Federated Identity Management Technologies and Systems

David Chadwick

Some Early FIM Systems

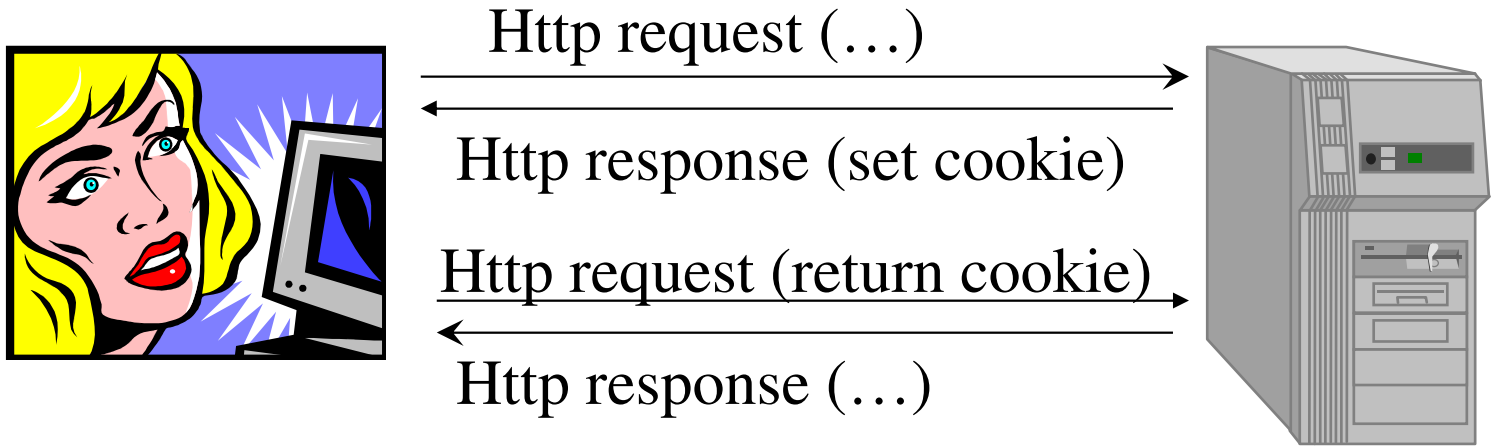
- Microsoft's Passport
- UK Athens

Some More Recent FIM Systems

- Shibboleth
- Liberty Alliance
- CardSpace and AD FS 2
- OpenID
- OAuth
- UMA

Single Sign On (SSO) using HTTP Cookies

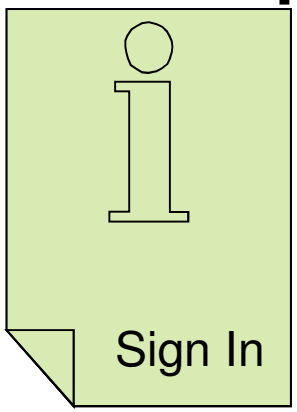
- Cookies – allow a web server/site to store state information for itself (often encrypted) on the user's browser
- A web site can store many cookies, and the client should return them all when it returns to the site
- Often used to enable SSO, since the authenticating site can tell if a user is already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



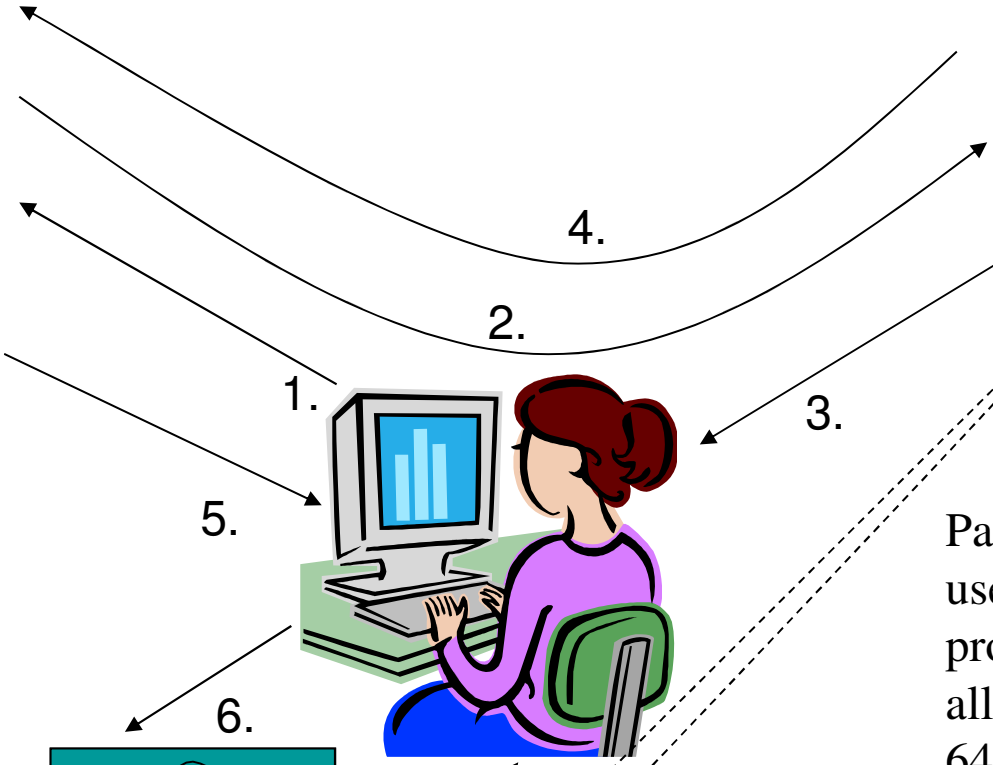
Microsoft's .NET Passport

- .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.

Passport Protocol Flows



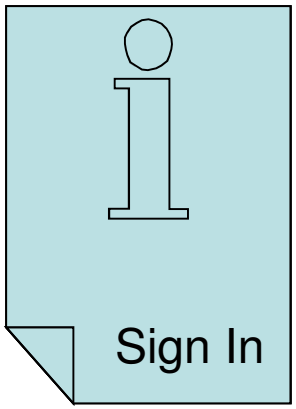
Participating Site A



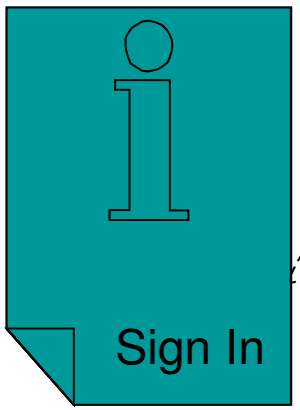
Passport Site

Passport site stores the user's credential and profile information, and allocates the user a unique 64 bit Passport User ID

(PUID)



Participating Site B



Participating Site C

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

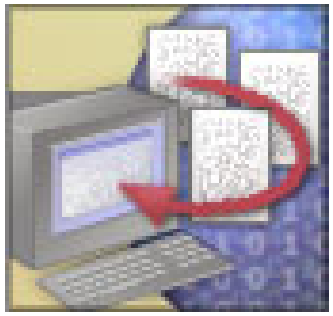
All referenced by a Passport Unique ID (PUID)

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

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)

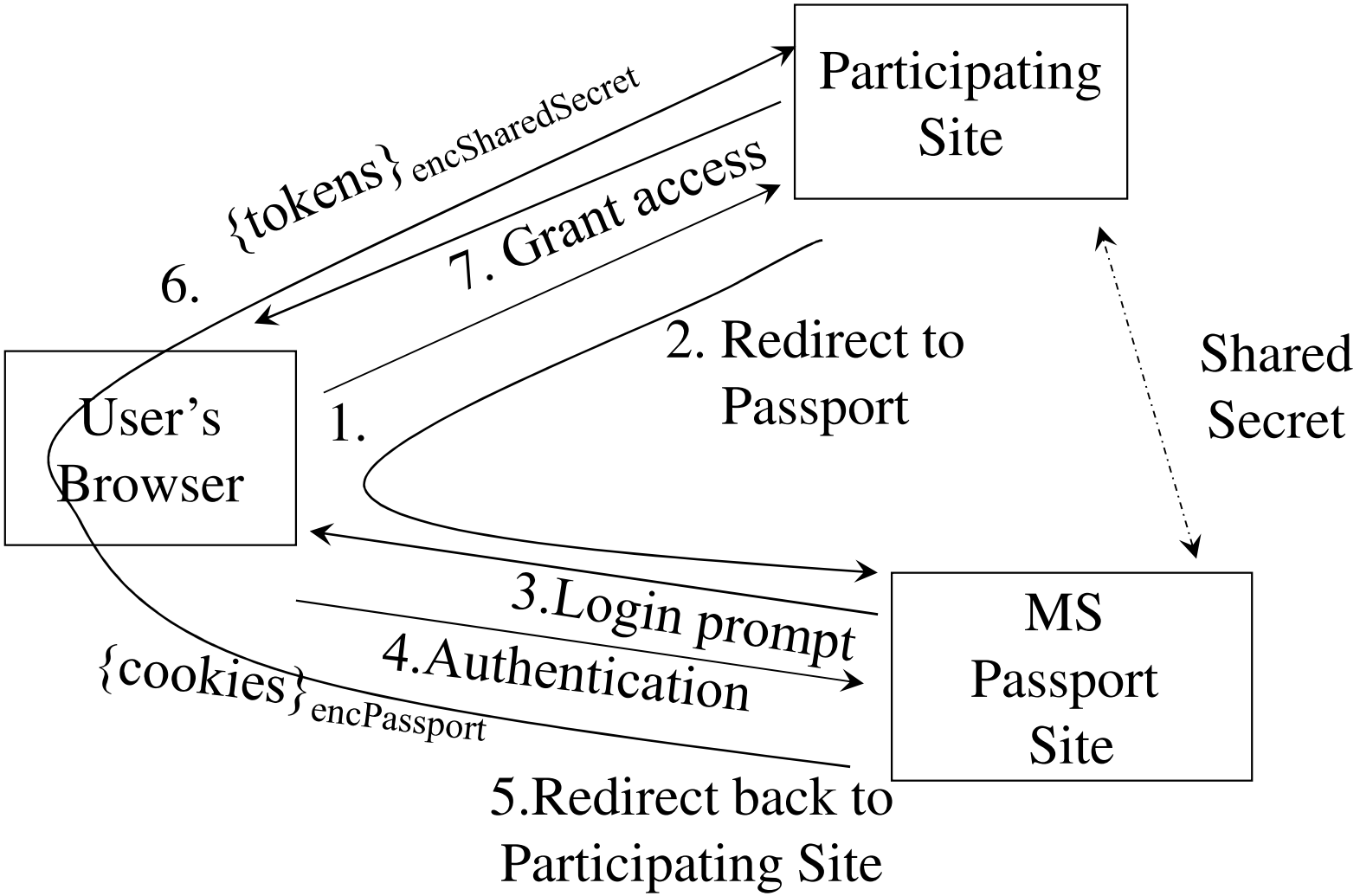


Tired of registration forms? You can speed registration and get personalized services at participating sites by sharing your .NET Passport information with them when you sign in. Select the boxes below to choose how much of your .NET Passport information Microsoft can share with other companies' .NET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

