Federated Identity Management

David W Chadwick

Computing Laboratory, University of Kent, Canterbury, CT2 7NF, UK
d.w.chadwick@kent.ac.uk

Abstract. This paper addresses the topic of federated identity management. It discusses in detail the following topics: what is digital identity, what is identity management, what is federated identity management, Kim Cameron's 7 Laws of Identity, how can we protect the user's privacy in a federated environment, levels of assurance, some past and present federated identity management systems, and some current research in FIM.

Keywords. Identity Management, Shibboleth, CardSpace, Federations

1 Introduction

What is digital identity? One can find many different variants of this definition on the Internet. Perhaps the most general definition is the one from a new draft ITU-T standard (X.1250) on global identity management [2], which states that identity is the "*Representation of an entity (or group of entities) in the form of one or more information elements which allow the entity(s) to be uniquely recognised within a context to the extent that is necessary (for the relevant applications)*." This definition is so general that it lacks precision of whose identity we are talking about (who or what is an entity?) and what data are we talking about (what is an information element?). Whilst an entity can be any object, in most cases it is personal identity that we are concerned about, so we will restrict this chapter to considering identity management of people rather than of any object. In this context, the information elements are restricted to Personal Indentifying (or Personally Identifiable) Information (PII), which is "*the information pertaining to any living person which makes it possible to identify such individual (including the information capable of identifying a person when combined with other information even if the information does not clearly identify the person)*." [1] We can consider that PII is simply the attributes[1] of a person, such as: their hair colour, sound of their voice, height, name, qualifications, past actions, reputation, medical records, etc. You might think that hair colour is not PII and is not a digital identity as it is too generic, but if we had a rule that stated that ginger haired people are granted a 10% discount at Ginger's hairdressing salon, then hair colour alone would be sufficient identity information to allow a person to be *uniquely recognised within a context to the extent that is*

[1] An attribute is defined in [3] as "*information of a particular type*".

necessary (for the relevant applications). So even something as generic as hair colour can be classed as a digital identity and as PII. To summarise, we can say that a person's (digital) identity comprises a set of attributes, and only a subset of these attributes are necessary to allow the person to be sufficiently recognised within a given context.

So what is identity management? In short it is the whole process of managing a user's identity attributes. Y.2720 [1] has a more comprehensive definition which states that identity management is: *A set of functions and capabilities (e.g. administration, management and maintenance, discovery, communication exchanges, correlation and binding, policy enforcement, authentication and assertions) used for:*

- *Assurance of identity information (e.g., identifiers, credentials, attributes);*
- *assurance of the identity of an entity (e.g., users/subscribers, groups, user devices, organizations, network and service providers, network elements and objects, and virtual objects); and*
- *enabling business and security applications.*

Before proceeding further, we should clarify the difference between an *identifier* and an *Identity*, and an *attribute* and a *credential*. An identifier is usually a series of digits and/or characters that is used to uniquely identify an entity within one domain or system. No two entities (or users) within the same system can have the same identifier. So an identifier is a rather special type of identity attribute, since no two users can share the same identifier, whilst they may have other identity attributes in common, such as hair colour. Furthermore, an identifier is tightly bound to the system or domain in which it is defined; it usually cannot be meaningfully moved between domains, unlike the other identity attributes. Indeed, different domains can use the same identifier to identify different users. An identifier is only one of the identity attributes that comprise that person's digital identity within a system. Different computer systems know different subset's of a person's identity attributes, but each computer system will have its own identifier which uniquely identifies this individual within this system. An individual whose identity is distributed throughout many systems will therefore have multiple identifiers such as: their passport number, login ID, social security number, email address etc., which are each unique within their own domains. Some systems may store the identifiers from remote domains as well as their own. For privacy (and other) reasons, users are typically wary about releasing their identifiers to third parties, since these can uniquely identify them, whereas their other identity attributes, such as age, typically cannot.

An *attribute assertion* is a *claim* made by someone (the asserter) that a particular person possesses a particular attribute. Usually attributes have to be conferred on individuals (or asserted) by authoritative sources. Whilst people may be trusted in some situations to assert some of their identity attributes themselves, for example, their favourite drink, they certainly wont be trusted in all situations to assert all of their identity attributes themselves, for example, their qualifications or criminal record. Thus different authoritative sources are usually responsible for assigning different attributes to individuals. For example, the university that one graduated from is the authoritative source of one's degree attribute. These authoritative sources are also known as attribute authorities (AAs). An identity provider is an attribute authority combined with an authentication service to authenticate its users. An

identity provider can authenticate a user and then issue an attribute assertion about the user. Attribute assertions typically have to be digitally signed to ensure their integrity and authenticity. A digitally signed attribute assertion is an *authorization credential*.

Whilst discussing credentials, we need to differentiate between authentic credentials and valid credentials.

- Authentic credentials are ones that have not been tampered with and are received exactly as issued by the issuing authority. Their digital signature is used to prove their authenticity.
- Valid credentials are ones that are trusted for use by the recipient, sometimes called the relying party.
  - Example 1: Monopoly money is authentic if obtained from the Monopoly game pack. It was issued by the makers of the game of Monopoly. Monopoly money is valid for buying houses on Mayfair in the game of Monopoly, but it is not valid for buying groceries in supermarkets such as Tesco's or LIDL.
  - Example 2: My credit card is an authentic credential. I can use it to buy groceries in Tesco, so it is valid there, but I cannot use it in LIDL as they do not accept credit cards. It is not valid there, but it is still authentic.

The difference between an authentic and a valid credential is whether the relying party does or does not trust the issuer of the credential to issue that particular credential.

Authoritative sources may remove attributes as well as assign them. For example, a university may remove a degree from a student, if it was subsequently proved that the student had committed plagiarism in their dissertation. Similarly, in the UK, the General Medical Council (GMC) is the only authoritative source of who is a doctor, and it keeps a register of them. If a doctor commits malpractice, the doctor may be *struck off* the register by the GMC. Thus in identity management systems, we cannot rely on the individual to assert his various attributes, otherwise he might lie about his various roles, and omit to tell about negative attributes such as the points on his driving license. Similarly we cannot rely on a single identity provider to assert all a user's attributes, but only the attributes they are authoritative for. For example, a credit card company would not normally be trusted to assert someone's degree qualification attribute. Consequently a set of authoritative sources may need to be consulted by service providers before the latter grant users access to their resources. X.1250 defines an authoritative identity provider as "*the Identity Provider responsible by law, industry practice, or system implementation*" for asserting a particular identity attribute.

This brings us to the topic of federations and federated identity management. A federation is defined in X.1250 as "*an association compromising any number of service providers and identity providers*"[2]. Implicit in this definition is trust. The fact that the various providers have formed an association between themselves means that they must have a certain level of trust between themselves, sufficient to be willing to exchange messages between themselves. When these messages contain the authentication and authorisation credentials of users, allowing users from one system to access resources in a federated system, we have federated identity management (FIM). With FIM, a user can use her credentials (authentication and authorisation)

from one or more identity providers to gain access to other sites (service providers) within the federation. FIM brings the following benefits to the various stakeholders:

- it gives users the single sign on (SSO) capability, allowing them to move between the various service providers without having to authenticate or log in again,
- it allows service providers to offload the cost of managing user attributes, passwords and login credentials to trusted identity providers
- it provides scalability, allowing service providers to offer services to a much greater number of users
- it allows identity providers to maintain close relationships with end users and sell them additional services, as well as extract fees from the service providers they support.

In a centralised system, as opposed to a federated system, the user typically presents their identifier and an authentication token (such as a password) to prove that they are entitled to be known by this identifier. The system then associates the user with this identifier and with all the attributes linked to this identifier. The user is then granted access based on these. In a distributed system the user might typically have different identifiers in each local system, so if the user authenticated to one identity provider using his local identifier, this identifier would not be known by and therefore could not be used by the other local systems to grant the user access. Single sign on would not be possible without some sort of federation. When X.509 based PKI systems were first designed, they tried to solve this problem by allocating each user a globally unique identifier (called an X.500 distinguished name) which would be known by all local systems in the distributed system and therefore could be used to grant the user access. Since this global identifier was bound to the user's public key in an X.509 public key certificate, a signature created by the user's private key could be used as an authentication token by each local system.

