

# Federated Identity Management

David Chadwick  
University of Kent

# Some Definitions

- What is Identity?
- A whole set of attributes that uniquely characterise a person
  - hair colour, sound of their voice, height, name, qualifications, past actions, reputation etc.
- Attribute – a property, quality or characteristic of an entity
- Identifier – a string used to uniquely identify an entity in a domain. Often used as login id or primary key in a database. A special type of attribute since it is usually the only one *on its own* that can uniquely identify an entity in a domain.
  - X.500/LDAP DNs, IP addresses, DNS names, URIs, key IDs, login IDs, 128 bit random numbers are all identifiers.
- Attribute Assertion – a statement made by an authority that an entity has a particular attribute. An authority can be the entity itself or a (trusted) third party.
- Attribute Certificate/Authorisation Credential – a cryptographically protected (usually digitally signed) attribute assertion that can be validated
- Attribute Authority (AA) – an authoritative source for asserting attributes about entities
- Service Provider – an entity that provides a service to clients
- Identity Provider – an entity that provides an authentication service, and is also an AA for a set of identity attributes of its users

# Federated Identity Management

- From the RSA Web Site
- “A federated identity is a single user identity that can be used to access a group of web sites bound by the ties of federation. Without federated identity, users are forced to manage different credentials for every site they use. This collection of IDs and passwords becomes difficult to manage and control over time, offering inroads for identity theft.”
- “Federated identity management builds on a trust relationship established between an organization and a person. A federated identity makes it possible for the end user to use one trust relationship to access information with another, related company without establishing new credentials.”

# Federated Identity Management

- From Microsoft's web site
- "Federated systems need to interoperate across organizational boundaries and connect processes utilizing different technologies, identity storage, security approaches and programming models. Within a federated system, identities and their associated credentials are still stored, owned and managed separately. Each individual member of the federation continues to manage its own identities, but is capable of securely sharing and accepting identities and credentials from other members' sources."
- From IBM Tivoli's web site
- "Federated identity management can be defined as an industry framework built on top of industry standards that let subscribers from disparate organizations use their internal identification data to obtain access to the networks of all enterprises in the group".
- SO WHAT IS FIM?

# Identities, Identifiers and FIM

- Identifiers are assigned within a domain to uniquely identify an entity. They usually have no meaning outside of the domain of issuance
- FIM requires identity information to be passed between domains, therefore
- We need to pass (signed) attribute assertions between domains in order to identify and authorise users between domains.
- FIM is not just Single Sign On, although SSO is part of FIM. Why?

Because you need authorisation information as well as authentication information for FIM. SSO is authenticating once in order to access many systems. SSO can provide authz/access control as well if identity based ACLs are used at each of the systems. But in general, where ABAC is used to provide FIM, then SSO needs to be combined with the distribution of different sets of attributes to the different systems so that appropriate access can be granted.

## So What is Federated Identity Management ?

- A group of organisations that set up trust relationships which allow them to send attribute assertions about users identities between themselves, in order to grant users access to their resources
- A user can use his credentials (authn and authz) from one or more identity providers to gain access to other sites (service providers) within the federation

# Authentic vs. Valid Credentials

- Authentic credentials are ones that have not been tampered with and are received exactly as issued by the issuing authority
- Valid credentials are ones that are trusted for use by the target resource site
  - 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 Tesco's or LIDL.
  - Example 2: My Amex card is authentic. I can use it to buy groceries in Tesco, so it is valid there, but I cannot use it to pay motorway tolls in France. It is not valid there, but it is still authentic.

# Kim Cameron's 7 Laws of Identity

## 1. User Control and Consent

- *Technical identity systems must only reveal information identifying a user with the user's consent.*

## 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.*

## 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.*

## 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 handles.*

KC's 7 LAWS OF IDENTITY - see <http://www.identityblog.com>



# Kim Cameron's 7 Laws of Identity

## **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. 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.*

## **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.*

## Some Early FIM Systems

- Microsoft's Passport
- UK Athens

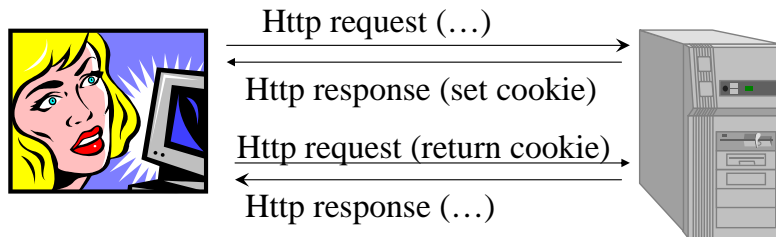
## Some Current FIM Systems

- Shibboleth
- Liberty Alliance
- Cardspace
- Higgins
- OpenID
- OAuth

## But first some background info

### HTTP Cookies

- Cookies – allow a web server/site to store state information for itself (often encrypted) on the user's browser
- A site can store many cookies, and the client should return them all when it returns to the site
- Often used to enable SSO, since the site can tell if a user is already authenticated or not



FOSAD 29-30 Aug 08

© 2008 David Chadwick

11

A cookie is simply a small piece of data to be stored on the client's PC. A cookie comprises a *name=value* pair and a set of descriptive attributes such as its version number, domain name, path and lifetime. The storing and retrieving process is usually invisible to the user.

Cookies are "created" when a web server sends a **Set-cookie2** HTTP MIME header attached to a response for a URI. For example the following cookie might be returned after a user logs into a site. It holds the user's identity:

HTTP/1.1 200 OK

Set-Cookie2: Customer="Bin Laden"; Version="1"; Path="/acme"

Once set, the user's browser should return the cookie whenever a matching URI is accessed again. Note that clients can read, delete and modify cookies if they are not properly protected. For this reason many servers will encrypt their cookies before sending them to the client. But nothing can stop the user deleting the cookies or even giving them to another user if she wants to.

The cookie *name=value* pair and associated attributes are returned by the browser, in the HTTP request header. For example, suppose the user returns to the previous site to select a shopping item (the POST command is used for this):

POST /acme/pickitem HTTP/1.1

Cookie: \$Version="1"; Customer="Bin Laden"; \$Path="/acme"

followed by [data holding the selected item] to be stored by the web server

The server might return the following cookie to show that the item has been added to the user's shopping basket

HTTP/1.1 200 OK

Set-Cookie2: Part\_Number="Rocket\_Launcher\_0001"; Version="1"; Path="/acme"

Suppose the user now returns to the site, to buy the item, then both cookies are returned

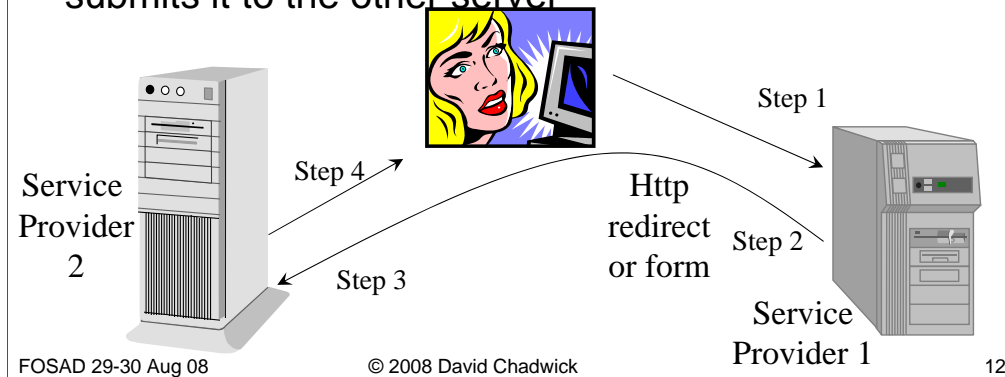
GET /acme/buy HTTP/1.1

Cookie: \$Version="1"; Customer="Bin Laden"; \$Path="/acme";

