Computer Science and Media, Stuttgart Media University

Identity in web 2.0

Created by Marc Seeger for the lecture "Application Security"

Marc Seeger
19.02.09

The Problem

In the current web 2.0 landscape, people have the ability to participate in a huge number of services. They keep track of their peers on Facebook, create or watch videos on YouTube, upload pictures to Flickr, keep friends informed via Twitter, write about things they care on Blogspot, manage career connections on LinkedIn and contribute towards a global knowledge on Wikipedia.

While the generated content still is primarily used on the platform it originates from, so called "mash ups" are combining content from several sources (e.g. Wikipedia and Google maps) to enrich the users experience for the particular service (in our Google maps + Wikipedia example, this could be something like "show me things that happened here in history").

The next logical step in this "combination" of content is said to be something called the "semantic web" which basically uses metadata to help computers process generated content. While combining and working with content seems to be one of the "big things" in the next years, the other main entities that web 2.0 consists of hasn't received that much attention yet: the user.

Content is generally expected to be "free" and to evolve over time by being used in mashups, being modified (e.g. games) or remixed (e.g. music).

The "prosumers" generating this content though, have a direct correspondence to a person in real life and aren't expected to change over different websites.

The current way the "internet landscape" handles this, is by forcing the user to create an account for every single service. Those accounts usually aren't linked at all and nobody can guarantee that the account called "JohnDoe2000" on Flickr represents the same person as the user "JohnDoe2000" on YouTube.

In some situations, users generate alter egos on the web for themselves, but as soon as you start producing content that you'd like people to associate with you in real life, you end up with the situation that the only thing connecting you with the content spread over all of those platforms is the alias you happen to have chosen for yourself. Especially if you have a common name or chose a common nickname for your "online identity".

There really seems to be the need for a user-centric identity approach that moves away from the simple "nickname on this community" approach to something more centralized.

This would effectively allow users to use different services/communities without having to signing up on each and every one of them.

The general solution

In real life

To find a proper technical solution, it is often a good idea to look for matching patterns in real life. A lot of the advances in modern technology were heavily influenced by already existing things in nature.
The whole field of Bionics (also known as Biomimetics) has supplied a number of fields with new ideas. The fields range from political science to car design to aeronautics to computer science.

Especially in computer science, things like cybernetics, swarm intelligence and artificial neural networks have benefited from or even being modeled after patterns seen in biology.

The imitating/replicating/copying from biology to produce new technical solutions can be basically broken down into three different areas:

- Mimicking structures/mechanisms (e.g. in engineering to produce efficient and durable materials)
- Replicating manufacturing methods found in nature (e.g. in pharmacy to produce chemical compounds)
- Imitating organizational principles from "social" organisms (ants, bees, humans, ...)

While structures and manufacturing methods won't get us very far with our problem, we can look at our current human society and look for efficient ways to handle a person's identity.

A pretty powerful concept of providing strangers with a proof of your identity or your membership in some organization is an identification card. Your local library may have one, your university probably has one, and your country does too (in Germany it's called "Personalausweis").

An identification card does even do more than JUST provide your name, it usually also provides a picture of your face (which humans tend to be pretty good at comparing), your age and "organization specific" things like e.g. your nationality or your status "Platinum membership".

This does not only allow you to prove who you are in concern to your name, but also allows you to provide additional credentials. This e.g. allows you to simply show your ID when buying e.g. something alcoholic to drink and thereby proof that you are over the legal drinking age.

Although the cashier behind the counter doesn't know you personally, he is able to trust the ID card because he can identify you from the picture and can identify the ID as something published by the government (in Germany: The "Bundesdruckerei").

So by simply presenting the cashier with something you have (the card and your face) that is approved by somebody he trusts (the organization he works for or the country he lives in), he is able to trust the additional credentials being displayed on the card.
This relationship allows asymmetric trust between several previous unknown parties.

In technology

The "real life" solution of an ID card is pretty much what systems like Kerberos or asymmetric cryptography systems (e.g. RSA) already provide to be able to implement single-sign-on mechanisms inside of corporate networks. While this might be nice, it requires a trusted 3rd party (for example the government) which defeats the whole decentralized "mashup" character of our current web 2.0 landscape.