One of the reasons this X.509 based identity management system failed was the privacy concerns about everyone knowing everyone else's globally unique identifier. The breakthrough came when it was realised that a user's identifier did not need to be globally unique, but could remain local to the system that allocated it. Authorisation to use a remote federated system could be granted based on the user's identity attributes, rather than on the user's identifier. If the identity attributes are provided by trusted authoritative sources, then a service provider can be assured of the identity of the user, even if the user's identifier is unknown (or temporary). This gave birth to early federated identity management systems such as Passport and Athens, and more lately to standardized systems such as Shibboleth [4] and CardSpace [5], which will be described later.

2 The 7 Laws of Identity

After the failure of Microsoft's Passport FIM system (see later), Kim Cameron thought long and hard about what is needed in order to build a successful FIM system. He discussed the issues intently on his blog www.identityblog.com. One of the end results was his 7 Laws of Identity [6] described below. Another was the design and

implementation of CardSpace which is now an integral part of Vista and Internet Explorer 7. The seven laws are summarised below.

1. User Control and Consent

Technical identity systems must only reveal information identifying a user with the user's consent. [6]

The underlying hypothesis here is that users will cease to trust a system that reveals their identity attributes to others, without the users' explicit consent. A user needs to have confidence that any system that is provided with his identity attributes will protect them and respect his wishes for how they should be used.

2. Minimal Disclosure for a Constrained Use

The solution which discloses the least amount of identifying information and best limits its use is the most stable long term solution. [6]

The underlying hypothesis here is that all systems are vulnerable to attack and the theft (or loss) of the confidential information that they hold. Therefore systems should minimize the PII that they capture and store, and should delete it as soon as the purpose of capture is complete.

3. Justifiable Parties

Digital identity systems must be designed so the disclosure of identifying information is limited to parties having a necessary and justifiable place in a given identity relationship. [6]

The underlying hypothesis here is that users resent their PII being given to third parties that have no proper role to play in a digital transaction. Thus if a user is posting pictures to his family blog, then he should not need to use a government identity provider, or Microsoft for that matter. This is hypothesized as the primary reason that Microsoft Passport failed to become the identity provider for the Internet.

4. Directed Identity

A universal identity system must support both "omni-directional" identifiers for use by public entities and "unidirectional" identifiers for use by private entities, thus facilitating discovery while preventing unnecessary release of correlation random identifiers. [6]

The underlying hypothesis here is that users do not want everyone to know their identifiers, they prefer to keep them private, whilst public web sites and commercial organisation do want everyone to know their identifiers and hence be able to contact them. When users establish communications with a public entity such as a service provider, they should be assigned a one-off (or private) identifier that is only for use in this communication (or with this service provider). The use of different identifiers with different service providers will prevent the service providers from colluding together to build global profiles of the user.

5. Pluralism of Operators and Technologies

A universal identity system must channel and enable the inter-working of multiple identity technologies run by multiple identity providers. [6]

The underlying hypothesis here is that diversity and competition between identity providers is good, and users should be able to switch between pseudonymous identities at will. This necessitates that there is an overarching meta-identity system that uses a common protocol for the transport of identity credentials, whilst supporting an infinite variety in the types of credential technologies that are supported.

6. Human Integration

The universal identity metasystem must define the human user to be a component of the distributed system integrated through unambiguous human-machine communication mechanisms offering protection against identity attacks. [6]

The underlying hypothesis here is that the vast majority of identity thefts succeed by attacking the link between the PC and the human rather than the links between the PC and the various identity and service providers. The human is the weakest link in the chain and therefore securing this communication link should be an essential component of any identity management system. A second hypothesis is that a well known *ceremony* is needed for this communication, one that will always be used by the user, that the user will become very familiar with, and can therefore immediately determine if and when an attack is taking place. The design of the Identity Selector in CardSpace, described later, is one attempt at making the human-computer link more secure through having a consistent ceremony.

7. Consistent Experience Across Contexts

The unifying identity metasystem must guarantee its users a simple, consistent experience while enabling separation of contexts through multiple operators and technologies. [6]

This law is very closely related to the last law, and is also re-iterating several of the previous laws. It is re-stating that the user needs to have a consistent experience (i.e. ceremony) regardless of the underlying credential technologies that are in use, or the pseudonym that the user chooses to use for any particular transaction. The user should be able to switch between pseudonyms or identities at will. In order for the user to recognize which identity he is using in any given transaction, identities should be "thingyfied" into icons that the user can easily recognize. This naturally leads to the concept of information cards, an electronic representation of the plastic cards we all carry around in our pockets today.

These seven laws of identity are not physical laws that all living things must abide by, like the law of gravity, but they are laws which identity management systems should endeavour to support, and which they break at their peril. The peril in this case is that users are likely to reject any identity management system that does not abide by the seven laws in its implementation. CardSpace has endeavoured to keep them, as we shall see later.

3 Related Issues

3.1 Privacy Protection

As the seven laws of identity management make clear, privacy protection is an important issue that FIM systems should take into account. In many countries there are legal requirement for information systems to protect user privacy. Invariably these all derive from the OECD privacy guidelines [8], which state eight data protection principles. These are:

1. Collection Limitation Principle

There should be limits to the collection of personal data and any such data should be obtained by lawful and fair means and, where appropriate, with the knowledge or consent of the data subject.

2. Data Quality Principle

Personal data should be relevant to the purposes for which they are to be used, and, to the extent necessary for those purposes, should be accurate, complete and kept up-to-date.

3. Purpose Specification Principle

The purposes for which personal data are collected should be specified not later than at the time of data collection and the subsequent use limited to the fulfilment of those purposes or such others as are not incompatible with those purposes and as are specified on each occasion of change of purpose.

4. Use Limitation Principle

Personal data should not be disclosed, made available or otherwise used for purposes other than those specified at the time of collection except with the consent of the data subject or by the authority of law.

5. Security Safeguards Principle

Personal data should be protected by reasonable security safeguards against such risks as loss or unauthorised access, destruction, use, modification or disclosure of data.

6. Openness Principle

There should be a general policy of openness about developments, practices and policies with respect to personal data. Means should be readily available of establishing the existence and nature of personal data, and the main purposes of their use, as well as the identity and usual residence of the data controller.

7. Individual Participation Principle

An individual should have the right:

a) to obtain from a data controller, confirmation of whether or not the data controller has data relating to him;

b) to have communicated to him, data relating to him within a reasonable time, at a charge that is not excessive, in a reasonable manner, and in a form that is readily intelligible to him;

c) to be given reasons if a request made under subparagraphs(a) and (b) is denied, and to be able to challenge such denial; and

d) to challenge data relating to him and, if the challenge is successful to have the data erased, rectified, completed or amended.

8. Accountability Principle

A data controller should be accountable for complying with measures which give effect to the principles stated above. [8]

But how are the above to be implemented in FIM systems and more importantly, can we build FIM systems that automatically safeguard at least some of the above? One method is to separate identity providers (IdPs) from service providers (SPs), and to store identity attributes with the IdPs only and not with the SPs. All modern FIM systems are designed to be capable of this. Then we give the user control over her identity attributes that are held at her various IdPs, and allow her to say which of these may be given which of which SPs. In Shibboleth this is implemented as Attribute

Release Policies that can be set by the user (see later) at each IdP. Then we define protocols that do not release the user's identifier from the IdP to the SPs, so that multiple SPs cannot collude together about a specific user. Further if the identifier is randomly generated each time, the SP cannot correlate the multiple sessions of the same user. The OASIS Security Assertion Markup Language (SAML) protocol [7] supports both of these schemes by allowing the IdP to create a new random identifier for the user for each association (this is used by Shibboleth). Alternatively if the user needs to be uniquely identified in order to obtain personalized services, then the IdP can generate a new permanent identifier to identify this user to this specific SP, and use this identifier every time. SAML also supports this, and this is used by Liberty Alliance in its identity management protocols [9]. In this way we prevent the SPs from knowing the real identity of the user of her unique identifier at the IdP. The user is either pseudonymously identified (as in Liberty) or randomly identified (as in Shibboleth). Finally if the IdPs make the attribute assertions which they give to the SP short lived, then we effectively remove the attributes from the possession of the SP at the close of each transaction (or shortly afterwards). Short lived assertions are another feature of the SAML protocol.