Part\_Number="Rocket\_Launcher\_0001"; \$Path="/acme"

# HTTP Redirect and Form-POST

- Http Redirect (status code 3xx) – allows one server to pass information to another server via the browser, as info in a URL
- Http Form-POST – one server builds a form with an action to POST it to another server, delivers the form to the browser in the message body, which then submits it to the other server



The Http redirect service works as follows:

1. The user agent sends an HTTP request to Service Provider 1 (typically a GET). In this step the user has either clicked on a link in a Web page or has typed in a URL.
2. Service Provider 1 responds with an HTTP response with a status code of 302 (a redirect) and an alternate URI in the Location header field (in this case the Location URI will point to Service Provider 2) Often the URI will contain a second, embedded URI pointing back to the original service provider (Service Provider 1 in this example).
3. The user agent sends an HTTP request to Service Provider 2 (typically a GET), specifying the complete URI taken from the Location field of the response returned in Step 2 as the argument of the GET. This URI will contain the second embedded URI pointing back to Service Provider 1, if it was present in Step 2..
4. Service Provider 2 can then act on the URI and respond to the user agent. If there was an embedded URI pointing back to Service Provider 1, then Service Provider 2 can respond with a new redirect whose Location header field contains the URI pointing to Service Provider 1, and the two service providers can continue talking to each other in this way via the user agent.

Not that there is a limit to the amount of information that can be carried in a redirect URI which is usually 1024 characters.

The form-POST-based redirection works as follows:

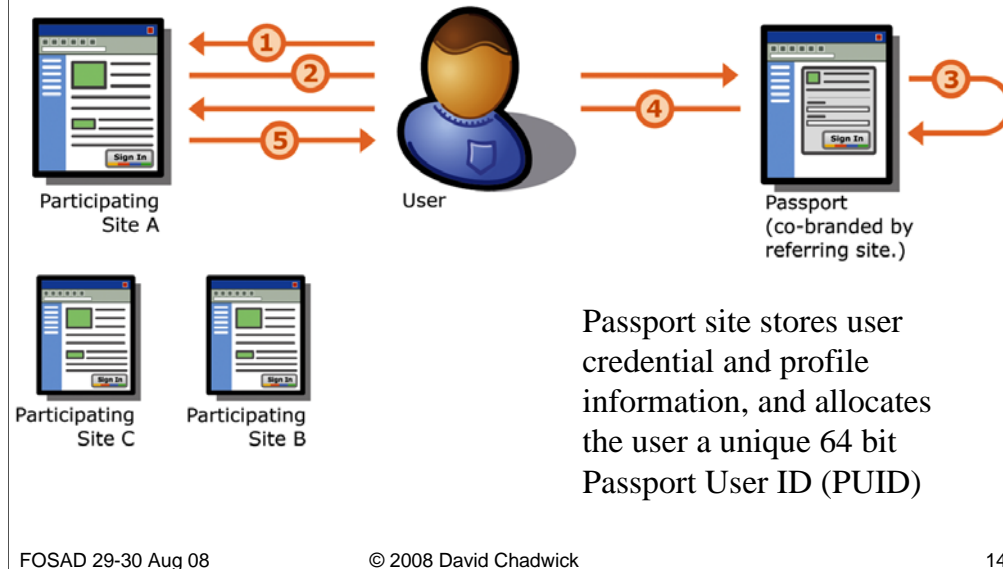
1. This is the same as before.
2. Service Provider 1 responds by returning a HTML form to the user agent. The form contains an action parameter pointing to Service Provider 2 and a method parameter with the value of POST. Arbitrary data may be included in other form fields. The form may also include a JavaScript or ECMAScript fragment that causes the next step to be performed without user interaction.
3. Either the user clicks on the Submit button on the form, or the JavaScript or ECMAScript executes. In either case, the form and its arbitrary data contents are sent to Service Provider 2 via the HTTP POST method.
4. Service Provider 2 acts on the contents of the form, and can if it wants to, reply with a form-POST-based redirection in the opposite direction to Service Provider 1.

There is no limit to the amount of information that can be carried in a form and a Http POST command.

## Microsoft's .NET Passport

- .NET Passport is an authentication system that allows users to access multiple sites using the same credentials
- Each site remains in charge of its own authorisation, and may use Passport information to help in this
- How does it work? Users register at a site, but their credentials and profile information are stored centrally by Microsoft at the Passport server. This means that sites must trust Microsoft to hold user credentials and authenticate users correctly.

# The Registration Process



1. In this example the user browses to Site A, a participating site or service (or browses to [www.passport.com](http://www.passport.com)), and they click the “Sign In” button (or click the “Register” button on [Passport.com](http://Passport.com)).
2. The user is redirected to a co-branded registration page displaying the registration fields that were chosen by Site A. (The minimum number of fields required is two: email name and password.) Here the user chooses whether or not they want to opt in to share their information with other Passport-enabled sites that they sign in to.
3. The user reads and accepts terms of use (or declines, and the process ends), and submits the form. (On [Passport.com](http://Passport.com) the user is shown a congratulations page and the sign up process ends here.)
4. The user is then redirected back to Site A with their encrypted authentication ticket and profile information attached.
5. Site A decrypts the authentication ticket and profile information and continues their registration process, or grants access to their site.

NOTES: Sites B and C do not receive any information about the user. The user does not need to download any software.

## Credential Information Stored by Passport

- The following are mandatory: e-mail address (unique identifier) and password
- The following are optional: secret questions and answers, mobile phone number and PIN, security key

*All referenced by a Passport Unique ID (PUID)*

## Profile Information Stored by Passport

- The following attributes are stored by Passport if the participating sites require it, and are shared between sites if the user opts-in
  - Birth Date, Country / Region, First Name, Gender, Last Name, Occupation, Postal Code, Preferred Language, State, Time Zone

FOSAD 29-30 Aug 08

© 2008 David Chadwick

15

The e-mail address is special in that it is treated as both a credential and a profile attribute. It is the unique identifier attribute of the user that is used by Passport. Passport then issues PUID for the user which is used at the unique identifier for the participating sites.

The secret questions and answers are used to re-authenticate the user if he forgets his password.

The mobile phone number and PIN are only used to authenticate the user, if the user is visiting a site from a mobile device.

The Security Key may be required by a participating site for additional security

None of the credential information leaves the Passport site or is shared with any participating site (except the e-mail address if the user chooses this).