Authentication with .NET Passport



Passport Cookies and Tokens

- Follows the Kerberos model
- Authentication Ticket and Profile Information of the user are encrypted with key shared between Passport and Participating Site
- Ticket granting cookie holds a symmetric key used to encrypt the other cookies, encrypted with secret key known only to Passport
- Authentication cookie holds the SSO information
- Profile cookie holds the user's profile
- Participating Sites cookie 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 and is used for single logout to clean up 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 3rd 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.
- Also partially fails first law because consent is not fine grained enough
- Finally the Kerberos-like protocol used by Passport was not standardised, so other organisations might be wary of using it
- Passport has now been superceded by Windows Live ID, which is an identity meta-system that provides support for Passport, CardSpace and OpenID

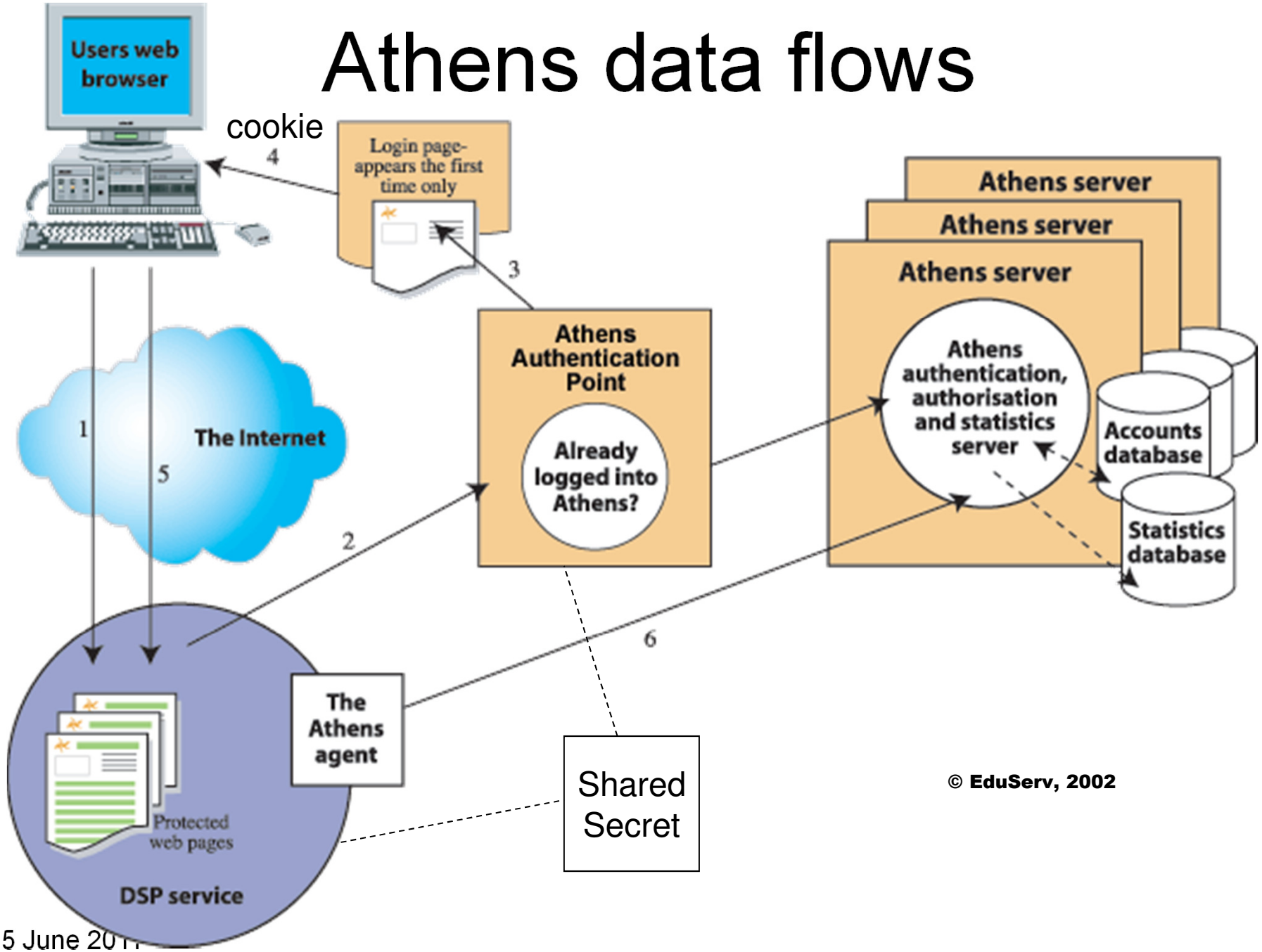
Athens

- 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

How does it work?

- Originally a “trusted third party” network service similar to Passport
  - 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

Athens data flows



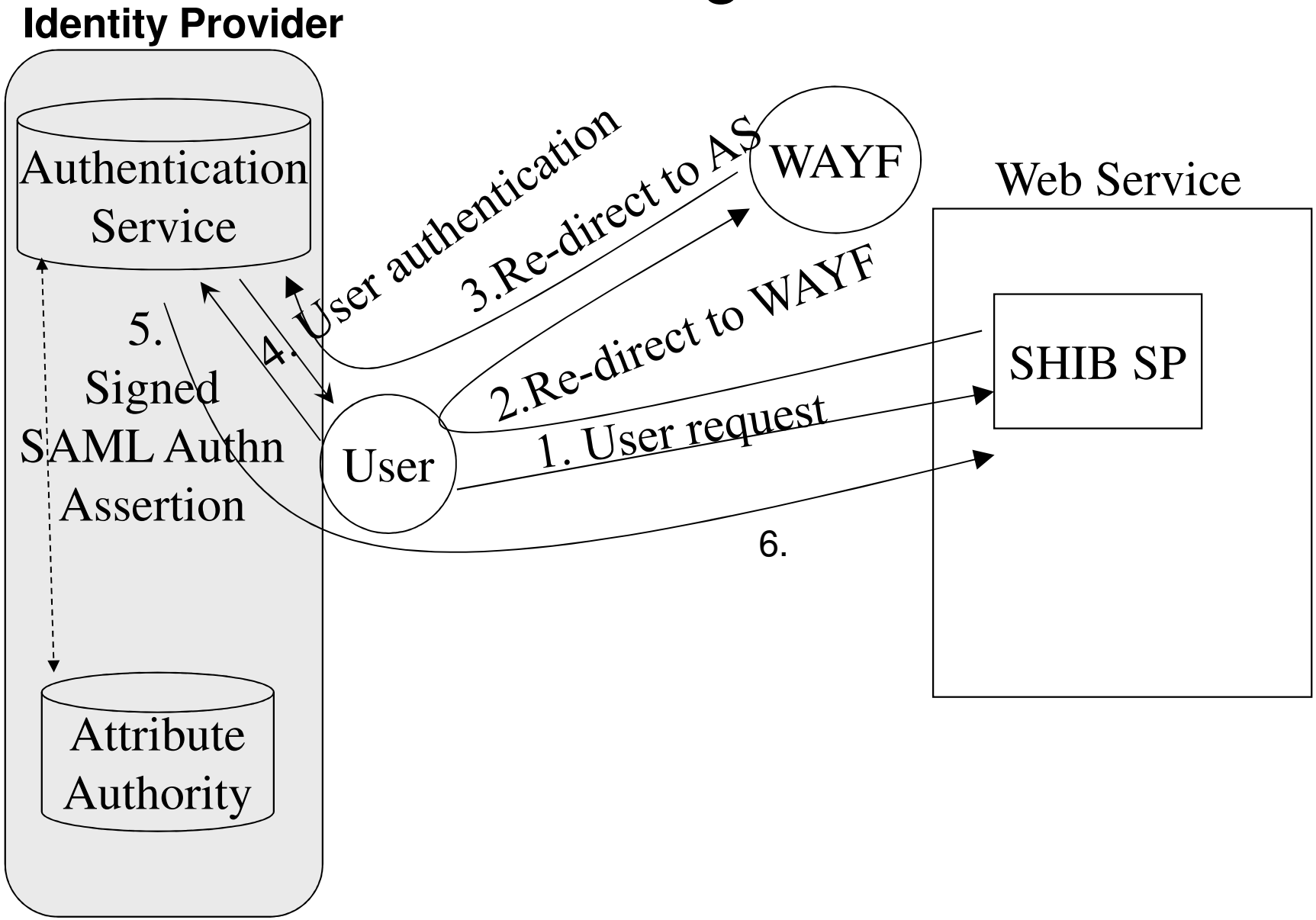
Why did Athens Fail ?

