of the user encrypted to the public key of the receiving partner,
- Audience Restriction is set to the name of the receiving partner and
- Advice contains an <AssertionIDRef> that points to the authentication assertion which authenticated the user.

The whole assertion is then signed by the issuing partner. The Advice element ensures that a secure mapping is made between the authentication assertion's random identifier and the user's PId. Encrypting the PId allows the issuing partner to send the <sec:Token> to the other partner via any intermediate third parties without the third party knowing the user's permanent identifier, which is a secret that is only shared between the two partners.

5.3 Service Request Protocols

The user initially contacts a service provider asking for a particular service, using http. The user is redirected to either an IdP (static-permanent mode) or LS (dynamic-permanent mode) by a “Where Are You From” service [7] or some proprietary means. The redirection message contains a standard SAMLv2 Authentication Request <samlp:AuthnRequest>. The AttributeConsumingServiceIndex attribute of the request is used to specify that both user attributes and Referrals (EPR attributes) should be returned in the response from the IdP (static-permanent mode) or LS (dynamic-permanent mode) with SP aggregation. If however the SP wishes the LS to perform LS aggregation in dynamic-permanent mode, then only user attributes are requested. When the user is redirected to an LS for authentication in dynamic-permanent mode, the LS proxies the SP’s authentication request according to section 3.4.1.5 of [8]. The LS generates a new <samlp:AuthnRequest> message which matches that received from the SP and sends this to an IdP (or other LS) of the user’s choice. Once an IdP is reached the user authenticates to the IdP, and the IdP returns an authentication assertion, the requested attributes and Referral EPRs to the LSs it is linked to. The LS uses its Referral EPR to find the user’s account in its database, enabling the LS to either create an additional set of Referral EPRs (SP aggregation) or contact the IdPs directly in order to retrieve the user’s attributes (LS aggregation). The LS creates a new authentication response to the SP, based on the original one, in accordance with the proxying rules in [8], and returns the authentication assertion, set of attribute assertions and Referral EPRs to the SP.

The SP will receive in response a standard SAML2 response <samlp:Response> message containing: a signed authentication assertion with a random identifier as the Subject element; one (SP aggregation) or more (LS aggregation) attribute assertions containing the user’s attributes, each signed by their authoritative sources; and zero, one or more attribute assertions containing Liberty Alliance ID-WSF EPRs that represent Referrals to one or more LSs or IdPs. If the returned attributes are sufficient to grant the user access to the service, then no further attribute aggregation is needed and the procedure stops here. If however this initial set of attributes is insufficient, and one or more Referrals are present in the response, the SP may initiate (further) attribute aggregation operations as below.

The SP uses each Referral to construct a Discovery Query to the service endpoint indicated in the Referral. A Discovery Query is designed to enable the requester to obtain a list of ID-WSF EPRs that provide a particular type of service. These EPRs are returned in the Discovery Query Response message. In our case we define two types of service query: the first, sent from an SP to a LS, asks for the EPRs of the linked IdPs’ discovery services; the second, sent from a LS or SP to an IdP’s discovery service, asks for the EPR of its SAML v2.0 attribute authority so that the latter can subsequently be queried for the user’s attributes. In order to make it easy for SP implementers, we allow both types of service to be specified in a single Discovery Query message, so that the SP code can create the same message to be sent to both a LS and an IdP. In this way LSs and IdPs can be interchanged and linked together without the SP needing to be concerned about this. The message recipient on the other hand does know what type of service it provides, and therefore produces the correct response, ignoring the other type of service query.

We place two security tokens into the wsse:Security header of the Discovery query message, in order to identify the user. The first is the <sec:Token> copied from the Referral, which identifies the user by their encrypted PId, and the second is the single sign on authentication assertion provided by the authenticating IdP which indicates that the user has been recently authenticated and identifies the user by a random identifier. The two are linked together by the Advice element in the <sec:Token>.

If the recipient of the Discovery Query is the LS, it will extract the encrypted PId, decrypt it, then look up the PId in its database to find the set of linked IdP accounts. If the LS does not support LS aggregation it will return a Discovery Query Response containing Referral EPRs that point to the discovery services of each linked IdP. The SP can then send Discovery Query messages to each of the linked IdPs. If the LS supports LS aggregation, it will send out Discovery Query messages to each of the linked IdPs. Either way each linked IdP will receive a similar message. The primary difference in these messages is that the encrypted PId in the <sec:Token> has been replaced by one that refers to the PId held by the respective IdP. Each linked IdP will decrypt the token and lookup the PId in its database. It will check that the AssertionIDRef in the Advice field matches the one in the single sign on authentication statement, and if so, will map the random ID from the authentication statement into the PId of the user's account. It will extract the LoA from the authentication statement, and if this is less than or equal to the Registration LoA for the attributes in the user's account, it will return the EPR of the Attribute Authority (AA) instance that can return attributes of the user, in the Discovery Query Response message.