# Privacy Protection - User Opt-In

- User can choose to share e-mail address, name and other profile information with all participating sites (but must be same for all sites)



**Tired of registration forms?** You can speed registration and get personalized services at participating sites by sharing your .NET Passport information with them when you sign in. Select the boxes below to choose how much of your .NET Passport information Microsoft can share with other companies' .NET Passport sites at sign-in:

- ☐ Share my e-mail address.
- ☐ Share my first and last names.
- ☐ Share my [other registration information](#).

[Tell me more about .NET Passport, privacy, and security.](#)

Users can choose how much of their information can be shared between participating sites. This allows users to protect the privacy of their information that they give to Passport.


A participating site may base its authorisation on the shared profile information and/or on local information held at the participating site.

The unique Passport UID is used to uniquely identify the user by the participating site, and to link the user's attributes to data held at the participating site.



# Human Interaction Protocol - CAPTCHAs

- **C**ompletely **A**utomated **P**ublic **T**uring test to tell **C**omputers and **H**umans **A**part
- Designed to stop automated user registration programs and possible DOS attack by flooding registration process
- User is asked to type in some characters, that most programs are incapable of reading

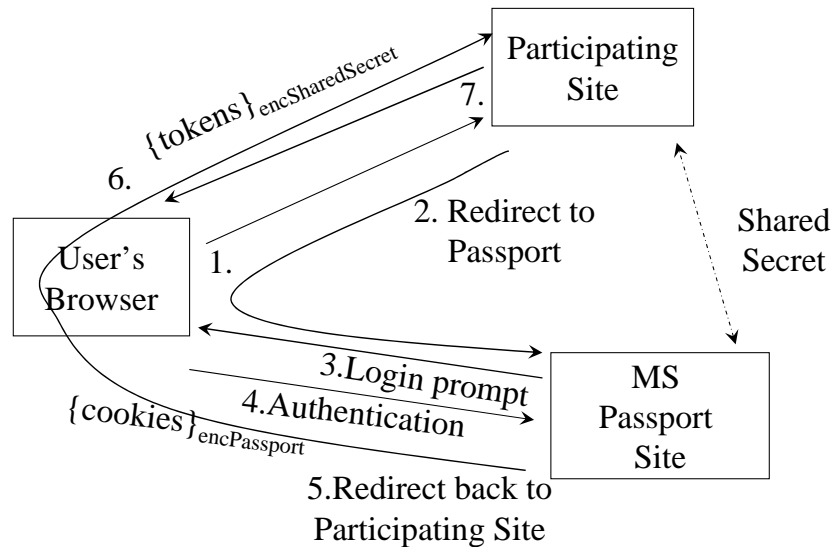
<b>Registration Check</b>	Type the characters that you see in this picture. <a href="#">Why?</a>
	
<a href="#">I can't see this picture.</a>	
<input type="text"/>	
Characters are not case-sensitive.	

FOSAD 29-30 Aug 08      © 2008 David Chadwick      17

If registration in Passport was fully automated, then programs could register users automatically without any human involvement. This would allow a program to register thousands of users e.g. from a user database with or without their knowledge, or for multiple programs to effect a Denial Of Service attack on the Passport service. By requiring users to interpret what characters they see on the screen and type it in stops most programs from being able to do this, as the characters are distorted.

These are often known as CAPTCHAs – Completely Automated Public Turing test to tell Computers and Humans Apart

# Authentication with .NET Passport



FOSAD 29-30 Aug 08

© 2008 David Chadwick

18

.Net Passport authentication is based on Kerberos authentication. .NET currently uses proprietary formatted tokens, but MS originally said that they would migrate towards standard Kerberos tokens in the future.

1. The user contacts the participating site and tries to access a protected page or clicks on the Passport logo
2. and is redirected to the Passport site.
3. The Passport site displays the login page and
4. the user enters their login credentials (email address and password). These are sent encrypted to the Passport site via SSL. The user is authenticated.
5. The Passport site sends a redirect reply to the user, containing several cookies (a ticket granting cookie, a Participating Sites cookie, an authentication cookie and a profile cookie) which are stored there to shortcut future authentication attempts within the lifetime of the cookies. These cookies are encrypted with a secret key known only to Passport.
6. The user is redirected back to the Participating Site and the redirect message contains two encrypted tokens within it. Both of these tokens are encrypted with a secret key shared between the Participating Site and Passport. The first token contains the Authentication Ticket, the second the Profile Information of the user.
7. The Participating site decrypts the tokens and from these knows that the user has been successfully authenticated by Passport, and is given the user's profile attributes and the PUID. The Participating site can then turn these into its own two equivalent cookies, which it encrypts with its own secret key, and stores them in the user's browser. It can use these as the user navigates through secure pages to know that the user has been successfully authenticated.

## Moving Between Participating Sites

- When a user moves to another Participating Site (step 1), the site redirect the user to the Passport site (step 2)
- The user's client sends the Authentication cookie and Profile cookie to Passport during redirection. Passport then knows the user has already successfully authenticated (modified step 2)
- The Participating Sites cookie on the user's machine is updated by Passport and the user is redirected back to the Participating Site (step 5)
- The Participating Site receives the encrypted tokens from Passport and knows the user has been authenticated (step 6)
- When the user logs out of Passport, all cookies are deleted and the Participating Sites cookie is used to clean up all Participating sites computers

FOSAD 29-30 Aug 08

© 2008 David Chadwick

19

The Participating Sites cookie on the user's machine holds a list of all the Participating sites that a user has visited within the current session. This is updated by Passport every time the user moves between Participating sites.

## Why did Passport Fail?

- Because all participating sites have to trust Microsoft to hold the identity of the user, and to authenticate the user properly
- Fails Kim Cameron's 3<sup>rd</sup> Law of Justifiable Parties. Why should Microsoft be involved in a federation between a car hire company and a hotel? It might be OK for Microsoft related site federations such as Hotmail and MSN, but not for all federations between all commercial companies.

# Athens

- Athens was the de facto standard for secure access management to online services for the UK Education and Health sectors (replaced by Shibboleth on 1/8/8)
  - Originally designed by a team at the University of Bath (JISC-funded)
  - Now owned, developed and operated by EduServ (<http://www.eduserv.org.uk>)
  - Besides JISC, Athens is also used by the National Health Service (National Electronic Library for Health)
  - 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, Oxford University Press