3.2 Level of Assurance

Different IdPs will authenticate users in different ways and to different strengths. For example, usernames and passwords are weaker than public-key certificates and private keys. A relying party's *level of assurance* (LOA) that the user is really who it thinks she is depends not only on the electronic authentication method used by the IdP but also on the initial registration process used by the IdP. For example, registering electronically over the Web is much weaker than turning up in person with a passport. Registering over the Web is equivalent to self asserting your identity attributes. So a relying party should rightly give little credence to these identity attributes.

The National Institute of Standards and Technology (NIST) recommends four LOA levels, with level 4 being the strongest and level 1 the weakest. Some SPs may wish to grant a user different access permissions based on the LOA during the current session. For example, if the user authenticates with an LOA of 1, she can read the resource, but with an LOA of 3 she can modify its contents. Limitations of the NIST recommendation are that: the LOA only applies to user authentication, and not to her identity attributes for authorisation, and it is a compound metric that is dependent on both the strength of the registration process and the electronic authentication method being used.

In the latest research being carried out at the University of Kent, we believe it's more useful if the LOA is split into two separate metrics, one for registration of the identity attributes, and one for the authentication method being used in the current session.

Prior to any computer-based authentication, a user must register with a service and provide various credentials to prove her identity. For example, before a new student can register to use the University of Kent's computing services, she must first present her passport and existing qualifications to prove she is entitled to register as a student. We call this the *registration LOA*. Different systems will require different registration

documents and have different registration procedures, and will therefore have different registration assurance levels. Any identity attributes that are assigned to the user during registration or afterwards by the IdP, and for which the IdP is the authoritative source, will be given the registration LOA. For example, after successful registration, the University of Kent allocates the student a login ID (her identifier) and associates various attributes with this in its database, for example, degree course, e-mail address, department, tutor, and so on, for which the university is the authoritative source. These will be assigned the registration LOA. Other identity attributes for which the IdP is not the authoritative source, such as date of birth and name, may be given the same registration LOA value or a lower one depending upon the quality of the documents used at registration time. For example, if the person's name and date of birth were taken from their passport, they would have the same registration LOA as the other identity attributes. If they were asserted by the user without any documentary evidence to support these assertions, they would be given the lowest LOA value. Typically an IdP would not send these attributes to an SP, because it is not authoritative for them.

After registration the IdP will issue the user with authentication credentials, and these will have an associated *authentication LOA.* For example, the University of Kent may offer different authentication mechanisms for student login, such as un/pw with Kerberos, un/pw with Secure Sockets Layer (SSL), one-time passwords via a mobile phone, and so on. The system assigns each of these mechanisms an *authentication LOA* with the proviso that no authentication LOA can be higher than the registration LOA that originally authenticated the user's identity attributes. The reason for this is that the user's identity attributes can never be asserted at a higher assurance level than was carried out during the registration process. Consequently, registering over the web using self assertions must always have the lowest LOA (in NIST this is 1) assigned to both the registration LOA and the authentication LOA. However, if an IdP never asserts any identity attributes to the SP, and merely authenticates the user and presents a uniquely generated identifier to the SP, then the authentication LOA can be set to the strength of the authentication method that is used. The SP can be sure that each time this user contacts it, that it is the same user because the identifier will be the same each time. The SP also has an assurance of this to the value of the authentication LOA. In this scenario, the SP has no idea who the user is, as no identity attributes have been provided, but it does know that it is the same user each time. This is how the early versions of OpenID worked [11], before attribute assertions were added to it. (Note that there is no level of assurance provided by OpenID).

When a user logs in for a session, the authentication service assigns her a *session LOA* equivalent to the authentication LOA of the authentication mechanism she chose to use. Thus the same user may have different session LOAs with the same SP, due to the fact that she used different authentication methods with her IdP in the adjacent sessions.

A new addition to the SAML protocol [10] is adding support for passing the session LOA in each assertion sent from the IdP to the SP. Thus the SP can obtain the level of assurance for the identity attributes that are to be used for authorizing the user in the current session. Note that the SAML specification [10] only allows one LOA

value to be passed with an assertion, so all the identity attributes in the assertion must have been registered to that level if they are to be passed.

3 Some early FIM systems

3.1 Microsoft's .NET Passport

.NET Passport is an authentication and single sign on system that allows users to access multiple service providers using the same credentials. Each service provider remains in charge of its own authorisation, and may use Passport provided identity attributes to help in this. It works as follows. Users register at a service provider site, but their credentials and profile information are stored centrally by Microsoft at the Passport server. This means that sites must trust Microsoft/Passport to hold the user credentials securely, and to authenticate the users correctly during sign on.

Referring to figure 1 below, the system works as follows:

1. The user browses Site A, a Passport participating site, and clicks the "Sign In" button.

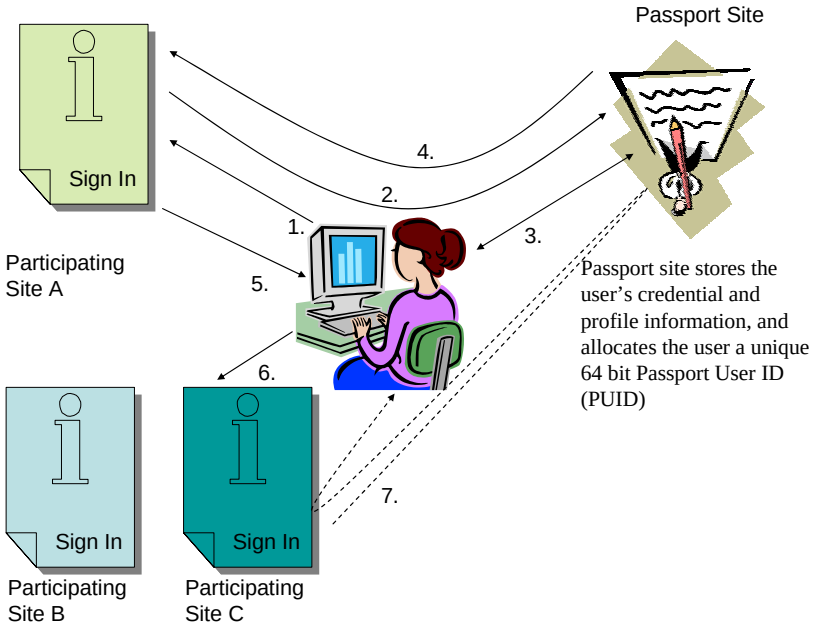

Figure 1. Message Flows in Passport.

2. The user is redirected to a co-branded registration page at the Passport site, which displays the registration fields chosen by Site A. (The minimum number of fields required is two: email address and password.)
3. The user reads and accepts the terms of use (or declines, and the process ends), and submits the registration form containing their profile information. The registration form is sent encrypted to the Passport site via SSL to protect the user's privacy. Next the user can choose whether or not they want to opt in to share their profile information with other Passport-enabled sites or not. They can select any or all of three tick boxes, viz: share email address, share first and last name, share all other profile attributes, which may comprise: Birth Date, Country / Region, First Name, Gender, Last Name, Occupation, Postal Code, Preferred Language, State, and Time Zone. After completing registration, the user is shown a congratulations page and
4. is then redirected back to Site A with their encrypted authentication ticket and profile information attached. The redirect message contains four cookies (a ticket granting cookie containing a secret key, a Participating Sites cookie, an authentication cookie and a profile cookie) which are stored in the user's browser to shortcut future authentication attempts within the lifetime of the cookies. The last three cookies are encrypted with the secret key in the ticket granting cookie, the latter is encrypted with a secret key known only to Passport.
5. Site A decrypts the authentication ticket and profile information and continues the user's registration process, or immediately grants her access to the site.
6. When a user moves to another Participating Site, say Site C, the user is again shown the "Sign In" button as in Step 1, which she may decide to click.
7. If so, she is redirected to the Passport site (as in step 2). The user's browser sends the four cookies back to Passport during redirection. Passport then knows the user has already successfully authenticated and redirects the user back to site C (as in step 4). The redirect contains an authentication ticket (generated from the authentication cookie) and profile information (generated from the profile cookie) which Site C can decrypt and use to authorize the user. The redirection also contains an updated Participating Sites cookie containing the list of all Participating Sites the user has visited during this session.
8. When the user logs out of Passport, all four cookies are deleted from the browser and the Participating Sites cookie is used to clean up all Participating sites computers.