When the LS or SP receive the Discovery Query Response, they can extract the EPR of the AA instance and send it a SAML Attribute Query message, using the random ID from the authentication statement to identify the user. The AA can now respond with a signed attribute statement asserting that this user has the various attributes that are in his account. It is important that the attributes are encrypted for the SP, in order to preserve user privacy. This protects the attributes not only during transfer, but also stops the LS from viewing them, in the case of LS aggregation. In order to achieve this we needed to extend the Attribute Query message by adding the equivalent of the AssertionConsumerServiceURL that SAMLv2 defines for Authentication Requests. This tells the recipient who the ultimate consumer of the assertion will be. No such field is currently standardised for Attribute Queries, so we needed to create it.

6. Conclusions and Next Steps

We have produced a conceptual model for attribute aggregation in federations, in which the user is in total control of linking his various identity accounts and attributes together, whilst retaining a high degree of privacy protection. Each of the linked IdPs only know one of the user's identity accounts, the one he has authenticated to, and do not know any of the user's other identity accounts, even after he has linked them together. This is because the linking is done via a trusted third party called a linking service. Even the linking service does not know the attributes or real identities of the user; it only knows that a user has various accounts at various identity providers, but it does not know any of

the account IDs. It only knows pseudo account IDs that have been generated especially for it. Finally, the service provider gets full assurance that the user is the genuine holder of the various asserted attributes, since the authentication assertion and all the attribute assertions contain the same (random) user identifier, and they are all signed by their authoritative sources.

Underpinning the conceptual model is a fully worked out trust model that indicates the amount of trust that the various parties must have in each other, but none of these are more onerous that those required by the trust models of today's existing federations.

We have mapped the conceptual model onto standard protocols based on SAMLv2, and implemented a demonstration system which may be experimented with at http://issrg-beta.cs.kent.ac.uk:8080/loademo.html. We will shortly release the system as open source software under a BSD type license, as part of the PERMIS software suite (www.openpermis.org).

Our next step is to add attribute aggregation to the CardSpace protocols and Information Card model, so that users will be able to use the Identity Selector to select multiple cards in a single session.

Acknowledgements

The authors would like to acknowledge the UK JISC and the EC's FP7 programme for supporting this work under the Shintau project and the TAS³ - Trusted Architecture for Securely Shared Services - project (grant agreement n° 216287) respectively.

References

[1] Catlett, C**. "**The Philosophy of TeraGrid: Building an Open, Extensible, Distributed TeraScale Facility." 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid, 21-24 May 2002
[2] see http://www.incommonfederation.org/
[3] See http://msdn2.microsoft.com/en-us/netframework/aa663320.aspx
[4] Tom Barton, Jim Basney, Tim Freeman, Tom Scavo, Frank Siebenlist, Von Welch, Rachana Ananthakrishnan, Bill Baker, Kate Keahey. "Identity Federation and Attribute-based Authorization through the Globus Toolkit, Shibboleth, GridShib, and MyProxy". NIST PKI Workshop, April 2006
[5] Jill Gemmill, John-Paul Robinson, Tom Scavo, Purushotham Bangalore. "Cross domain authorization for federated virtual organizations using the myVocs collaboration environment". Concurrency and Computation: Practice and Experience, Vol 21, No 4, 2009. pp 509-532.
[6] George Inman, David Chadwick, Nate Klingenstein. "Authorisation using Attributes from Multiple Authorities – A Study of Requirements". Presented at HCSIT Summit - ePortfolio International Conference, 16-19 October 2007, Maastricht, The Netherlands. Available from http://sec.cs.kent.ac.uk/shintau/pages/publications.html
[7] Morgan, R. L., Cantor, S., Carmody, S., Hoehn, W., and Klingenstein, K. (2004), "Federated Security: The Shibboleth Approach", *Educause Quarterly*, Vol. 27, No. 4,

[8] OASIS. “Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V2.0”, OASIS Standard, 15 March 2005
[9] Liberty Alliance. “Liberty ID-WSF Authentication, Single Sign-On, and Identity Mapping Services Specification”, Version 2.0.
[10] Liberty Alliance. “Liberty ID-WSF Security Mechanisms Core” Version 2.0
[11] W. E. Burr, et al. “Electronic Authentication Guideline”, NIST Special Publication 800-63, Sept. 2004
[12] C. Cahill and J. Hodges, eds., “Liberty ID-WSF Discovery Service Specification,” v2.0, Liberty Alliance Project. Available from www.projectliberty.org/liberty/content/download/3449/22973/file/liberty-idwsf-disco-svc-2.0-errata-v1.0.pdf.
[13] Liberty Alliance. “Liberty ID-WSF Security Mechanisms Core” Version 2.0
[14] See http://www.ukfederation.org.uk/
[15] David W. Chadwick, Alexander Otenko. "The PERMIS X.509 role based privilege management infrastructure". Future Generation Computer Systems, Volume 19, Issue 2, February 2003, Pages 277-289
[16] Ralf Groeper, Christian Grimm, Siegfried Makedanz, Hans Pfeiffenberger, Wolfgang Ziegler, Peter Gietz, Michael Schiffers. "A concept for attribute-based authorization on D-Grid resources". Future Generation Computer Systems, Vol 25. Issue 3. March 2009. pp 275-280.