These cover both education and NHS, unless otherwise stated

497 FE + HE sites; 769 sites total including NHS

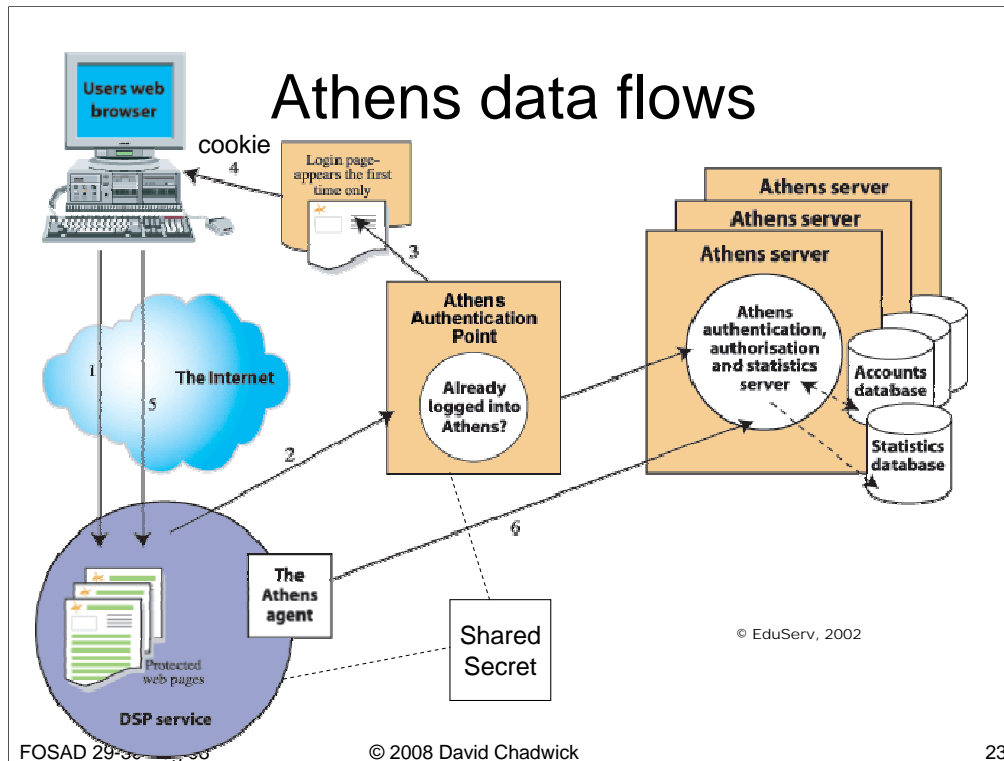
Approximately 2 million user accounts

Average authenticated access requests per day 85,650  
(August 2002)

51 content providers, offering between them 249 Athens-controlled resources

## How does it work?

- Originally a “trusted third party” network service
  - Essentially a large database of user IDs and passwords and authorisation data (says which sites which users can access)
  - Replicated to provide a resilient service
  - Each participating college or university administers its own part of the database
  - Content providers refer access requests to Athens for validation, and run special plug-in software to achieve this
  - Users login over SSL, so that their passwords are encrypted



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 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 an 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.

## Why did Athens Fail ?

- In the UK it didn't. It was very successful with millions of users and hundreds of sites
- But it uses proprietary protocols, and therefore other countries were not prepared to adopt it.
- Also sites cannot leverage it to set up their own mini-federations
- The UK, US, Europe and Australia are now moving to Shibboleth which provides similar functionality but is standards based and uses open source software



# Shibboleth

- Shibboleth is a project run by the Internet2 consortium in the USA (academic partners and IBM)
- It defines a protocol for providing users with access to remote resources via authentication at their home site and authorisation via a set of user attributes provided by the home site
- Shibboleth access takes place in two stages
  - Obtaining a handle(authn assertion) for an authenticated user
  - Using the handle to get a set of attribute assertions for the user

FOSAD 29-30 Aug 08

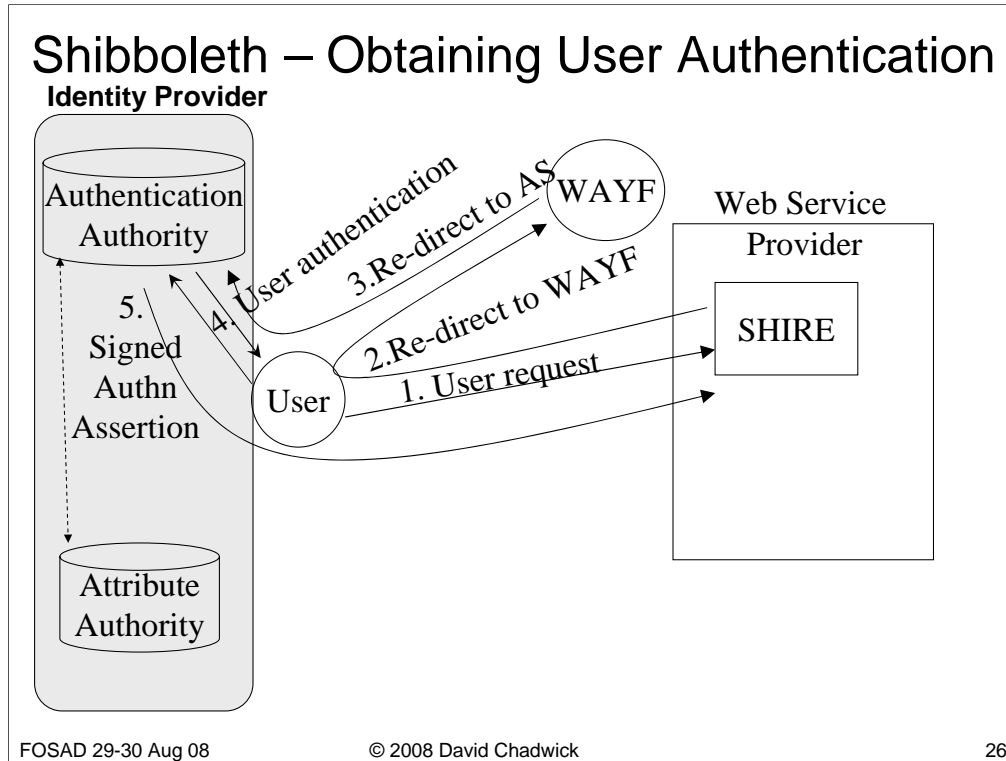
© 2008 David Chadwick

25

More details about Shibboleth can be obtained from  
<http://shibboleth.internet2.edu/>

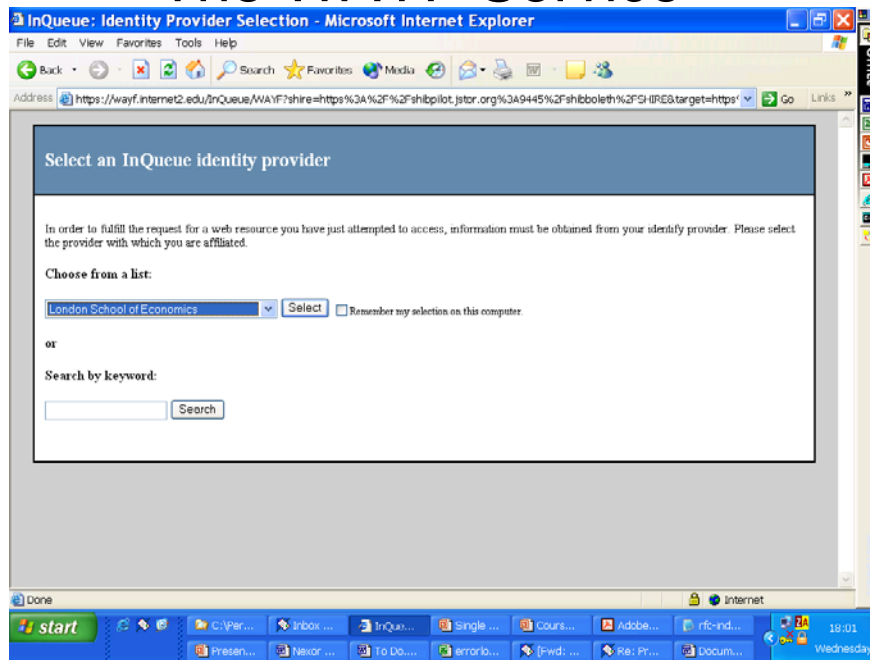
Shibboleth can be demonstrated by going to  
<http://www.library.gsu.edu/shib/>