Passport has been very successful when used between Microsoft owned sites such as MSN and Hotmail. But because all participating sites have to trust Microsoft to hold the identity of the user, and to authenticate the user properly, it fails Kim Camerson's 3rd Law of Justifiable Parties. Why should Microsoft be involved in a transaction between a car hire company and a hotel? It is clearly not appropriate for Microsoft to be involved in all federated communications between different commercial companies. Another failing of Passport is that it does not provide good privacy protection for a user's attributes, since these are made available to Microsoft, and the user does not have fine grained control over how they are released to the affiliated sites (see step 3 above).

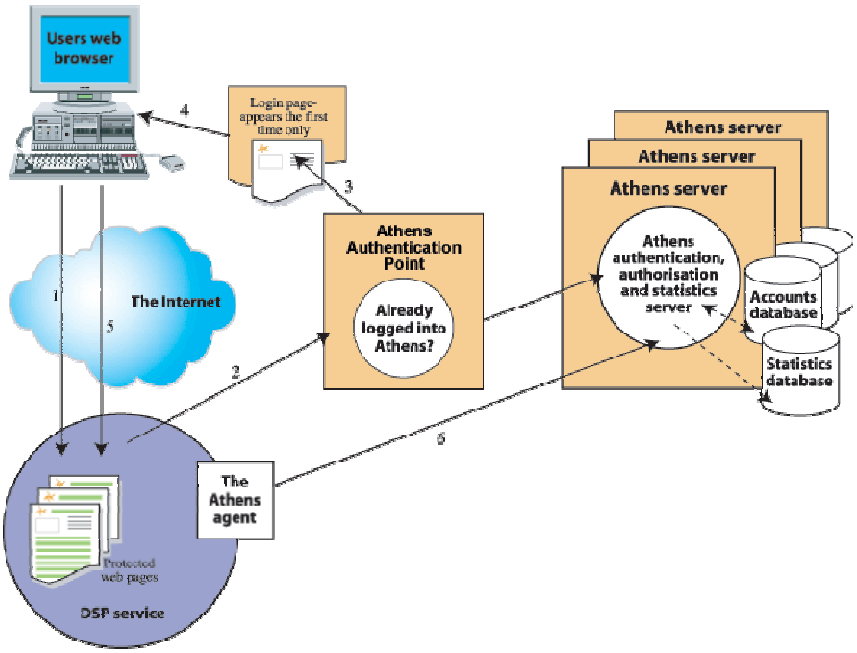

Figure 2. Message Flows in Athens. © EduServ, 2002

3.2 UK Athens

Athens was the de facto standard for secure access management to online services for the UK Education and Health sectors during the late nineties and first decade of two thousand. By 2002, 769 user sites, with over 2 million users, were using Athens to connect to 249 resources at 51 service provider sites such as Elsevier, Wiley, Science Direct, and Oxford University Press. It is therefore one of the most successful federated identity management systems to date.

Athens was originally designed by a team at the University of Bath in a series of JISC-funded projects, and was subsequently made into a commercial service that is owned, developed and operated by EduServ (http://www.eduserv.org.uk). Essentially EduServ is a "trusted third party" identity provider. EduServ operate a large database of over 2 million user IDs and passwords along with authorisation data (which says which sites users can access). The service is replicated to provide a resilient service. Each participating college or university administers its own part of the database to keep their user base up to date. This makes the administration manageable.

Referring to Figure 2 below, Athens works as follows:

1. The user contacts a data service provider (DSP) e.g. Elsevier
2. The user is re-directed to the Athens Authentication Point (AAP) via an SSL connection

3/4. The AAP displays the login page to the user. The user types in his Athens username and password which is sent encrypted back to the AAP over SSL. If the

user is authenticated correctly, the AAP writes an encrypted cookie (with a validity time of 8 hours) back to the browser to enable Single Sign On (see 7 below) and redirects the user back to the DSP.

5. The redirection message carries an encrypted token to signal the user's successful authentication. This token is symmetrically encrypted with a secret shared between the DSP and the AAP, and it contains the user's Athens username and a short expiry time (60 secs). The user is now authenticated and tries to access various data sources at the DSP, but may not be authorised to access everything.

6. The DSP web server calls a special Athens Agent plug in software which communicates with the Athens database to see if the user is authorised to access this particular data source. The user is granted or denied access depending upon the reply. The Athens Agent plug-in is provided either as a toolkit (C, Java, Perl implementations all available) for integration into the supplier's system or as pre-packaged modules for Apache and MS IIS.

7. If the user moves to a different DSP, then the user is re-directed to the AAP as in step 2 above. This time however the AAP is sent the cookie by the browser, so the AAP knows the user has been authenticated and does not need to ask him to login again. The user is redirected straight back to the DSP site.

On 1 August 2008, Athens was superseded by Shibboleth as the JISC preferred federated identity management system for UK education. It is now being slowly phased out. Since it appears to have been so successful, why is this? Firstly it uses proprietary protocols and is not easily adaptable to becoming a general federated identity management system. Sites cannot leverage it to set up their own mini-federations. In essence it is a centralised system and suffers from the same trust problems as Passport. Why should all transactions have to be authenticated by the central Athens server? Finally, during the last few years, there have been significant efforts in standardising federated identity management protocols, and the US, Europe and Australia are migrating towards these systems. Consequently the time has come to do the same with Athens.

4 Some current FIM systems

4.1 Shibboleth

Shibboleth [4], conceived and developed by the Internet 2 consortium (http://www.internet2.org/), is a system designed to ease the formation of federations and collaborations between organisations. Shibboleth provides a protocol to allow collaborating organisations to more easily authenticate and authorise each other's users. In Shibboleth, authentication is always performed by the user's own organisation – the identity provider – using the organisation's existing authentication scheme. The organisation might use usernames and passwords, or Kerberos, or one time passwords etc. It does not matter, providing the scheme is secure enough for the federating service providers. In this way, the user always uses his existing login credentials and so single sign on is enabled through this. Whilst the current system

does not support Levels of Assurance, it clearly will not be difficult to add it, since Shibboleth uses standard SAML protocols.

Authorisation always takes place at the site the user is trying to access – the service provider – using identity attributes of the user provided by the identity provider. A trust relationship exists between the identity and service providers, so that the service provider trusts the identity provider to correctly authenticate the user, and to provide the correct set of attributes for the user. Messages are digitally signed by the identity provider so that the service provider can validate that they are correct and trustworthy. Single sign on is enabled throughout the federation so that if the user contacts a second service provider, the identity provider can immediately send a digitally signed message to the new service provider saying that the user has already been authenticated correctly. Therefore the user does not need to login again.

Importantly, Shibboleth provides strong privacy protection of the user's details. The user's username (identifier) is privacy protected because the user is identified with a different pseudonym, or random identifier, each time he contacts a service provider. Thus the service provider is not able to profile the user's accesses since it has no way of linking together the different identifiers from the different sessions, or of linking an identifier to the actual user. (Note that in the case of abuse of a service provider by a user, the service provider can request the help of the identity provider to identify the abusive user, by scanning its activity logs. However, such breaches of privacy are expected to be exceptional, and only triggered by abuse in the first place.) The user's attributes are privacy protected because both the user and the home site can set Attribute Release Policies (ARPs) that say which attributes can be released to which remote sites. This helps to protect the privacy of the user's attributes, since only the minimum necessary need to be sent to each service provider. It also provides the user with full control and the ability to give consent for the user of his attributes. If a service provider wishes to provide personalized services to the user, then one of the user's identity attributes will need to uniquely identify the user, and be provided in each session. The user will need to agree to this through his ARP.