Our main goal was to be able to have an "identity sharing" mechanism which allows us to provide something that simply states that we are person xyz.
It might also provide additional information (e.g. an email address, our age, ...), but they don't need to be (and usually won't be) relied on by the party we provide our identity to.
The only thing that must be secured is, that nobody else is able to use our "chosen" identity to log in to a system that features our solution.
To keep this from happening, you can use everything from a simple authentication using a password to a multi-factor authentication, depending on the needs of the user.

A technology that tires to fill this void is called "OpenID" and the main topic of this paper.

OpenID

Before going into detail about the different parts of OpenID, please keep two things about OpenID in mind:

1. You can claim that you own an OpenID
2. You can prove that claim

That's the main thing behind OpenID. Nothing more, nothing less. And this simplicity is basically the beauty of it. If the history of computer science has taught us anything, it's that there are two things that allow a technology/algorithm/system to thrive:

- It has to be simple
- It has to be open

We've seen this with HTTP, SMTP, RSA, TCP, IP and with basically anything that keeps the internet going as it is at the moment.
OpenID follows the UNIX philosophy of only solving one tiny problem and leaving the rest of the "big picture" up to other tools/services.
In the case of OpenID, it's a problem that could be formulated such as:

- "Does the user trying to log in with OpenID 'johndoe.net' really own the OpenID 'johndoe.net'?"

Now let's look into the details:

Namespace

As with anything that involves identification technology, one has to pay attention to the "main identifier". That's why, a lot of the time, you have to provide your date and place of birth in addition to your name in the real world.
There are a LOT of people called "Marc Seeger", only a few of them were born in Stuttgart and probably only one on March 28th, 1985. This is also why our ID card features a number at the bottom which is unique and created by adding together some other information (e.g. your birthday).
As numbers are generally a bad idea to choose as a unique identity (because people can't remember them), OpenID uses another unique naming scheme already present and established on the internet: Uniform Resource Locators (URLs).
Inside the World Wide Web, URLs are unique and within its pure form (without censorship, transparent proxies, load balancing etc), everybody trying to visit an http server at a certain URL will end up at the same server.
This concept of using URLs to create separate namespaces has also been adopted in other areas such as programming languages. To keep classes with the same name from being ambiguous, namespaces are part of the import statement. The namespace in Java are, by convention, started with the element which would be last in the domain e.g. "org.apache.xmlrpc.server.RequestProcessorFactoryFactory".
OpenID uses the default internet-domain-naming-scheme which is well known to users, e.g. "marc-seeger.de".

Identification

We now know that our OpenIDs look just like regular domains. To be able to compare OpenIDs to actual real-world ID cards, we have to think about the way they are used. Usually, you are simply able to show your ID card and after recognizing the type of ID (e.g. your "Personalausweis") and comparing the picture on the ID to you, the identification is pretty much finished. This works because the person you are identifying to recognizes your ID card and believes it to be genuine. In most official ID cards, holograms or other things are embedded which will result in a big effort if somebody tries to forge this ID card.
As everything digital can be easily copied, the protection from forgery must rely on another mechanism. In OpenID, the usual identification process requires the user to actually be redirected to his Identity Provider and prove that it's actually him wanting to identify to some other person. This works by means of passwords, secure tokens or any other authentication principle (be it single factor or multi factor).
The real world equivalent of proving our "Personalausweis" would actually be an employee of the Bundesdruckerei following you where ever you go and telling other people that he actually manufactured this Personalausweis and that it's an official document. While some providers such as VeriSign offer additional security such as cryptographic tokens, the main idea behind OpenID is NOT to actually show that an OpenID in question belongs to exactly this person in real life, but rather that the user trying to use the OpenID is in control of it. This allows single sign-on solutions for the internet as a whole, without the need of services to work together. To actually be able to present credentials that are