And then selecting the JSTOR trial at <http://shibpilot.jstor.org/>



1. The user makes a request to a web site (the Service Provider). 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 find out attributes of the user in order to grant him/her access. The Shibboleth Indexical Reference Establisher (SHIRE) is the service that will try to get an authentication assertion about the user. It does this by sending the user back to his home site via a Where Are You From (WAYF) service.
2. The SHIRE 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. The WAYF knows the name and location of the Authentication Authority for each origin site that is participating in Shibboleth. The user picks his home site, and then
3. the User is re-directed to the Authentication Server at his home site by the WAYF service.
4. The Authentication Service is responsible for making sure the user is authenticated locally at the origin site, and for creating a SAML authentication assertion containing a one-off random handle that can be used to retrieve attributes about the user. The Authentication Service prompts the user to login and provide his authentication tokens. The home site can use whatever type of authentication it likes e.g. username/password, Kerberos, digital signatures etc. This is of no concern to the remote Web Service. Once the user has authenticated him/her self, the Authentication Authority produces a SAML Authentication Assertion containing a random handle. The handle ensures that the user's name remains private to the local site, and the Web Service will never know the identifier/username of the user that is accessing it. Thus Shibboleth automatically provides Privacy Protection of user identifiers.
5. The Authentication Server passes the handle (in the form of a SAML Authentication statement) along with additional info back to the user's browser inside an HTML form that POSTS the data back to the destination SHIRE. This info includes the location of the AA server at which the handle will be usable. This message is digitally signed by the Authentication Authority to prove its authenticity. The SHIRE must check the signature and the message contents to ensure its validity.

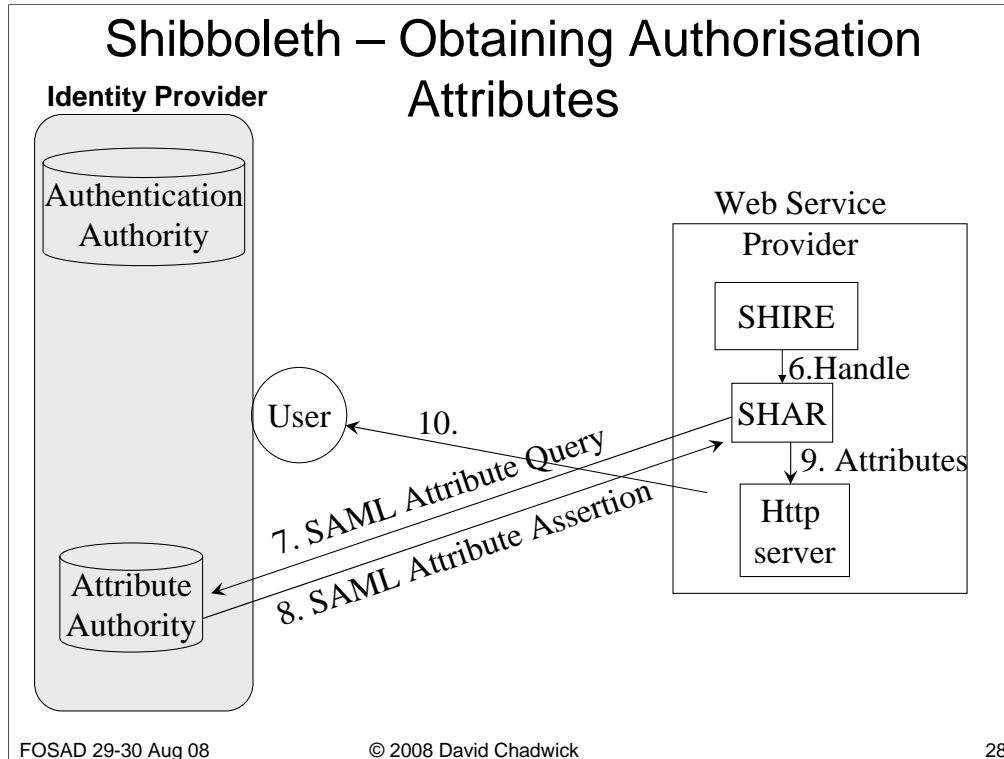
# The WAYF Service



FOSAD 29-30 Aug 08

© 2008 David Chadwick

27



6. The SHIRE passes Authentication Assertion containing the handle, AA contact info, and the origin site name to the SHAR (Shibboleth Attribute Requestor).
7. The SHAR sends a SAML Attribute Query Message to the Attribute Authority (AA) server at the user's home site. This request needs to be protected. It needs to be mutually authenticated and have message integrity. SSL with client side authentication would satisfy this.
8. The AA server returns SAML Attribute Assertion to the SHAR. This message needs to be protected by mutual authentication, message integrity and message confidentiality. Thus SSL with client side authentication would satisfy these requirements.
9. Once the SAML response is received and validated, the embedded attributes are passed to the web service by the SHAR.
10. The Web service can now grant or deny access to the user based on these attributes. The web service could make use of a PDP to determine which privileges to grant to the user, based on his attributes.

# Privacy Protection in Shibboleth

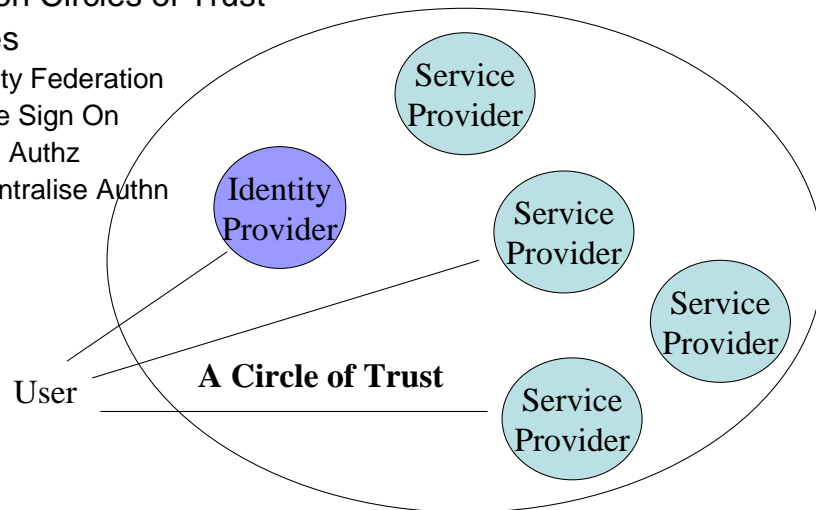
- Privacy Protection is an important feature of Shibboleth
- User identifiers remain private to the home site. Users are only identified via a random handle, which changes each time the user accesses a site
- Users are authorised to access resources based on their identity attributes rather than their identifiers e.g. ISI student at Salford, professor at UCL etc.
- User attributes are only released to resource sites according to an Attribute Release Policy set by the home site and the user, thereby ensuring user privacy, and the attributes are encrypted during transfer to the resource site.