- 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/SAMLv2 which provides similar functionality but is standards based and uses open source software

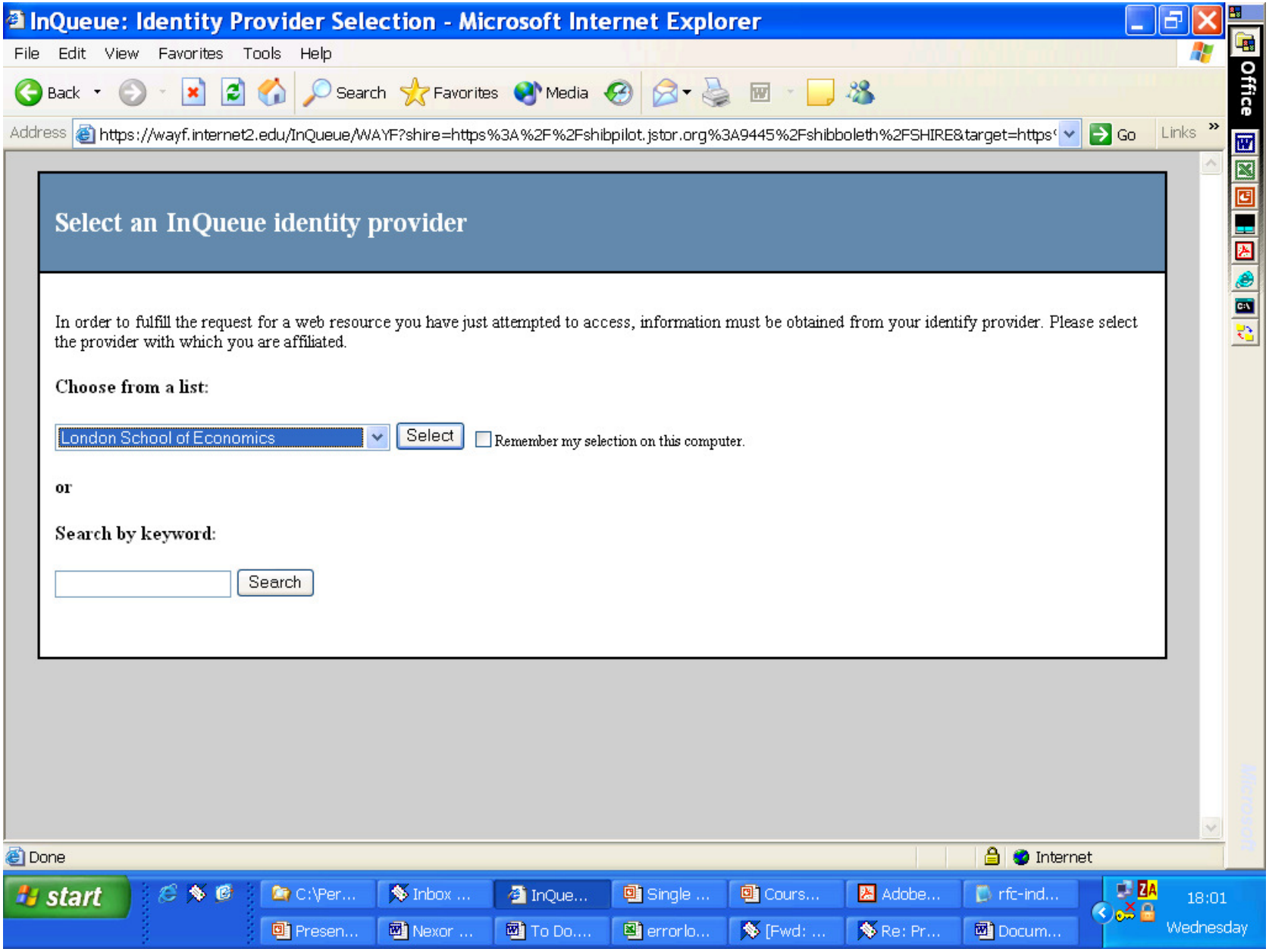
Shibboleth

- Shibboleth is a project run by the Internet2 consortium from the USA (which has over 350 academic and commercial partners)
- Shibboleth uses the OASIS SAML 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 v1 access took place in two stages but these can be combined into one exchange in v2
  - i) Obtaining an authentication assertion for a user from his home site (IdP)
  - Ii) Using this to get a set of attribute assertions for the user
  - The two messages can be combined into one exchange using SAMLv2 to make the protocol more efficient
- UK academic federation has 800 services (IdPs and SPs) interconnected with Shibboleth

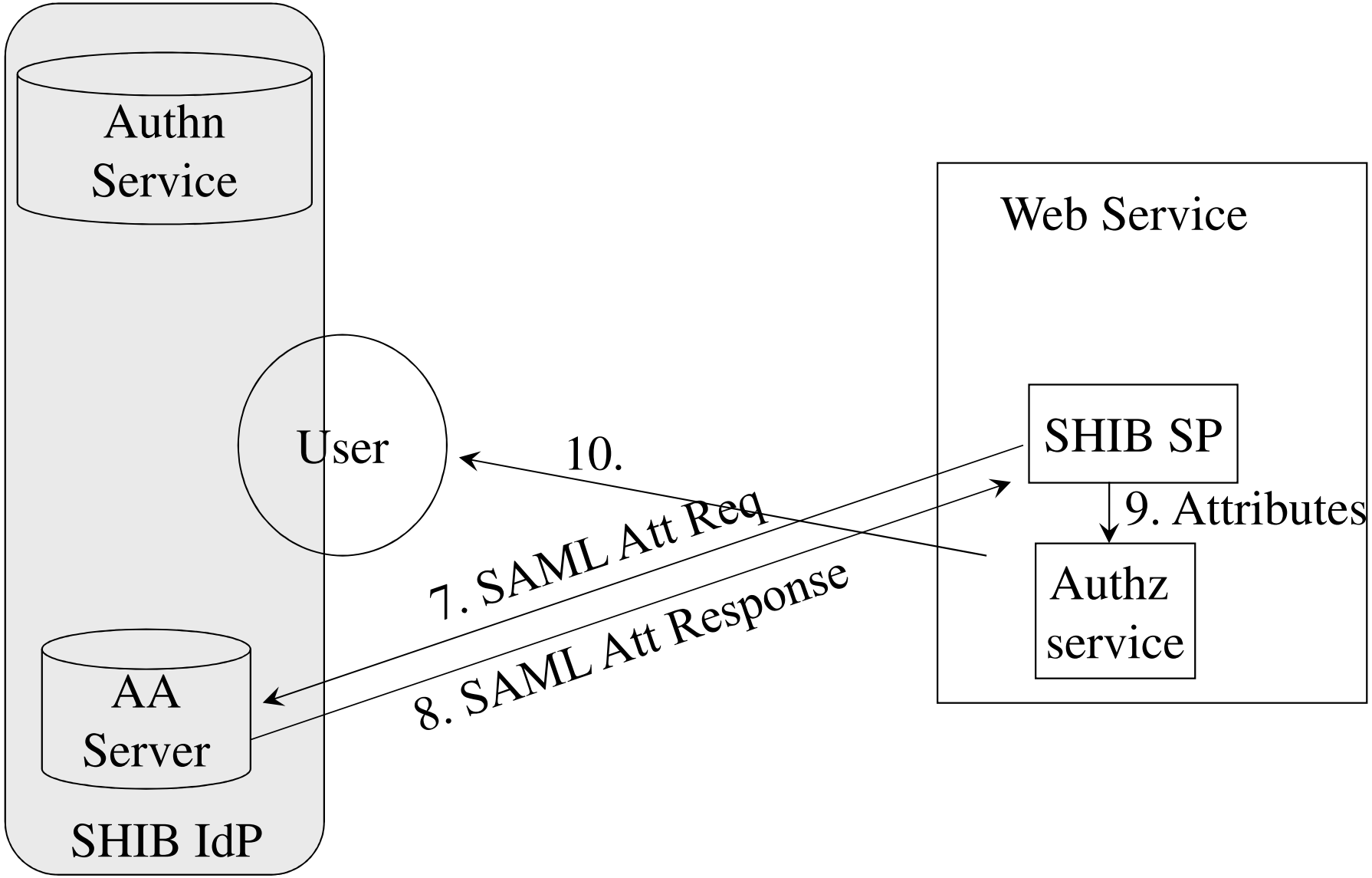
Shibboleth – Obtaining User Authentication