4.1.1 How Shibboleth works

Shibboleth is a Web based middleware layer that defines the protocols that are sent between a user's Web browser, his home/identity provider's Web server and the target resource site's Web server. Shibboleth is standard's based, and the current version makes use of the SAML v2 protocol [7] for encoding its messages.

When a user contacts a Shibboleth service provider from their browser, requesting access to a particular URL, Shibboleth single sign on and access control takes place in separate stages as shown in Figures 4 and 5 (although it is possible to combine both stages into a single communication exchange, which is not shown). In a large distributed open environment the authentication stage has a number of complications. Firstly how does the resource site/service provider know where the user's home site/identity provider is in order to redirect the user to the correct authentication service? This is known as the discovery problem. Secondly, how can the SP trust that the authentication statement that is returned is authentic and that the random identifier identifies the current user? A number of different solutions have been attempted to solve the discovery problem. The one used by OpenID [11] is to use globally unique IDs based on DNS names so that the DNS can be used to discover the location of the

user's IdP. Another is to pre-configure one or a small number of IdPs into the SP and force the user to use one of these. However, the method currently favoured by Shibboleth is to use a Where Are You From Service which asks the user to pick the site that he or she is from (see Figure 3), from a pre-configured list of all IDPs that the SP trusts. The answer to the second question is provided by the Shibboleth trust model. This requires the sites to establish trust relationships between themselves as part of the process of forming a federation. They do this by exchanging their public key certificates or CA root keys, so that they can validate messages signed by each other.

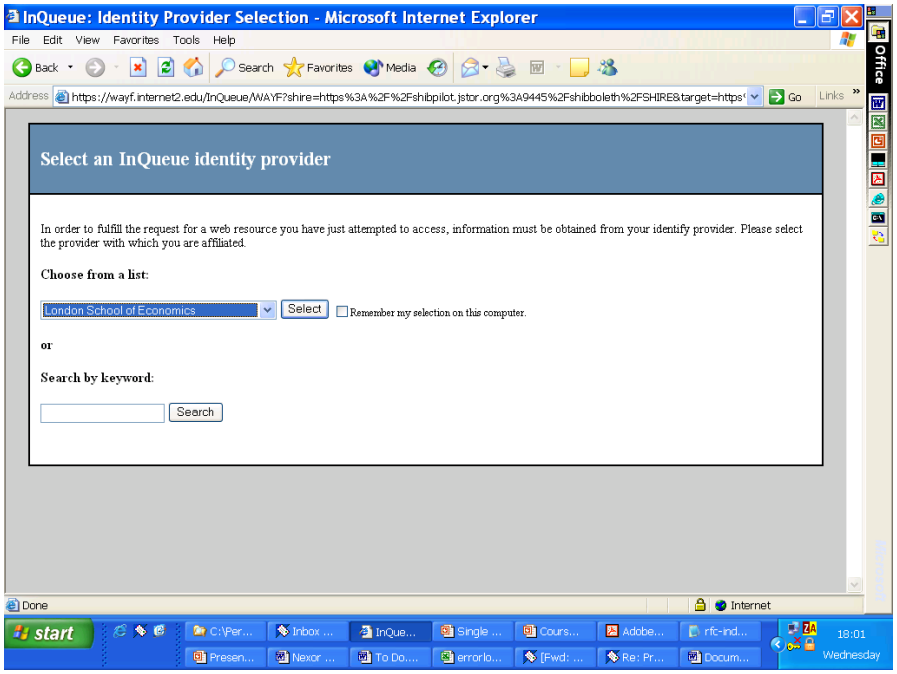

Figure 3. Screen shot of a typical Shibboleth Where Are You From service

Referring to Figure 4 below:

1. The user makes a request to a web site. The user can be stationed at his home site, or anywhere else on the Internet. The web site knows nothing about the user, so needs to authenticate and obtain the attributes of the user in order to grant him/her access. The SP first needs to discover where the user is from. This is the function of the Where Are You From (WAYF) service.
2. The Shibboleth SP uses the Http Redirect reply to re-direct the user to its Where Are You From service. This prompts the user to choose his home site from a picking list (see Figure 3). The WAYF knows the name and location of the Authentication Service of each trusted IdP that is participating in Shibboleth. If the SP does not trust a particular IdP then it won't be listed in its WAYF picking

list. If a user is deceitful and tries to fool the system by picking an IdP to which he does not belong, he will have difficulty authenticating to that site's authentication service, since he won't have any valid credentials for it. However, if he picks his own home site, he should find authentication is no problem. Consequently the honest user picks his home site, and then

3. the user is re-directed to the Authn Service at his home site by the WAYF service.
4. The Authn Service (AS) is responsible for making sure the user is authenticated locally at the IdP, and for creating a random identifier that can be used to retrieve attributes about the user. The Authn Service prompts the user to login and provide his authentication token. The IdP can use whatever type of authentication it likes e.g. username/password, Kerberos, digital signatures etc. This is currently not relayed to the SP, but the LOA will be added as a future extension. Once the user has authenticated him/her self, the AS produces a random identifier for the user. The content of the identifier is left entirely up to the home site. A random identifier ensures that the user's local identifier remains private to the IdP, and the SP will never know the true identity of the user that is accessing it. Thus Shibboleth automatically provides Privacy Protection of the user's identity.

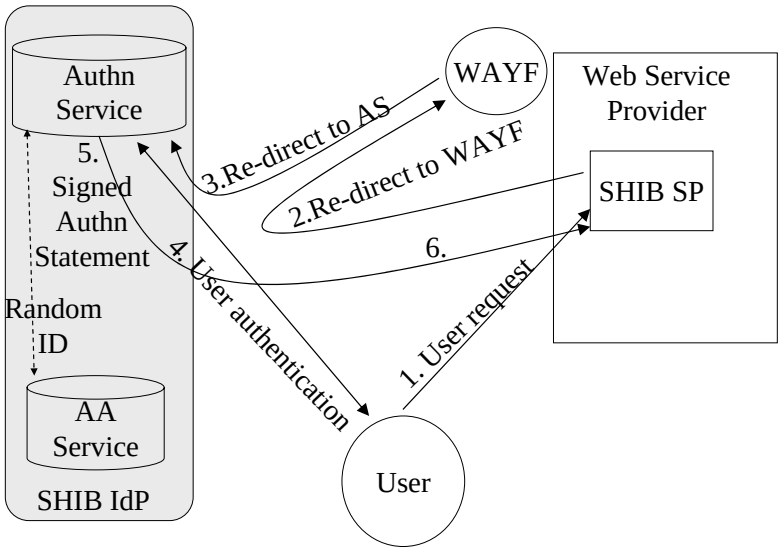

Figure 4. Authenticating the user in Shibboleth

5. The AS passes the random identifier (in the form of a SAML Authentication statement) back to the user's browser inside an HTML form
6. that POSTS the data back to the destination SHIB SP. This information includes the location of the AA service at which the random identifier will be usable. This

message is digitally signed by the AS to prove its authenticity. The SHIB SP must check the signature and the message contents to ensure its validity. The AS ensures that the AA Service knows the new random identifier for the user.

7. Referring to Figure 5 below, the SHIB SP sends a SAML Attribute Query Message to the Attribute Authority (AA) service at the user's IdP. This request needs to be protected i.e. mutually authenticated and have message integrity. SSL with client side authentication is used for this.
8. The AA server returns a SAML Attribute Response Message (ARM) containing a SAML Attribute Statement. This message also needs to be protected by mutual authentication, message integrity and message confidentiality, so SSL is used.
9. Once the ARM is received and validated by the SHIB SP, the embedded attributes are passed to the web service's authorisation code. Shibboleth provides its own simple authorisation code for sites to use, or they can use more sophisticated policy bases systems such as XACML [12] or PERMIS [13].
10. The authorisation code now grants or denies access to the user based on their attributes.