## Weaknesses with Shibboleth

- Only provides attributes from a single attribute authority to the service provider
- Open to phishing attacks. An evil site can put up a false WAYF and point the user to a site masquerading as his IdP in order to capture the user's login credentials.
- Does not provide Single Sign Off
- As currently implemented, most IDPs only send non-identifying attributes to the SP (such as student at Kent), so Shibboleth cannot be used for services that need to know who the user is in order to give a personalised service e.g. data repositories

# Liberty Alliance

- A consortium of 50 companies including Sun, HP, banks, telecom providers, Visa, Mastercard, and various suppliers
- Based on Circles of Trust
- Provides
  - Identity Federation
  - Single Sign On
  - Open Authz
  - Decentralise Authn



FOSAD 29-30 Aug 08

© 2008 David Chadwick

31

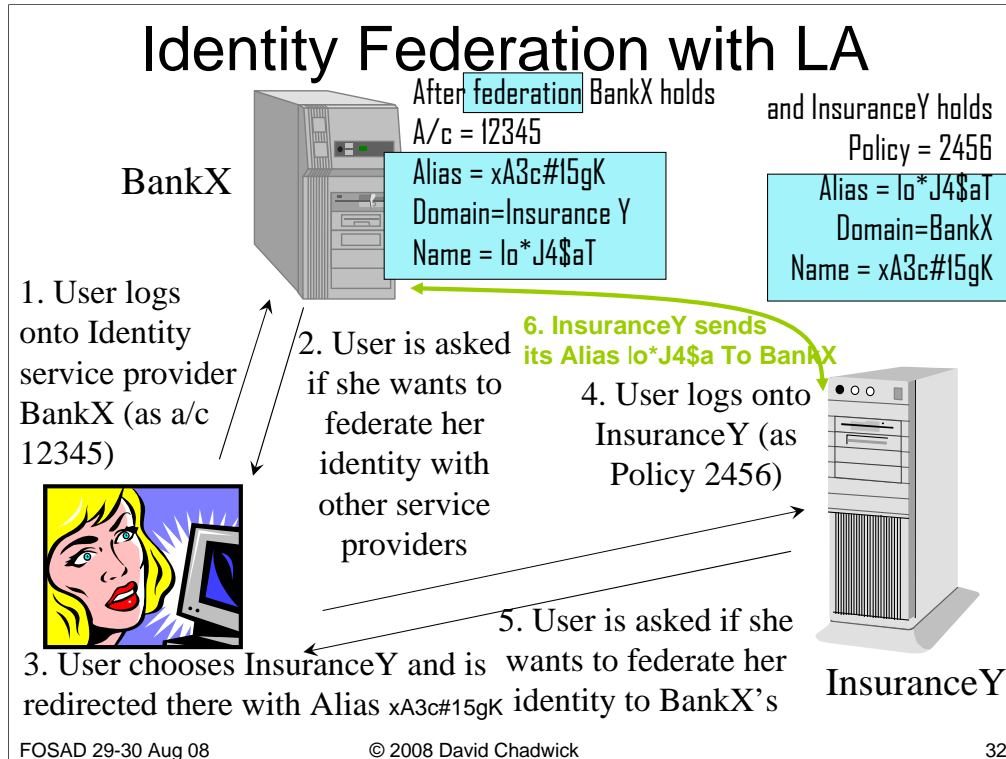
A Liberty Alliance circle of trust comprises a number of service providers and identity providers.

Service providers are organizations offering Web-based services to users. This includes most organizations on the Web today: Internet portals, retailers, transportation providers, financial institutions, entertainment companies, not-for-profit organizations, governmental agencies, etc.

Identity providers are service providers that provide user authentication and attribute services and offer business incentives so that other service providers will affiliate with them. Establishing such relationships creates the circles of trust.

Identity federation is based upon linking users' service providers and identity provider accounts. This account linkage, or *identity federation*, underlies the other Liberty services.

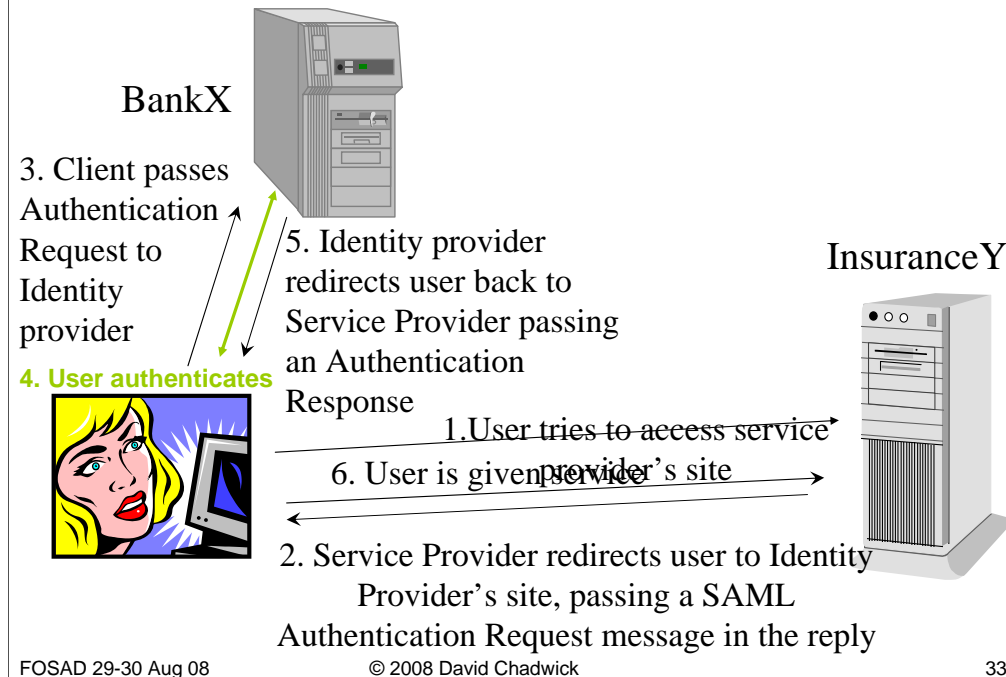
Single sign-on enables users to sign on once with an identity provider (i.e. with a member of a circle of trust) and subsequently use this with the various Websites (service providers) in the circle of trust without signing on again.



Each user is typically known by a different login identity at each service provider site (e.g. A/c 12345 at BankX and Policy 2456 at Insurance Co Y). Identity federation causes these identities to be linked together, but the user still continues to use each site specific login identity at that site (unless SSO or federated IDP login is being used). No site knows the login identity used by the user at any other site i.e. login identities are not exchanged. Rather the login identities are referenced by Liberty handles. A Liberty handle is a permanent identifier known by both sites and unique within the circle of trust. A Liberty handle could be created by performing a hash of the user's login identity and other information known only to the provider. Because the handle is at least 128 bits, it is virtually guaranteed to be unique. Note that each handle is only known by the two sites that are federated together, and a service provider will use different handles for the same user with different service providers. The user is in charge of federating sites together, and new handles are created each time, so that multiple sites cannot exchange information about the same user by using one handle.