The WAYF Service



Shibboleth – Obtaining Authz Attributes



Privacy Protection in Shibboleth

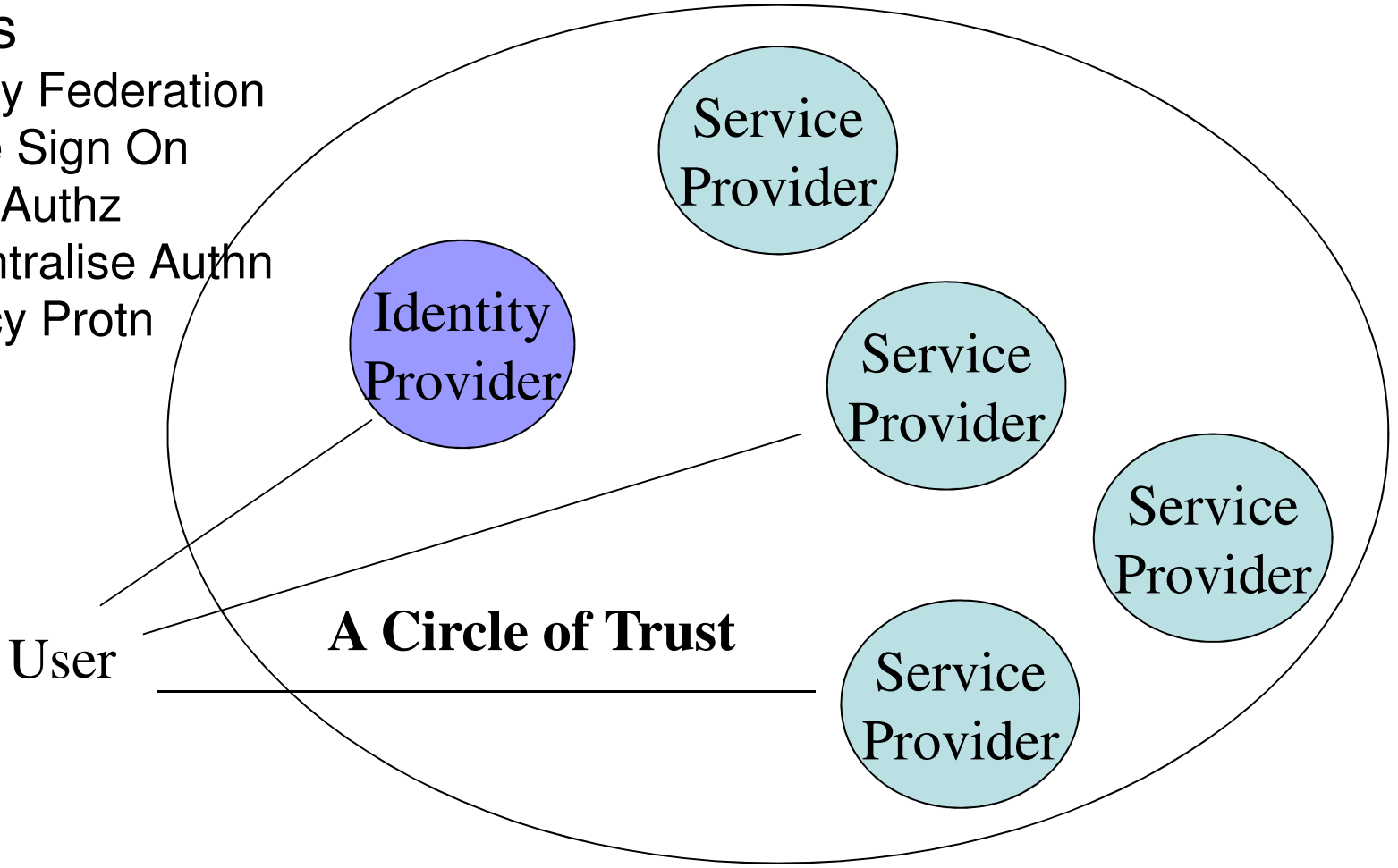
- Privacy Protection is an important feature and strength of Shibboleth
- User identifiers remain private to the home site. Users are only identified via a transient identifier, which changes each time the user accesses a site
- Users are authorised to access resources based on their identity attributes rather than their identifiers e.g. 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
- Users do not give consent and don't know which attributes are being sent
- 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 (even though SSL is used, user's typically don't look at the certificates)
- Does not provide Single Sign Off (yet)
- Uses bearer credentials i.e. do not contain a token so that user can prove they belong to him. This means they can be stolen from a browser and used by an imposter.
- 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

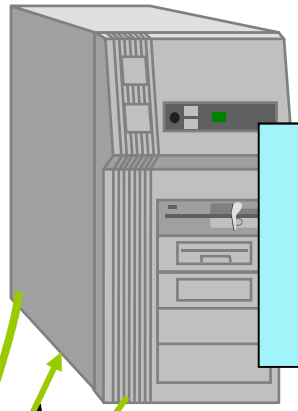
Liberty Alliance

- 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
  - Privacy Protn



Identity Federation with LA ID-WSF

BankX IdP



After federation BankX holds
A/c = 12345
PId_y out = xA3c15gK
Provider=Insurance Y
PId_y in = lo1J41aT

and InsuranceY holds
Policy = 2456
PId_x out = lo1J41aT
Provider =BankX
PId_x in = xA3c15gK

1. User logs onto Identity service provider BankX (as a/c 12345)

2. User is asked if she wants to federate her identity with other service providers

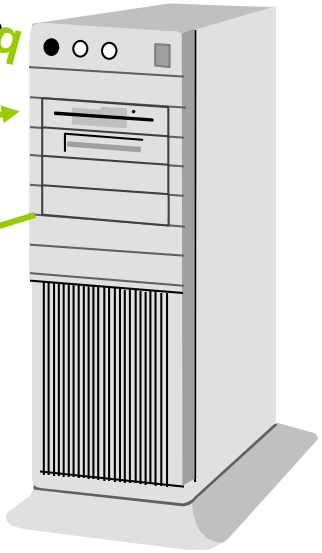
7. SAML Manage NameID req with xA3c15gK

6. InsuranceY sends its PId lo1J41aT To BankX in SAML response

4. User logs onto InsuranceY (as Policy 2456)

3. User chooses InsuranceY and is redirected there with SAML req

5. User may be asked to confirm if she wants to federate her identity to BankX's

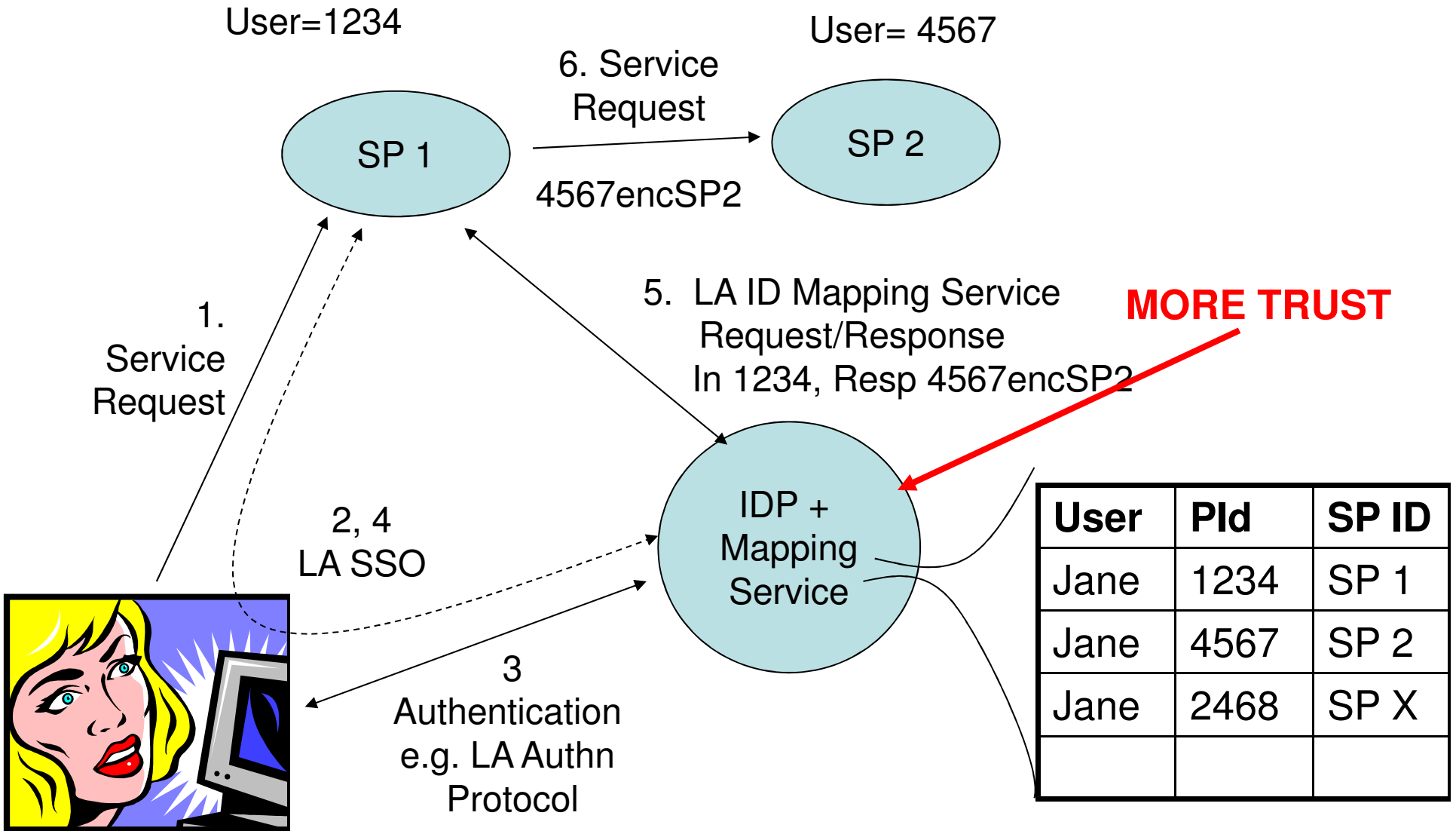


InsuranceY IdP

Privacy Protection in LA

- A strong privacy protecting feature of the Liberty Alliance ID-WSF protocols is pairwise Persistent Identifiers (PId) between each IdP and SP
- Each user has a different PId with each SP
- Therefore no correlation handles exist since each SP knows the same user by a different PId
- Downside is this significantly complicates the protocol exchanges
- Need an Identity Mapping service built into each IdP

Liberty Alliance ID Mapping Service



User	PId	SP ID
Jane	1234	SP 1
Jane	4567	SP 2
Jane	2468	SP X

ID Mapping Contents

- ID Mapping Request contains one or more of
  - Request ID (optional)
  - Policy, describing the security token to be returned e.g. enc PId For SP 2
  - Security Token to be exchanged e.g. Authn Statement for PId 1234
- ID Mapping Response contains
  - Status i.e. OK, partial or failed
  - For each successfully processed request element
  - Request ID (optional)
  - Requested Security Token e.g. Authn Statement for PId 4567 encrypted for SP 2