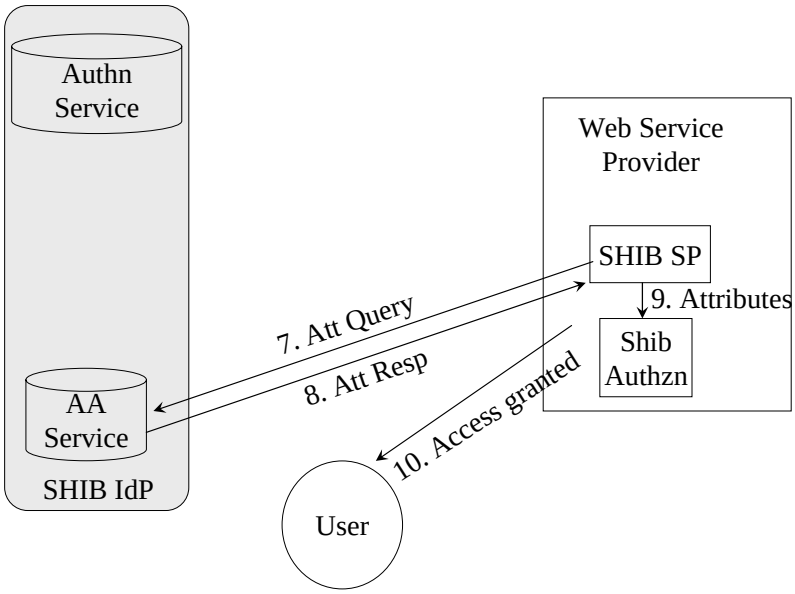

Figure 5. Authorising the user in Shibboleth

The latest version of the Shibboleth specification has introduced a performance improvement over the original version, by optionally allowing stage one and stage two to be combined together into one message exchange. In this way, the initial request from the SP may ask for both authentication and attribute statements, and the digitally signed SAMLresponse message may optionally contain the user's attributes as well as the authentication assertion.

4.1.2. Privacy Protection and Attribute Release Policies in Shibboleth

Shibboleth has four mechanisms to ensure user privacy. Firstly it allows a different pseudonym for the user's identity (the randomly generated identifier) to be returned each time. Secondly it allows the service provider to request specific user attributes to be returned, as opposed to "all", so as to minimise the potential loss of privacy. Thirdly, and most importantly, it requires that the attribute authorities provide some form of control over the release of user attributes to service providers, which they term attribute release policies (ARPs). Both users and administrators should have a say about which attributes can be released. Finally, in order to stop third parties from seeing the attributes as they are transferred over the Internet, the connection between the AA service and the SP should be protected by using SSL/TLS with strong encryption enabled.

A basic ARP rule at the AA consists of the following:

- A destination service provider 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 identity provider (typically none are implemented at present, but this allows for more sophisticated privacy controls to be added in the future).

The identity provider can have as many of these rules as it needs for each destination SP. The destination SP ID allows the AA to find the right ARP rules to use when it receives an Attribute Query Message from an SP, since the latter contains the service provider ID as part of the SAML protocol message.

ARPs are specified in XML according to a predefined schema (shibboleth-arp-1.0.xsd). An example ARP is given in Figure 6 below. This ARP consists of 3 rules, one that will release an attribute to any service provider, one that will release an attribute to any service provider within a specific DNS domain, and one that releases a specific attribute value to a specific SP.

The <Target> element of an ARP rule identifies the service provider. It has one child element which is either:

- <Requester> which contains a matching rule (only two are currently defined, regular expression or exact) and an SP ID to match against, or
- <AnyTarget/> which indicates that all SPs match.

The <Target> element is followed by an <Attribute> element which specifies the attributes that can or cannot be released to this target. As shown in the examples, for any particular attribute type, either *any* attribute value can be specified, or a specific value. Each attribute can either be Permitted or Denied from being released. Note that in most cases it is not necessary to Deny an attribute from being released, since it will only be released if it occurs in a Permitted element. However, an administrator (or a user) may want to allow all attribute values to be released except one particular value. In this case, the attribute with *any* values would be Permitted to be released in one ARP rule, and then the specific value would be Denied to be released in another ARP rule.

Each IdP administrator creates the site's ARP and stores it in a file called *arp.site.xml*. This ARP applies to all users for which the AA is answerable.

```
<?xml version="1.0" encoding="UTF-8"?>
<AttributeReleasePolicy
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance";
xmlns="urn:mace:shibboleth:arp:1.0"
xsi:schemaLocation="urn:mace:shibboleth:arp:1.0 shibboleth-arp-1.0.xsd">
<Rule>
<!-- This rule will release the edu person affiliation attribute with a value
member@example.edu to any service provider -->
        <Target>
                <AnyTarget/>
        </Target>
        <Attribute name="urn:mace:dir:attribute-def:eduPersonAffiliation">
                <Value release="permit">member@example.edu</Value>
        </Attribute>
</Rule>
<Rule>
<!-- This rule will release the user's edu person principal name attribute to any
example.edu/Shibboleth service provider -->
        <Target>
<Requester matchFunction="urn:mace:shibboleth:arp:matchFunction:regexMatch">
https://.*\.example\.edu/Shibboleth</Requester>
        </Target>
        <Attribute name="urn:mace:dir:attribute-def:eduPersonPrincipalName">
                <AnyValue release="permit"/>
        </Attribute>
</Rule>
<Rule>
<!-- This rule will release a specific contract value of the edu person entitlement
attribute to the www.external.com/contract.asp service provider -->
        <Target>
<Requester matchFunction="urn:mace:shibboleth:arp:matchFunction:exactShar"
https://www.external.com/contract.asp</Requester>
        </Target>
        <Attribute name="urn:mace:dir:attribute-def:eduPersonEntitlement">
                <Value release="permit">urn:example:contract:113455</Value>
        </Attribute>
</Rule>
</AttributeReleasePolicy>
```

Figure 6. An example Attribute Release Policy

Each user may also have his/her own ARP and this is stored in a file called *arp.user.<$PRINCIPALNAME>.xml*. in the same directory as the site ARP file. User ARPs can be maintained either by the IdP administrator or by the users themselves, according to the site's local policy. The MAMS project in Australia has produced a user friendly GUI editor for ARPs, called the Shibboleth Attribute Release Policy

Editor (ShARPE). This is designed for use by both end users and IdP administrators, and it is distributed as open source software [14].

When an attribute query request is received, the Shibboleth software computes an effective ARP by locating the user's and site's ARPs and extracting from these each rule that matches the SP ID. All the attributes and values that are requested in the query are then compared with the effective ARP, and only those that are permitted to be released are included the response. If the same attribute is simultaneously Permitted in one rule (say in the site's ARP) and Denied in another rule (say in the user's ARP), then it will be denied from being released.

We can see that attribute release policies provide a flexible and powerful tool for preserving the privacy of a user's attributes, and they are one of the strengths of the Shibboleth infrastructure.