# Single Sign On with LA



1. User tries to access SP site but is not authenticated.

2. The user is redirected back to the Identity Provider with a redirect response such as

<HTTP-Version> 302 <Reason Phrase> <other headers>

Location: <https://<Identity Provider Single Sign-On Service host name and path>?<AuthnRequest>>  
<other HTTP 1.0 or 1.1 components>

Note that the redirection specifies that an SSL connection (https) must be set up between the user's browser and the Identity Provider's web server.

Alternatively, a form-POST reply may be sent in the message body, such as

```
<html> <bodyonLoad="document.forms[0].submit()">
  <form action="https://< Identity Provider Single Sign- On Service host name and path" method="POST">
    <input type="hidden" name="LAREQ " value="<base64 encoded Authn Request>" >
  </form>
</body>
</html>
```

The Authentication Request is a digitally signed SAML Authentication Request message and contains the following parameters/requests

Prompt the user for their credentials if the user is not currently authenticated.

Prompt the user for their credentials, even if the user is presently authenticated

Federate the user's identity at the Identity Provider with the user's identity at the Service Provider.

Issue an anonymous and temporary identifier for the user to the service provider (if federation is not required)

Use a specific authentication mechanism for the user (for example, smartcard-based authentication or username/password-based authentication)

Restrict the ability of the Identity Provider to proxy the authentication request to additional identity providers.

3. The client passes the Authentication Request to the Identity Provider.

4. If the user is already authenticated, the Authentication Response is sent back to the Service provider via a Http redirect or form-POST message body. If the user isn't authenticated, or the Service Provider wants her to be authenticated again, then the Identity Provider will prompt the user for their credentials across the SSL link before returning the Authentication Response.

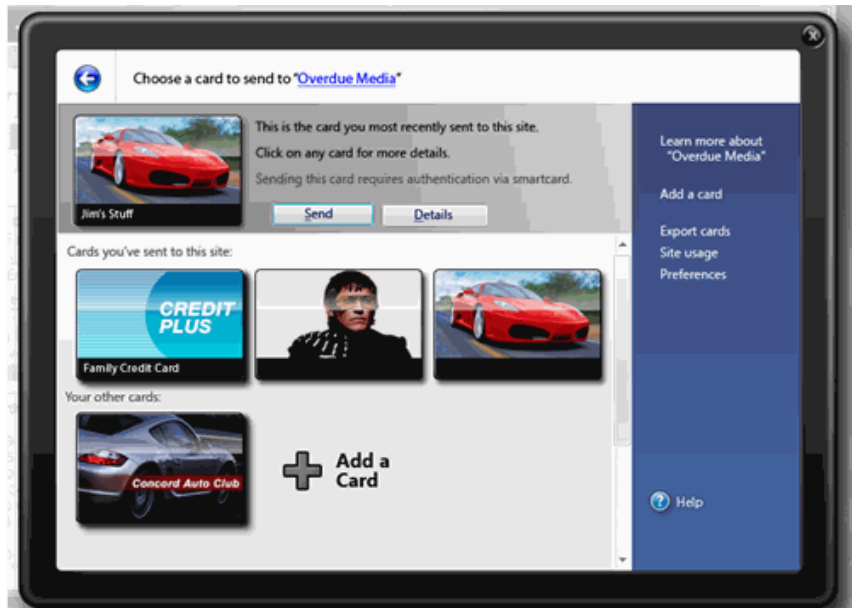
## Other Liberty Specifications

- Discovery Service – enables web services to find out the locations of identity based services for a particular user (or group of users)
  - e.g. Where is the calendaring service and degree certificate issuing service for a user?
- People Service – enables users to share identity based information with other users and grant access to their identity based information
  - e.g. share photographs stored at some service with their friends
- Others still being defined

# Microsoft's CardSpace

- Latest Identity Management system added to MS Vista
- User has a set of InfoCards on his desktop, held in his Identity Selector, in which each card represent the user's profile at an Identity Provider (IDP) – *managed* cards
- User can also create his own *self issued* cards
- User goes to SP site, which sends its requirements for authentication/authorisation (security policy) back to the user's web browser
- The InfoCards that fulfil the SP's policy are lit up, ones that dont are greyed out
- User clicks on the card she wants to send to the SP
- Cardspace/web browser then goes to the IDP site, the user authenticates to the IDP using whatever scheme the IDP requires (un/pw, Kerberos, X.509 cert), and is redirected back to the SP site, carrying a security token signed by the IDP to verify that the user was authenticated according to the SP's policy, and carrying any authorisation attributes that the IDP has

# Example CardSpace Screen



FOSAD

36

# CardSpace Protocols

- Uses Web Services protocols
  - WS-MetadataExchange between the user's browser and the SP for fetching the security policy
    - although HTTPS can be used
  - WS-SecurityPolicy for describing the security policy
    - although a HTML document can be used
  - WS-Trust between the user's browser and the IDP for fetching the security token
  - WS-Security between the user's browser and the SP for delivering the security token
    - although HTTPS can be used

# Limitations of CardSpace

- Nice user experience/metaphor, but...
- Only a single infocard (and hence single security token) can be sent to an SP
  - cant send student union card + credit card to an SP to get a student discount on purchasing a book
- Cant (easily) move cards between machines, mobiles, PDAs etc.
  - all cards are held in a .crds backup file which has to be protected (similar to root CA certificates in a PKI) otherwise user can be duped into sending his login un/pw to a spoof IDP
- Limited to latest Windows PCs, so...

# Higgins

- Open source information cards and identity framework being developed by IBM, Novell, Oracle, CA, Google, Parity and others, designed to interwork with Cardspace and many other protocols
- Goal is to use same card icon and identity selector metaphor as Cardspace, but to integrate identity data from multiple heterogenous sources held in multiple formats e.g. LDAP directories, SQL DBs, social networking sites etc.
- Through a common Context Data Model and support for multiple security tokens, claims, credentials, certificates etc.

# Higgins Context Data Model

- Context Data Model is used to unify identity data held in different underlying systems (or contexts). Based on RDF triples: subject predicate object. (Also has similarities with LDAP data model).
- Entity – a real life person, animal etc that is represented as different nodes in different contexts e.g. a credit card holder in Visa database, a university employee in HRM.
- Digital Subject/Node – a representation of an entity in a given context
- Context – a set of nodes that are disjoint from other sets of nodes (contexts)
- Attribute – a property of a node (predicate)
- Attribute value – an object that holds a literal or complex value
- Node correlation – a kind of attribute whose value identifies a target node that is a different representation of the same entity
- Node relation – a kind of attribute whose value identifies a target node (in same or different context) related to the subject node holding this attribute



# Higgins OWL (HOWL)

- HOWL is used to define (the upper ontology of) the Higgins Context Data Model (i.e. superclasses, properties and relationships)
- Every Context then defines its own lower ontology based on higgins.owl (i.e. it must import higgins.owl definition file). The lower ontology defines the OWL classes, properties and instances that it uses (e.g. visa person, visa card number, credit limit etc in a Visa Higgins system)
- Context Providers should be able to export their context ontologies and map their underlying data repositories into Higgins Contexts
- Higgins thus acts as a wrapper around existing identity repositories

## Limitations of Higgins

- Only supports user selection of a single infocard (as in MS CardSpace)
- Client identity selector only supported in Firefox browsers
  - V1.1 (Spring 2009) will have Adobe AIR Selector for Firefox, IE and Safari browsers
- No support for mobile devices yet. Only being added in Higgins V2