Liberty Alliance Discovery Service

- How do we find the services related to an identity?
  - E.g. a user's credit card service, a user's calendar service etc.
- Ans. Each user has a Discovery Service that SPs can query to find out about other services of the user
- The Disco Service is closely linked to the user's IdP (usually part of it)

Discovery Service Construction

- A service type, named by a urn, is described by <wsdl:types>, <wsdl:message>, and <wsdl:portType>
- A service instance is a deployed physical instantiation of a particular service type, is hosted by some provider and is identified by a URI.
- A service provider may deploy one or more service instances when deploying a service (each instance is described by <wsdl:binding>, <wsdl:service>, and <wsdl:port>)
- For an identity service, the service instance *resource* is either *data related to some user's identity* or *a service acting on behalf of some user*.
- When a client sends a request message to an identity service instance (identified by a LA EndPointReference - see later) information in the message serves to implicitly identify the *resource* being acted upon.
- The LA Discovery Service facilitates the discovery and invocation of these service instances

WS Addressing EPR

- Address – URI of the service instance (actually IRI – internationalised URI)
- Reference Parameters – associated with the endpoint to facilitate a particular interaction
- Metadata – describes the behaviour, policies and capabilities of the endpoint

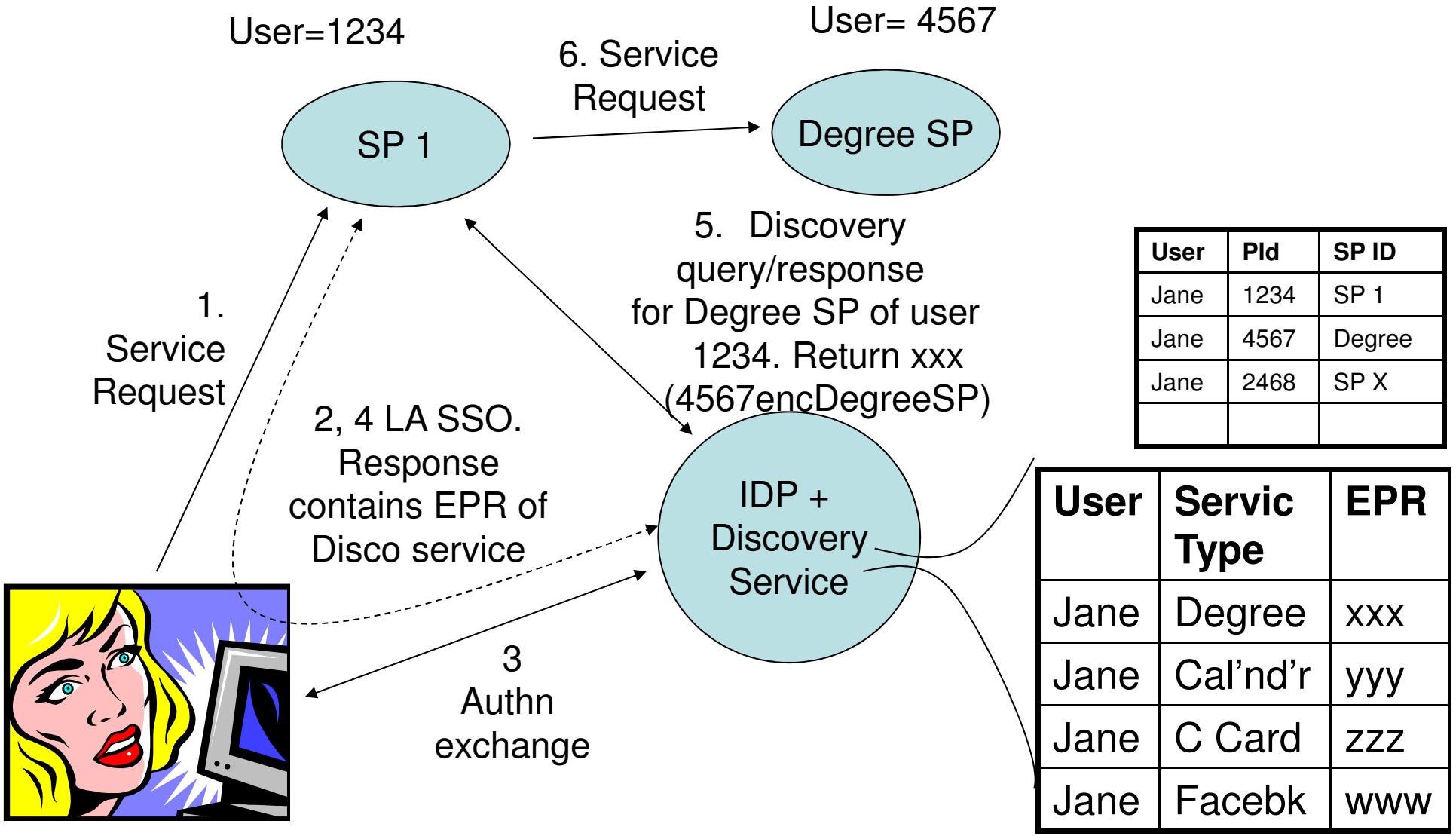
ID-WSF EPR Contents

- Some new attributes of the EPR
  - the ID of the provider of the service instance
  - notOnOrAfter – the expiry date/time of the EPR
- Some new elements inserted in the Metadata field
  - Info about the service type and various actions and options supported
  - Security information containing the security and identity tokens needed by the endpoint and the security mechanism to be used to connect to the endpoint

Disco Service Interaction Model

- Service instances are identified via a Liberty profile of WS-Addressing End Point References, known as ID-WSF EPRs
- A user's user agent contacts a SP, is authenticated by an IdP using LA protocols and the SP is sent a discovery bootstrap assertion (along with the authn assertion) which contains a pointer to the user's Discovery Service instance (as an ID-WSF EPR)
- The SP, acting as a web service consumer (WSC) uses the ID-WSF EPR to query the user's Discovery Service for a pointer to some other desired service(s) of the user
- The Discovery Service returns one or more ID-WSF EPRs that point to the user's service instance(s) of the requested type(s).
- The WSC (SP) now employs the returned ID-WSF EPR(s) to interact with the identified service instance(s), which themselves will be acting in the role of a web service provider(s). The WSC (SP) returns the results to the user's user agent.

Discovery Service



User	PId	SP ID
Jane	1234	SP 1
Jane	4567	Degree
Jane	2468	SP X

User	Servic Type	EPR
Jane	Degree	xxx
Jane	Cal'nd'r	yyy
Jane	C Card	zzz
Jane	Facebk	www

Disco Service from the SP's viewpoint

- A SP e.g. university, may register its generic Service Metadata (inc. service type) with a Discovery Service
- A service instance, acting as a WSP, is then deployed at some addressable endpoint and is providing some service on behalf of a user, e.g. issues her degree transcript (the university can have thousands of these)
- It asks the user if this service should be discoverable
- If so, the WSP registers itself with the Discovery Service by supplementing the generic Service Metadata in the DS with the user details.
- The Service Metadata now describes the WSP's service instance so that the Discovery Service can mint new ID-WSF EPRs on demand for any WSC that wishes to invoke that WSP