4.2 CardSpace

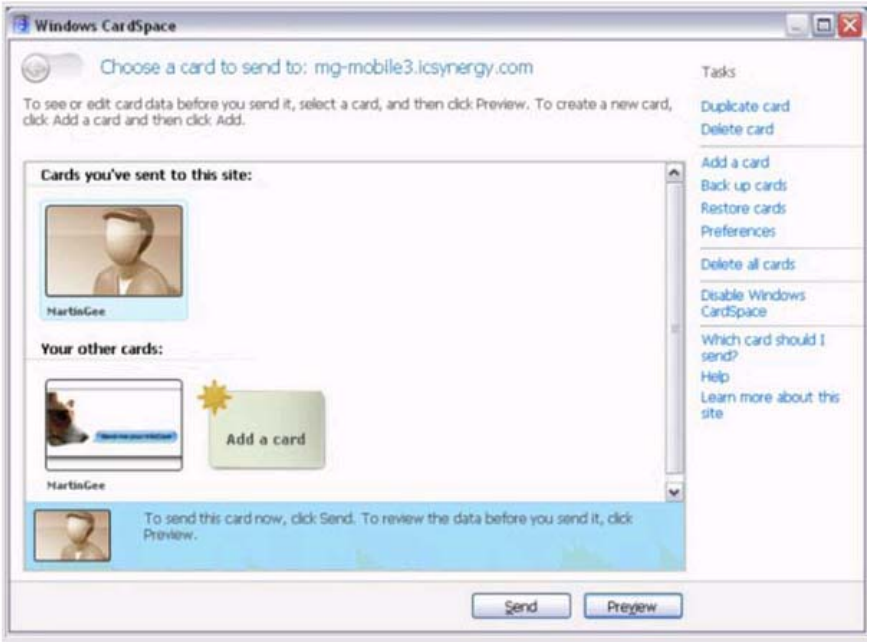

Figure 7. The CardSpace Identity Selector

Information Cards are the core component of Microsoft's CardSpace identity management and authorisation system. A good high level overview of CardSpace can be found in [15]. Information Cards are a representation of a person's online digital identity. Information Cards have some excellent features in terms of both usability and security. From a usability perspective, the metaphor that Information Cards use for electronic credentials is the plastic card that everyone is familiar with. These are displayed on the user's desktop so that the user can select the card he wants to use in

any transaction (see Figure 7). Cards that are acceptable to the service provider (SP), and hence selectable, appear in full colour, whilst cards that are incompatible with the SP's requirements are greyed out and hence not selectable. Cards can be self generated or managed by an IdP. Self generated cards contain information (attributes) asserted by the user himself, whereas managed cards contain attributes that are asserted by the IdP. Microsoft calls attribute assertions "claims". The fact that the attribute assertions (or claims) of the managed cards do not actually reside on the user's desktop, but are pulled from the IdP on demand, is largely hidden from the user. The only telling feature is that the user has to enter his login credentials with the IdP in order for the claim to be picked up and sent to the SP. This could be seen as a usability disadvantage or inconvenience to users, since the user is distracted from his/her primary task, which is accessing a service provider, into providing authentication credentials to an alternative party, the identity provider. But this is really not that much different to users entering their PINs today in order to activate their plastic cards.

From a security and privacy perspective, CardSpace contains some excellent features. It has been designed specifically to conform to Kim Cameron's 7 Laws. Firstly it is resistant to phishing attacks, unlike Shibboleth for example where an SP could redirect users (via its fraudulent WAYF service) to a malicious entity masquerading as the user's identity provider. This is subverting the discovery procedure in order to point to a malicious IdP. In CardSpace the discovery information is stored securely on the user's PCs in the meta-information of their information cards. Phishing can only succeed if the attacker can subvert the user's PC without the user's knowledge, in order to plant subversive information cards in the user's identity selector. Besides the fact that it would be extremely hard to do, the user would most likely recognize this if it did happen. Secondly there is nothing of high value in the user's identity selector that can be stolen by an adversary, since the managed cards don't contain the actual credentials or claims; these are only generated on demand by the IdPs when asked to do so. The credentials/claims are short lived, cryptographically protected, designed to be transferred as quickly as possibly from the IdP to the SP via the user's desktop, and as an additional protection can be encrypted to be read by the SP only. So there is little opportunity for an attacker to steal them whilst in transit. However, some researchers from Germany have already shown that the credentials can be captured and stolen from the user's desktop and then effectively used to masquerade as the user [16]. Fortunately they also show how the vulnerability can be protected against with only a small change to the CardSpace protocols. This requires the SSL channel ID of the connection between the user and the SP, to be included in the credentials sent from the IdP. In this way the credentials cannot be used by any system other than the user's PC. Self asserted claims/credentials can also be stolen in the same way, and also protected in the same way.

4.2.1 CardSpace in more detail

Figure 8 shows the data flows in more detail.

1. The user contacts a web site using his web browser, and is invited to sign in. The user chooses to use information cards for this and selects Send Information Card.
2. The web site displays the login page to the browser and embeds in this page, as an object, the site's CardSpace Security Policy (see 4.2.2).

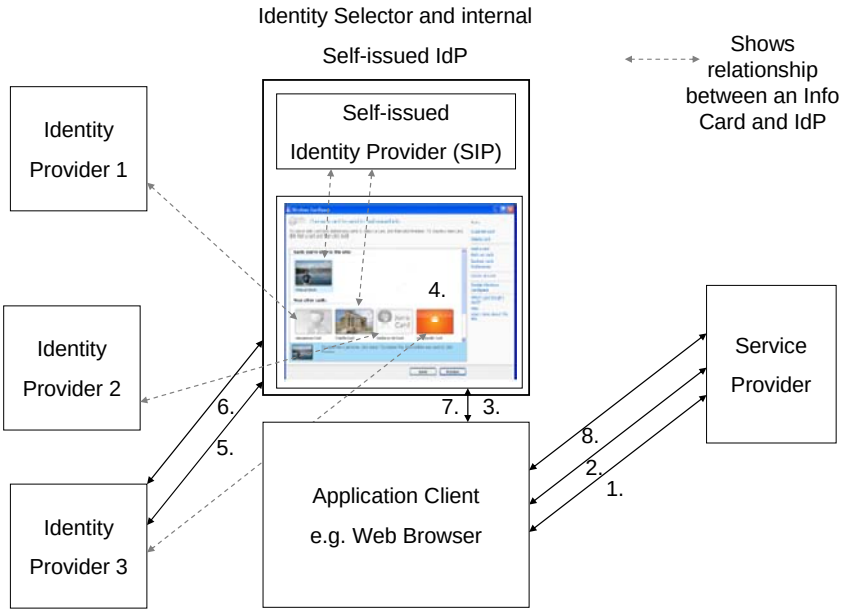

Figure 8. CardSpace message flows

3. The browser passes the Security Policy to the Identity Selector application which evaluates it and brightly displays the Information Cards that match the policy. Cards that do not match are greyed out. Cards that have been sent to the site before are placed in the top half of the Identity Selector, cards which have not been sent before are placed in the bottom half.
4. The user decides to send a managed card this time (as opposed to the self issued card which was sent last time). He selects the card and can then choose to either send the card straightaway or click Preview to check its contents before sending. If the card is self issued it could be protected with a PIN, in which case the user will be asked to enter this now. If the card is managed, the user may be asked to enter her password or X.509 key PIN, depending upon the authentication mechanism being used.
5. When the user clicks Send, the Identity Selector fetches the security policy from the IdP (see section 4.2.3) and from this determines the security parameters that are needed in order to request an attribute assertion (claim) for this information card. The card itself says how the user is to be authenticated to the IdP. CardSpace currently supports four authentication mechanisms for managed cards: username and password, KerberosV5 token, X.509 public key certificate and a self-issued token. No authentication is needed for the SIP to issue a self-issued card.
6. The Identity Selector requests an attribute assertion (claim) from the IdP using the security parameters from the IdP's policy and the credentials specified in the

information card. If username-password was specified, the user will have been prompted to provide his password, and this will be encrypted in the request to the IdP. The IdP returns the attribute assertion to the Identity Selector.

7. The Identity Selector returns the attribute assertion/credential to the web browser.
8. The web browser POSTs the credential to the SP (web site).

4.2.2 Service Provider's Security Policy

The SP's policy is an XML document which describes its requirements for the authorization credential that is to be presented. It uses XML elements defined in WS-SecurityPolicy [17] and WS-Trust [19]. Expressing its credential requirements is done through the following policy parameters:

- the issuer – this is the WS-Addressing [18] Endpoint Reference of the Identity Provider who is to issue the credential. This is usually the logical name of the IdP specified as a URI e.g. https://kent.ac.uk/idp. This must match the name of an IdP in one of the information cards stored in the user's Identity Selector. This field can be blank, indicating that any issuer is acceptable to the SP, or it can take the specific value of self-issued (http://schemas.xmlsoap.org/ws/2005/05/identity/issuer/self), meaning that a self issued card is acceptable;
- the token type – this is the type of credential that should be issued and presented e.g. a SAML attribute assertion or an X.509 attribute certificate;
- the attributes (or claims) that the credential should contain. Each one of these attributes can be flagged as mandatory or optional, and
- the proof key – this is used by the SP to prove that the credential presenter is entitled to present this credential. The default value for this is PublicKey, meaning that the credential should contain the pubic key of the holder (user), but it can also take the value of Bearer, meaning that no proof of ownership is required, or symmetric, meaning that the user should hold the symmetric key that is inside the credential.