# OpenID

- A scheme in which users choose their own URI and the password to authenticate to it at any OpenID service provider, such as:
  - AOL - `openid.aol.com/screenname`
  - LiveDoor - `profile.livedoor.com/username`
  - LiveJournal - `username.livejournal.com`
  - SmugMug - `username.smugmug.com`
  - MyOpenID - `Bill.Gates.12000.myopenid.com` (guess who this belongs to!)
- Users can then SSO to other sites via their Open ID service provider
- Since there is no authentication of the identity chosen by the user, OpenIDs are typically only used on blogs and other non-commercial sites that do not need to know the real identity of the user

For further details see: <http://openid.net/>

# OpenID Mode of Operation

1. User goes to a Service Provider and is shown a login screen with a place for the OpenID of the user.
2. User enters her OpenID e.g. [www.chadwick.com](http://www.chadwick.com)
3. SP turns this into a URL and fetches the web page from there. This page tells the SP which OpenID Provider to use, and what name to verify there, through two links in the HEAD section of the web page
4. The SP contacts the OpenIDP via a http redirect
5. The user authenticates to the OpenIDP, and the provider stores a cookie on the user's web browser for SSO next time around
6. The user is redirected back to the SP with a token saying she has authenticated correctly
6. If the OpenIDP and SP already share a secret then authn is finished as the token can be validated using the shared secret
7. Otherwise SP must validate the token by sending it directly back to the OpenIDP asking if it is a correct one or not.

NOTE that OpenID requires trust between the SP and OpenIDP

FOSAD 29-30 Aug 08

© 2008 David Chadwick

44

The mandatory link in the HEAD section of the web page at the user's Open ID contains a pointer to the server of the OpenID provider e.g.

```
<link rel="openid.server" href="http://www.openid.com/server.bml">
```

The user would then need to authenticate to this server with the OpenID that they gave to the SP. If the user wants to use a different OpenID, say the OpenID of his home page, e.g. [www.chadwick.com](http://www.chadwick.com), but he does not run an OpenID Provider, then he can delegate authentication to one, by registering with it. Say the user registers the OpenID <http://exampleuser.livejournal.com/> with the OpenID Provider located at <http://www.livejournal.com/openid/server.bml>, then he will add the following two tags to the HEAD section of the HTML document located at his web page <http://www.chadwick.com>.

```
<link rel="openid.server"
href="http://www.livejournal.com/openid/server.bml">
```

```
<link rel="openid.delegate" href="http://exampleuser.livejournal.com/">
```

Now, when the SP sees these, it'll talk to <http://www.livejournal.com/openid/server.bml> and ask if the user is [exampleuser.livejournal.com](http://exampleuser.livejournal.com/), never mentioning [www.chadwick.com](http://www.chadwick.com) anywhere on the wire.

The main advantage of this delegation mechanism is that a user can keep their Identifier over many years, even as services come and go; they'll just keep changing who they delegate to. The user has proved they own the Identifier, because they have the rights to update their web page and put the appropriate links in it.

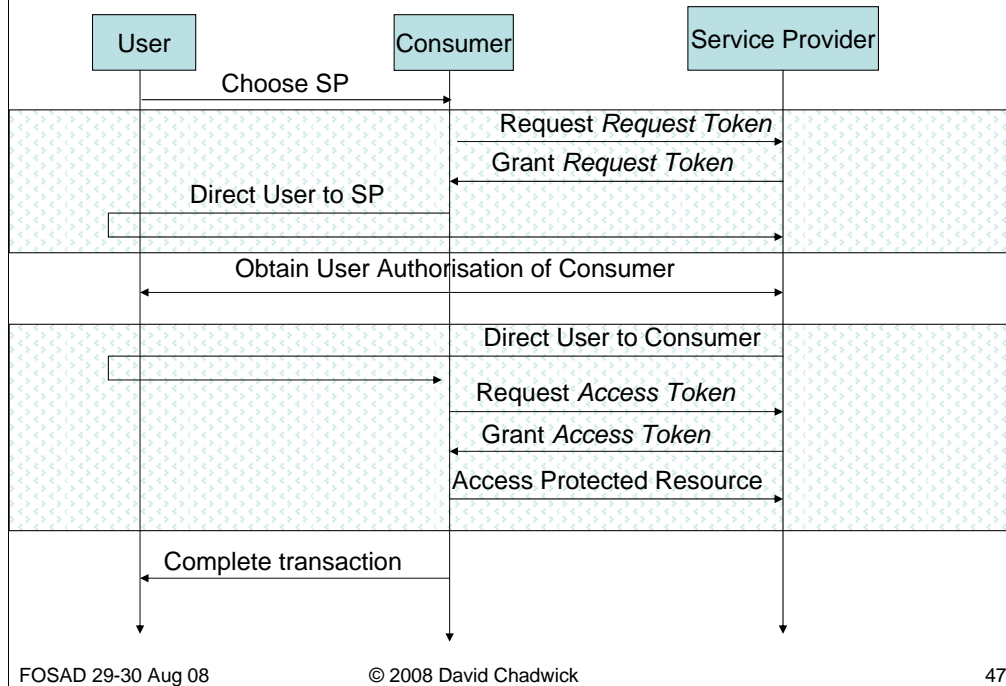
# Failings in OpenID v1

- No authentication of the identity of the user, so OpenID only provides some assurance that it is the same person as last time. But who is that person?
  - Therefore OpenID can only be used by Service Providers who don't care who the user is, but only that it is the same user each time
- User's OpenID can be re-assigned to another person if they cease relationship with OpenID provider
  - OpenID v2 proposes to address this via assigning a globally unique number as the real OpenID using XRI and XRDS
  - XRI (Extensible Resource Identifiers) are to URIs what domain names are to IP addresses
  - XRDS (Extensible Resource Descriptor Sequence) is to XRI what RRs are to the DNS
- Susceptible to phishing attacks. Evil SP can masquerade as user's OpenID IdP and capture user's credentials

# OAuth

- Designed to let one site (the consumer) access a user's attributes/private data held at another site (the service provider) if the user authorises this
  - E.g. a user wants his photos stored on Flickr to be printed out by a commercial printer
  - E.g. a user is applying for a job and wants the potential employer to access his degree transcript held at his university
- Designed to replace existing consumer site practices that today ask users for their un/pws at the SP site
  - No federation, consumer masquerades as user
- OAuth requires the consumer to have a password with the SP, so that the SP can authenticate the consumer

# OAuth Protocol Flow



## Advantages of OAuth

- A very simple spec and easy to implement
- Security is sufficient for many low value transactions

## Limitations of OAuth

- One way authentication of consumer to SP and
- Uses passwords rather than public keys
- Still contains a lot of proprietary/non-standard features in the protocol “any additional parameters as defined by the Service Provider”
- Does not support user SSO, but can be combined with other protocols that do e.g. OpenID or Shibboleth



## Limitations of Current FIM Systems

- Cannot use credentials from multiple AAs in the same user session (attribute aggregation)
- With Cardspace, cannot copy cards to other devices
- Shibboleth and OpenID are open to phishing attacks