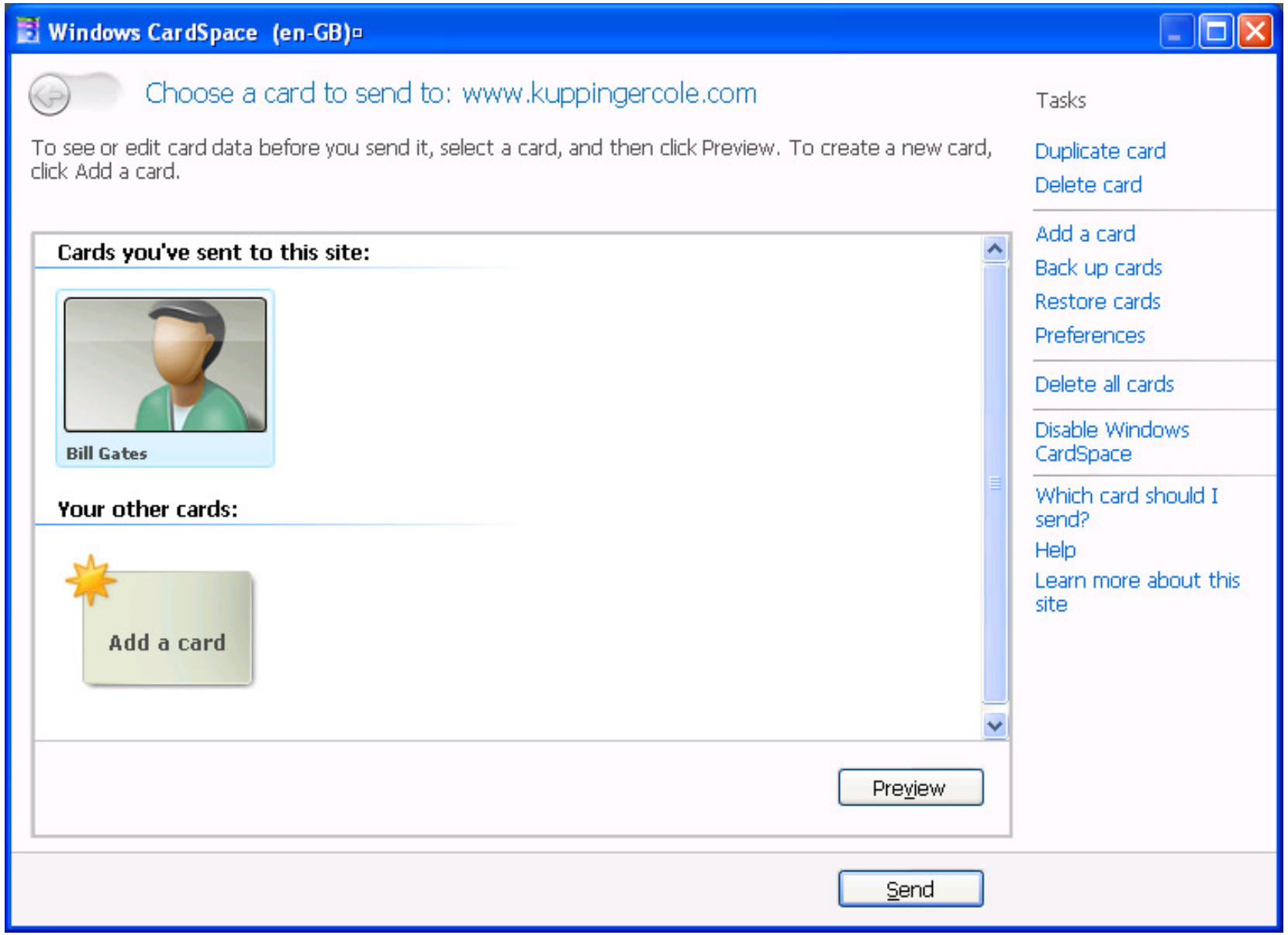
Weakness in LA Design

- Complexity. Liberty Alliance is built using a different PId for the same user at each SP (Kim Cameron's unidirectional identifiers) to stop SPs correlating their user accounts
- However, in order for SPs to pass a user's request between themselves, the IdP has to support the WSF mapping service
- Unscrupulous SPs can ask for the encrypted PId of a user at another SP (in fact at every discoverable SP), can log this, pass it to the other SP, who also logs this, then the SPs can compare their logs and correlate the user's accounts. So it does not stop collusion.
- Furthermore if a user provides globally unique attributes to his SPs, such as his telephone number or email address, then the SPs can still correlate the user's accounts

Microsoft's CardSpace

- An Identity Management system from MS which is part of Windows 7, Vista and can be added to XP
- User has a set of InfoCards held in his Identity Selector – a secured desktop application
- Each card represent the user's profile provided by either
- An Identity Provider (IDP) – *managed* cards
- The user him/her self – *self issued* cards
- User can add new cards by either browsing his filestore for managed cards or creating his own self issued cards

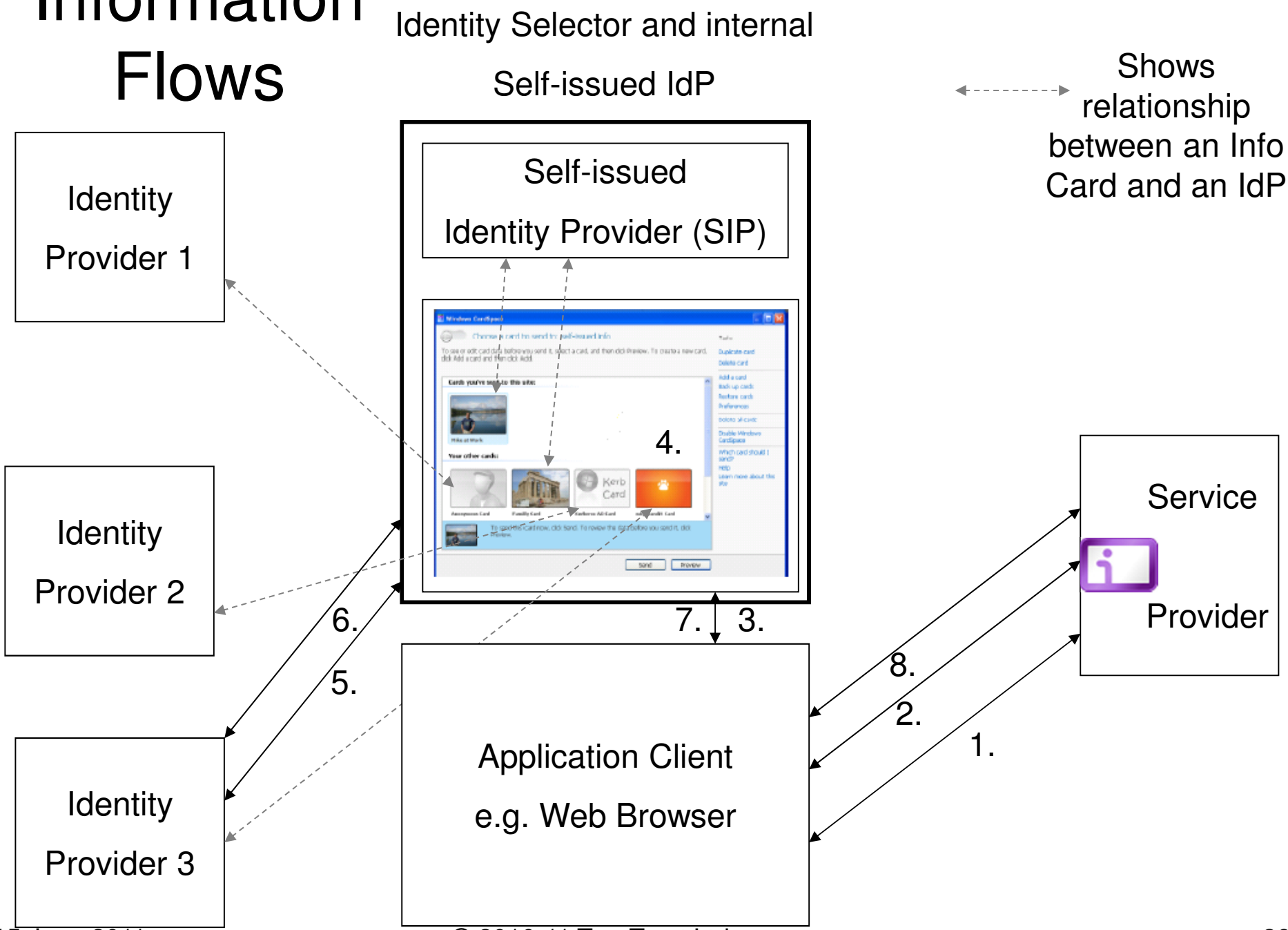
Example Identity Selector Screen



Accessing a Site

- User goes to SP site and clicks on the CardSpace icon 
- SP sends its requirements for authentication/authorisation (security policy) back to the user's web browser, which gives it to the Identity Selector
- The InfoCards that fulfil the SP's policy are lit up in the Identity Selector, ones that dont are greyed out. Ones sent before are in the top half of the display, others are in the bottom half.
- User clicks on the card she wants to send to the SP
- If a managed card, then Identity Selector asks the user to authenticate to the IDP using whatever scheme the IDP requires (un/pw, Kerberos, X.509 cert), and the Identity Selector goes to the IDP site, picks up the sites policy (see later) then requests a signed attribute assertion. It passes the assertion back to the browser which POSTs it to the SP.

Information Flows



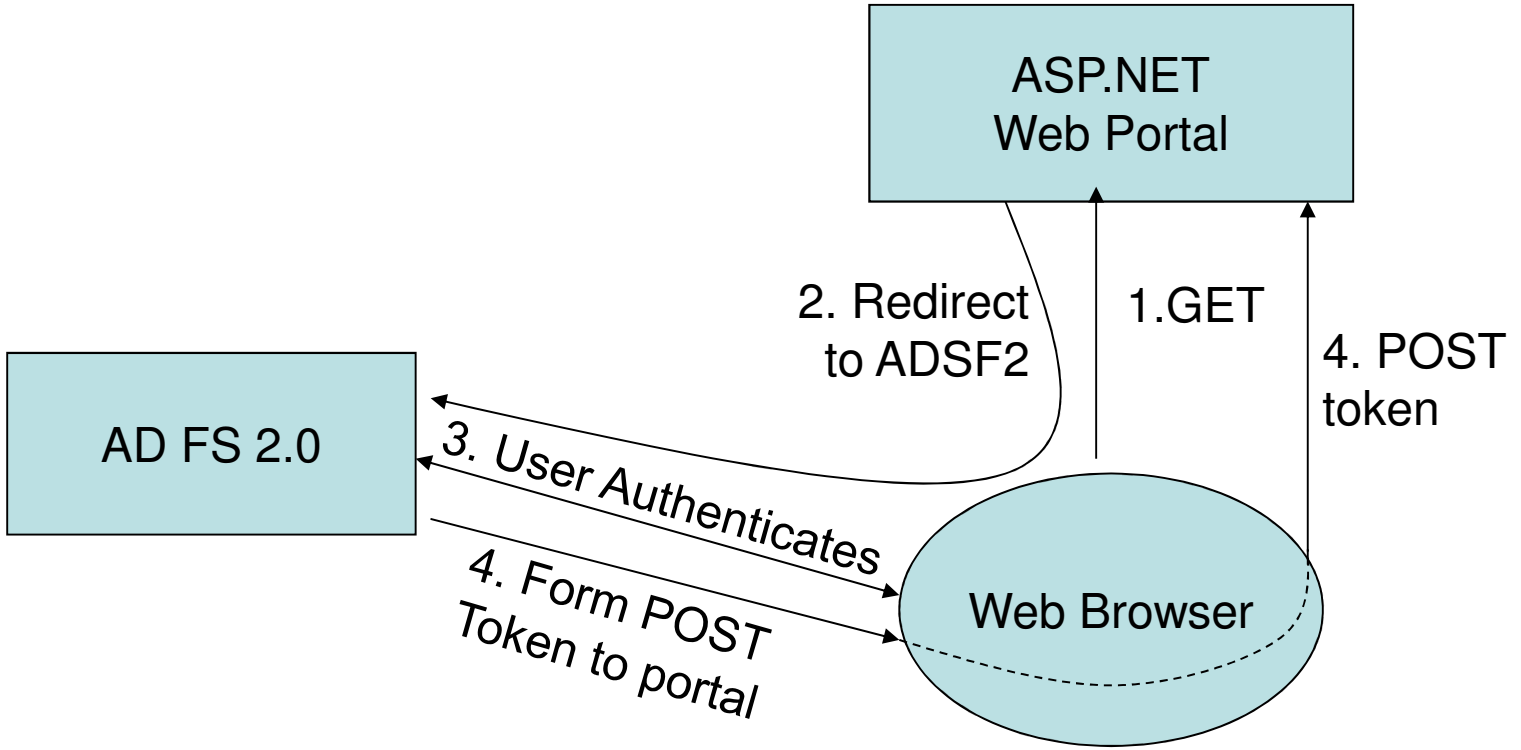
Limitations of CardSpace

- Potentially nice user experience/metaphor, but…user trials found it was too difficult to use
- Only a single infocard (and hence single security token) can be sent to an SP
  - cant send student union card + credit card + postage details to an SP to get a student discount on purchasing a book
- User has to be involved to select a card, so it cannot be used for delegation between services
- Originally limited to latest Windows PCs
- Microsoft discontinued CardSpace 1Q 2011