The SP's policy can also contain a URL pointing to where its privacy policy can be found so that the user can read it before deciding whether to send his credentials to this SP. No automated support for privacy policy enforcement is provided. This is currently a research topic.

The SP's policy can also contain details of how the message should be secured when the credential is sent, for example, the message should be signed by the user if it contains a PublicKey proof key.

The policy is retrieved using the WS-MetadataExchange protocol [20].

4.2.3 Identity Provider's Security Policy

The IdP's policy is used to tell the Identity Selector how it can retrieve a credential from the IdP. The policy is contained within the WSDL that specifies the protocol messages for accessing the IdP's credential issuing web service (called a Security Token Service (STS) in the WS* specifications). The policy contains details of the security that should be applied to the request and response messages, for example whether the XML messages should be digitally signed, or encrypted, or whether security should be performed at the transport level by SSL/TLS. An alternative is that

the XML messages are symmetrically encrypted using a short lived session key. The policy will always contain the X.509 public key certificate of the IdP.

4.2.4 Information Card Contents

Information cards are formatted as an XML document, signed by the IdP that issued them. They contain the following fields:

- the logical name of the issuer as a URI. SPs should ensure that the issuer name they use in their policies matches this name,
- an optional image and language dependent friendly name that can be displayed by the Identity Selector to the user,
- a unique reference number for this card (unique to the issuer)
- the issuing and expiry dates of the card,
- a list of Endpoint References, in decreasing order of preference, from where the associated credential can be obtained and the authentication mechanism that is needed to obtain it. If the authentication mechanism is username/password then the card contains the username of the user whose card this is. Each Endpoint Reference also contains the location of the IdP's security policy that controls access to this endpoint.
- the token types that can be issued, e.g. SAML assertion or X.509 attribute certificate,
- the list of attribute types (claim types) that can be issued,
- whether the name of the recipient SP must be provided or not. If this is required, then the issued credential will be encrypted specifically for this SP so that no-one else can read its contents,
- an optional pointer to the privacy policy of the IdP,
- a flag to indicate whether the SP must have been identified by an X.509 public key certificate or not. This prevents the IdP from issuing a confidential credential that could be sent to an unidentified SP.

4.2.5 Limitations of Information Cards

Good as they are, information cards currently have a number of limitations. Firstly they only support 4 authentication mechanisms: username/password, Kerberos V5 ticket, X.509 public key certificate and a self issued token (key pair). Other popular mechanisms such as one time passwords are not supported. Secondly it is not specified how a user collects her information cards for inclusion in her identity selector. This is left up to the individual IdPs to determine. The Identity Selector simply provides an interface for importing Info Cards into it from the local filestore, but how they get to the local filestore is not standardized. However, the biggest limitation of Information Cards is that the user can only select one card to present to a service provider for any given session. In many cases this is insufficient; for example, consider trying to purchase a car road tax license over the Internet. You may need to provide the following credentials: a credit card, a road worthiness certificate for your car, an insurance certificate, and a driving license. Each of these may be issued to you by different authoritative sources (IdPs). The current CardSpace model will only allow you to present one of these credentials to the SP. Research conducted by the University of Kent has devised and implemented a solution to this using a new

component called a Linking Service [21]. This allows the user to link his various IdPs together in a privacy preserving manner, by interacting with the Linking Service prior to service provision. It appears to work well with the Shibboleth model as it uses the Shibboleth style of interaction with the user. But it does not employ the CardSpace interface, and integrating this with CardSpace means that the user still only selects one card at service provision time, with the Linking Service providing the remainder. So there is still research to do to design a system which will allow the user to dynamically select several cards at service provision time.

5. Conclusion

This chapter has provided an introduction to the complex topic of federated identity management. FIM is still very much an active research topic and is likely to continue to be so for many years to come. This is because there are so many different issues to consider, some of which are competing or diametrically opposed to each other. Issues include: ease of use, user privacy, strong security, single sign on, total cost of ownership, user profiling and retention of users by service providers, scalability, fine grained access control, personalization of services, and anonymity. Whilst we have not covered all of these issues in this chapter, nevertheless I hope this chapter has been informative and enjoyable to read.

References

1. ITU-T. “NGN identity management framework”. Recommendation Y.2720
2. ITU-T. “Baseline capabilities for enhanced global identity management trust and interoperability”. Draft New Recommendation ITU-T X.1250 (X.idmreq), Feb 2009.
3. ISO/ITU-T. “The Directory: Models” ISO 9594-2/ITU-T Rec. X.501 (2009)
4. R. L. "Bob" Morgan, Scott Cantor, Steven Carmody, Walter Hoehn, and Ken Klingenstein. “Federated Security: The Shibboleth Approach”. Educause Quarterly.Volume 27, Number 4, 2004
5. Arun Nanda, Michael B Jones. “Identity Selector Interoperability Profile v1.5” Microsoft Corporation, July 2008. see http://download.microsoft.com/download/1/1/a/11ac6505-e4c0-4e05-987c-6f1d31855cd2/Identity_Selector_Interoperability_Profile_V1.5.pdf
6. Kim Cameron. “The Laws of Identity”, May 2005. Available from http://www.identityblog.com/?p=352/#lawsofiden_topic3
7. OASIS “SAML 2.0 profile of XACMLv2.0”. OASIS standard. 1 February 2005
8. OECD “Guidelines on the Protection of Privacy and Transborder Flows of Personal Data”. 23rd September, 1980
9. Liberty Alliance Project. “Liberty ID-WSF Web Services Framework Overview” Version: 2.0” Available from http://www.projectliberty.org/specifications__1
10. OASIS. "Level of Assurance Authentication Context Profiles for SAML 2.0" Working Draft 01. 01 July 2008
11. OpenID Authentication 2.0 – Final. Dec 5th 2007. Available from http://openid.net/specs/openid-authentication-2_0.html
12. OASIS “eXtensible Access Control Markup Language (XACML) Version 2.0” OASIS Standard, 1 Feb 2005

13. David Chadwick, GansenZhao, Sassa Otenko, Romain Laborde, Linying Su and Tuan Anh Nguyen. “PERMIS: a modular authorization infrastructure”. Concurrency And Computation: Practice And Experience. Volume 20, Issue 11, Pages 1341-1357, 10 August 2008.
14. For info about ShARPE see http://www.mams.org.au/confluence/display/SHA/ShARPE and http://www.federation.org.au/twiki/bin/view/Federation/ShARPE
15. David Chappell. “Introducing Windows CardSpace”. MSDN. April 2006. Available from http://msdn.microsoft.com/en-us/library/aa480189.aspx
16. Sebastian Gajek, Jorg Schwenk, and Chen Xuan. "On the Insecurity of Microsoft's Identity Metasystem". Technical Report TR-HGI-2008-003, Ruhr-Universitat Bochum, June 2008. Available from http://demo.nds.rub.de/cardspace/GaScXu08_CardSpaceTR.pdf
17. OASIS. “WS-SecurityPolicy 1.2”, OASIS Standard,1 July 2007
18. W3C. “Web Services Addressing (WS-Addressing)”. W3C Member Submission, 10 August 2004
19. OASIS, “WS-Trust 1.3”, OASIS Standard, 19 March 2007
20. BEA Systems, Computer Associates, IBM, Microsoft, SAP, Sun Microsystems, and webMethods. “Web Services Metadata Exchange (WS-MetadataExchange)” Version 1.1 August 2006
21 David W Chadwick, George Inman. “Attribute Aggregation in Federated Identity Management”. IEEE Computer, May 2009, pp 46-53