Microsoft Active Directory Federation Services 2.0

- Originally code named Geneva server
- It is an IdP/STS that can issue credentials/claims/tokens in a variety of formats
  - SAML 1.1 or SAML 2 assertions
  - Wrapped in either WS-Trust, WS-Federation or SAML 2 protocol messages
- Uses a backend Active Directory server as the identity store. Can also support other attribute stores such as SQL DBs and LDAP servers

AD FS 2 Protocol Flows

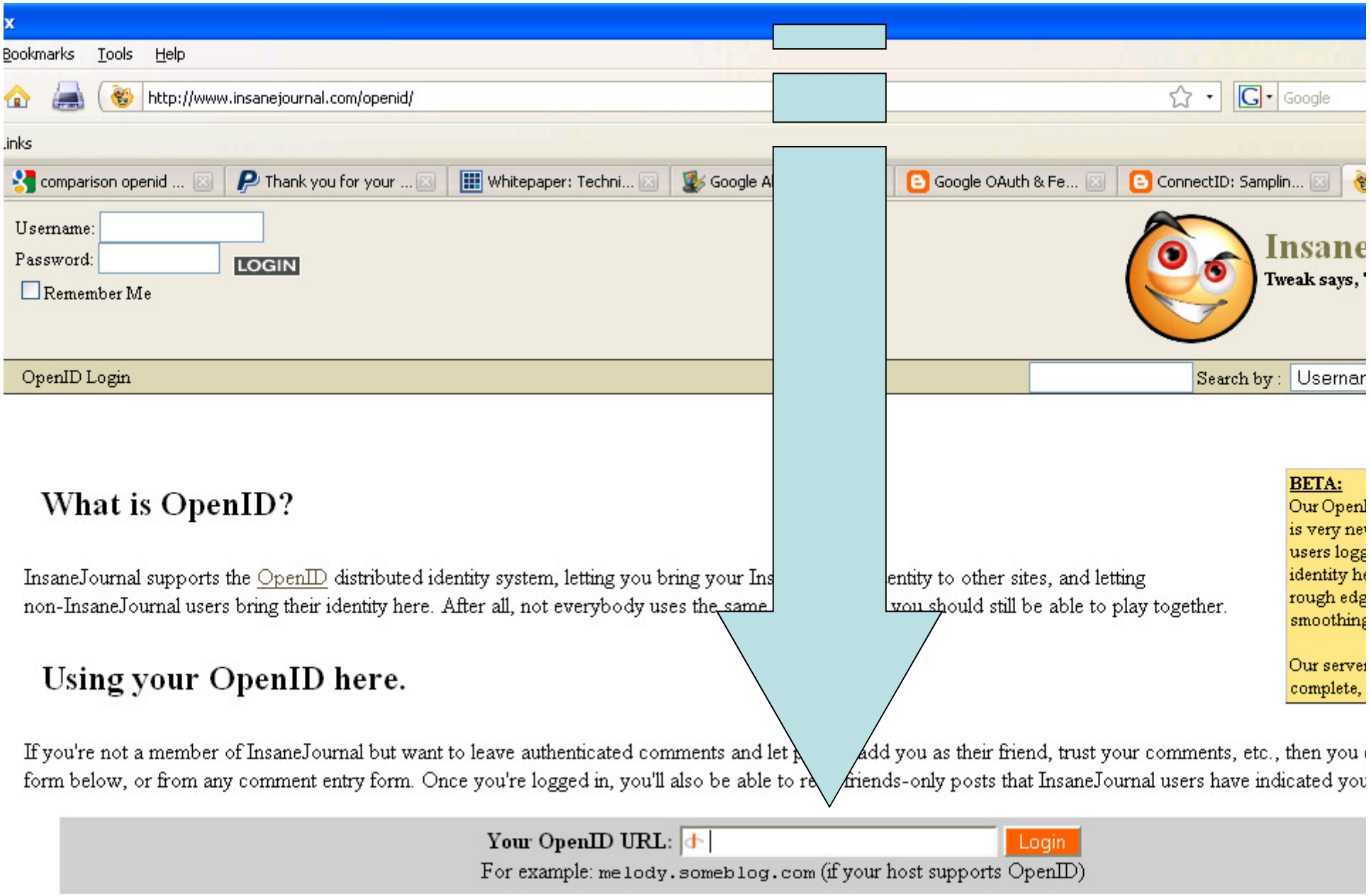


- More or less interworks with Shibboleth/SAML2 but there are some issues documented here:
- https://spaces.internet2.edu/display/SHIB2/MicrosoftInterop

OpenID

- A scheme in which users choose their own URI and password to authenticate 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 (my openID)
- 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 i.e. LoA = 1

An Example OpenID Relying Party



OpenIDv2 Mode of Operation

1. User goes to a Service Provider (called Relying Party) 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
3. SP/RP performs Discovery on the presented OpenID to find the URL of the OpenID IdP that will perform authentication of the user

Discovering the OpenID IdP

- The SP/RP has three choices:
- If the OpenID is an XRI, it performs XRI Resolution. This should return an XRD document that contains the necessary information
  - XRIs are to URIs what domain names are to IP addresses
  - However XRI standardisation has stalled within OASIS due to technical objections from W3C
- If the OpenID is a URL, the Yadis protocol must be attempted first. If this succeeds, the result is again an XRD document, otherwise
- HTML-Based discovery is attempted on the URL
  - an HTML document must be available at this URL and the HEAD element must contain a LINK element that points to the OpenID provider (attribute "rel" set to "openid2.provider" and "href" set to the OpenID IdP's URL). An optional LINK element may be included that contains an alternative OpenID for the user.
  - This is the most common choice today

YADIS Protocol

- YADIS attempts to answer the question “Given an identity URL, how do I know what protocol needs to be used to authenticate that user?”
- YADIS is a service discovery system that allows relying parties (RPs) to automatically determine the most appropriate protocol to use
- When a user enters a URL into an authentication interface, the RP needs to discover which services are available using that URL
- The RP makes an HTTP request to the URL and is returned either an XRD or a set of URLs that locate various XRDs
- Parsing an XRD provides the services and protocols that should be used, such as the protocol to use for authentication
- Consequently the owner of the original URL can choose which services he wishes to use in his XRD document

OpenIDv2 Operation (cont)

3. SP obtains (from URL or XRI) which OpenID Provider to use, and what name to verify there
4. The SP optionally established a secret with the OpenIDP using Diffie Hellman protocol (or a secret may already be pre-established)
5. The SP redirects the user to the OpenID Provider with an Authentication Request message
6. The user authenticates to the OpenIDP, and the provider may store a cookie on the user's web browser for SSO next time around
7. The user is redirected back to the SP with a token saying she has authenticated correctly
8. If the OpenIDP and SP already share a secret then authn is finished as the token can be validated using the shared secret
9. Otherwise the SP must validate the token by sending it directly back to the OpenIDP asking if it is a correct one or not.

Failings in OpenID v1 and v2

- 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 addresses this by using the XRI globally unique number as the real OpenID, and the RP stores this
    - XRI (Extensible Resource Identifiers) can comprise a globally unique non-re-assignable number and a re-assignable name.
- Susceptible to phishing attacks. Evil SP can masquerade as user's OpenID IdP and capture user's credentials
- Requires secrets to be shared between OpenIDP and SP so requires pre-established direct trust relationships
- But is used by Google's single sign on service

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/delegates this
  - E.g. a user wants his photos stored on Flickr to be printed out by a commercial printer. User must delegate access to 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. User delegates access to potential employer
- Designed to replace existing consumer site practice of asking users for their un/pws at the SP site
  - i.e. No federation, consumer masquerades as user
- OAuth requires the consumer to have a un/password with the SP, so that the SP can authenticate the consumer

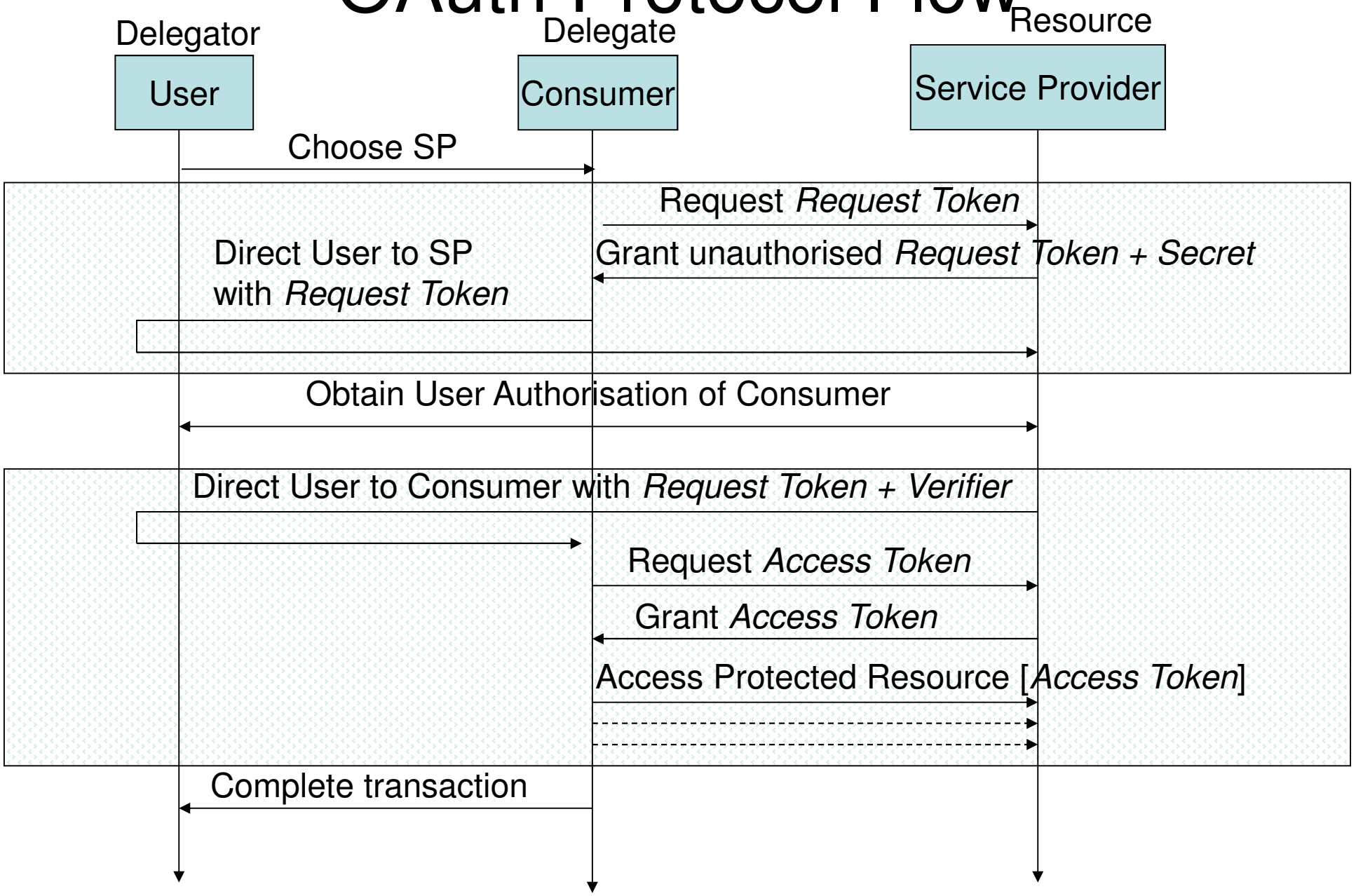
OAuth v1 Publications

- Originally published by OAuth Core Workgroup as "OAuth Core 1.0 Revision A", 24 June 2009. Available from http://oauth.net/core/1.0a
- Recently published as Informational RFC 5849, April 2010. But with changed terminology and stricter security requirements (e.g. SSL/TLS)
  - Consumer → client
  - Service Provider → server
  - User → resource owner
  - Consumer Key and Secret → client credentials
  - Request Token and Secret → temporary credentials
  - Access Token and Secret → token credentials

Mode of Operation

- Client/Consumer asks the SP for a temporary un/pw for this delegation (called the Request Token and Secret)
  - This allows the user to cancel this delegation midway through, without the SP needing to revoke the client's long term un/pw
- Client sends user to SP, who authenticates and delegates permission
- SP returns the (authorised) Request Token plus Verifier to client via user
  - Since Verifier is sent via the user, it will be a relatively short secret and hence weak
- Client uses temp un/pw/verifier to directly request an access token + secret from SP which is returned via this back channel (so user never sees it)
- Access token (un/pw) allows client to access user's resource whenever it wishes
- User is able to cancel this delegation up until the point the client has accessed the resource

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 so evil SP could send false info to client, and also capture user's un/pw
- Uses passwords rather than private/public keys so very weak unless long passwords are chosen. Also requires passwords to be held in the clear so can create secure hashes of messages
- Provides message integrity with secure hashes but no guarantee of request confidentiality
- Does not provide fine grained delegation (its all or nothing). Requires SP to provide this itself
- Still contains a lot of proprietary/non-standard features in the protocol "any additional parameters as defined by the Service Provider"
- For this reason the IETF OAuth WG is now producing OAuthv2

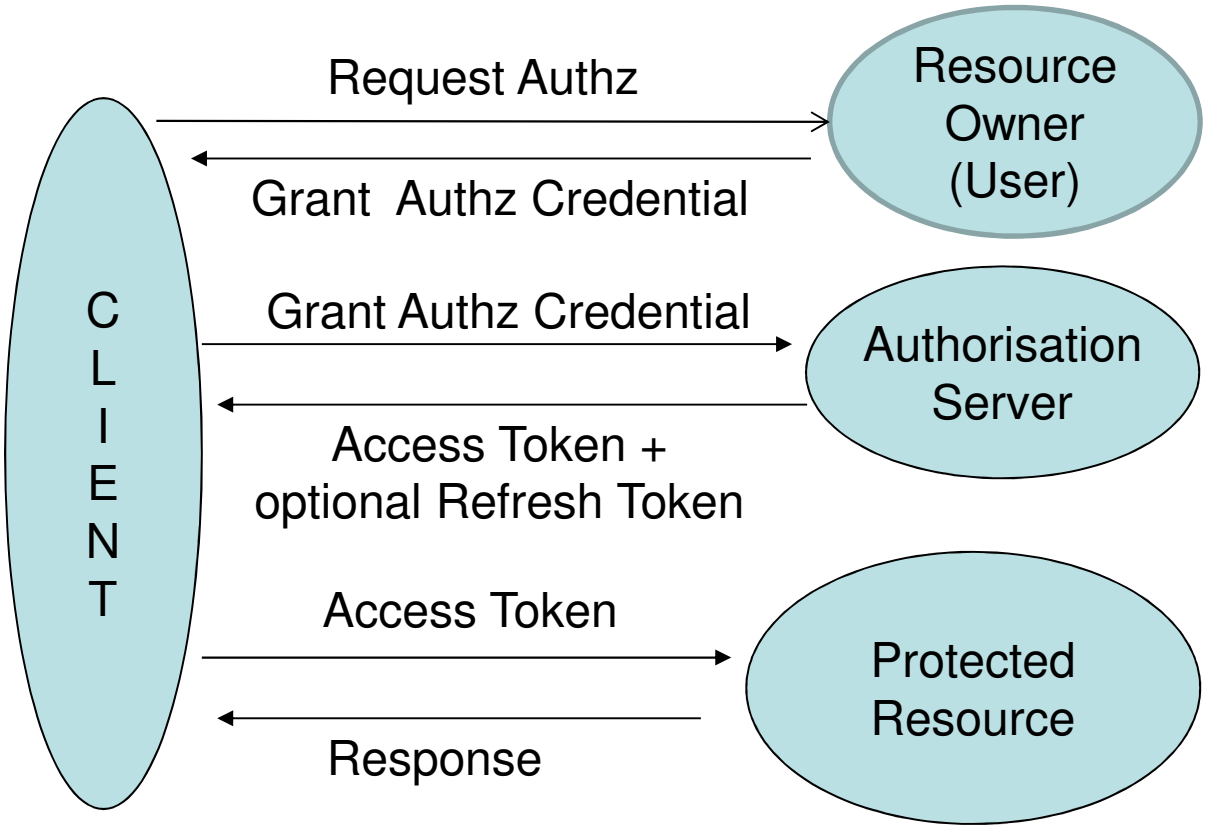
OAuthV2

- A delegation of authority protocol that can be adapted to many different scenarios
- Uses standard Http protocol and no XML
- Can be used to replace SAML protocol
- Has its own terminology which is different to previous ones
  - no IDP, no SP
  - Instead has 4 parties: Authorisation Server (IdP, AA), Resource Owner (user), Resource Server (SP) and Client (Application making requests on behalf of the resource owner)
- Still an Internet Draft

OAuth 2 Mode of Operation

- The client obtains an access token - a string denoting a specific scope, lifetime, and other access attributes
- Access tokens are issued to third-party clients by an authorization server with the approval of the resource owner
- The client uses the access token to access the protected resources hosted by the resource server
- Resource server trusts the external authorisation server (or they might be part of the same service)

OAuth2 Conceptual Protocol Flows



- Access token is a short lived bearer token. When it has expired
- the Client sends the Refresh Token to the Authorisation Server (instead of Credentials) and is returned a fresh Access Token

OpenID Connect

- Based on OAuth 2 protocol
- The protocol flow is similar to OAuth2
- Whereas OAuth 2 specifies two response types: token and code
- OpenID Connect specifies 4 new response types so that OpenID authn tokens can also be returned
  - id_token token – Both an access token and id_token should be returned
  - code token – Both an access token and an authorisation code should be returned
  - code id_token – Both an access code and an id_token should be returned
  - code id_token token – An access token, id_token and authorisation code should be returned
- Still in draft specification stage

Limitations of Current FIM Systems

- Many technical constraints
  - Cannot use credentials from multiple AAs in the same user session (attribute aggregation)
  - Often don't provide user consent
  - Lack strong assurance and fine grained access control
- Liberty Alliance was never adopted – too complex?
- OpenID Connect and OAuthv2 are still under development
- Shibboleth and OpenID are open to phishing attacks
- Microsoft discontinued CardSpace 1Q 2011

Conclusions

- Federated Identity Management is still in a considerable state of flux
- Many different initiatives, all lacking in some capability or other (security, privacy protection, usability, scalability, ease of administration etc)
- None of today's systems will be the FIM system we will be using in 5-10 years time
- Not until…

Car Analogy

- We move from here

+

+

- To here