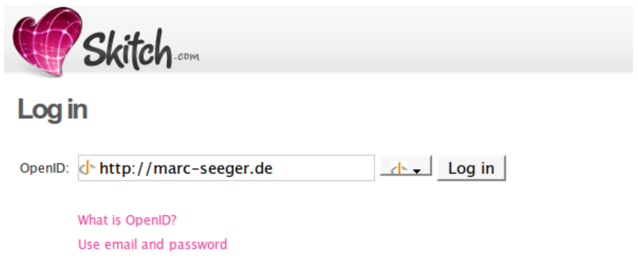
vouched for by a third party, you'd have to drop the decentralized nature of OpenID and rely on "big" identity providers such as VeriSign.

Technical Implementation

User experience

To look at the technical implementation of OpenID, we first have to look at the "usual" login/signup process on OpenID enabled pages:

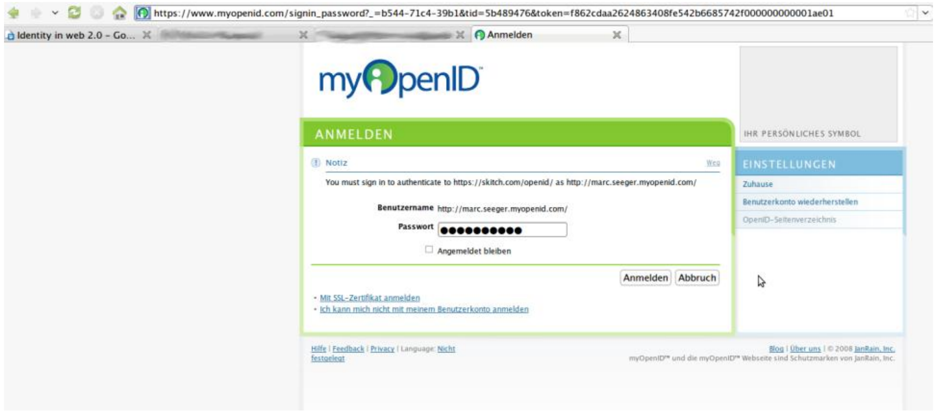
1. Type in your OpenID:



Log in

OpenID: http://marc-seeger.de

2. Wait for the redirect to your OpenID provider and enter your credentials:



https://www.myopenid.com/signin_password?_=b544-71c4-39b1&tid=5b489476&token=f862cdaa2624863408fe542b6685742f000000000001ae01

Identity in web 2.0 - Go... [redacted] [redacted] Anmelden

ANMELDEN

Notiz

You must sign in to authenticate to https://skitch.com/openid/ as http://marc.seeger.myopenid.com/

Benutzername http://marc.seeger.myopenid.com/

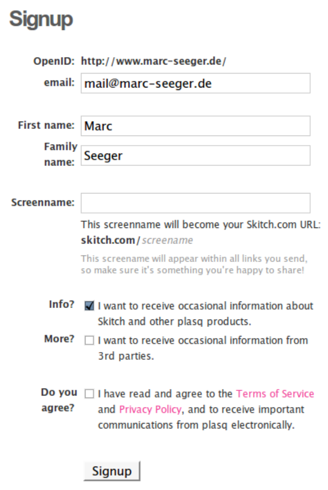
Passwort ●●●●●●●●●●●

☐ Angemeldet bleiben

myOpenID™ und die myOpenID™ Webseite sind Schutzmarken von JanRain, Inc.

myOpenID
Weg
Anmelden
Abbruch
Mit SSL-Zertifikat anmelden
Ich kann mich nicht mit meinem Benutzerkonto anmelden
IHR PERSÖNLICHES SYMBOL
EINSTELLUNGEN
Zuhause
Benutzerkonto wiederherstellen
OpenID-Seitenverzeichnis
Hilfe | Feedback | Privacy | Language: Nicht festgelegt
Blog | Über uns | © 2008 JanRain, Inc.

3. Wait for the redirect back to the original page:



Signup

OpenID:	http://www.marc-seeger.de/
email:	mail@marc-seeger.de
First name:	Marc
Family name:	Seeger
Screenname:	
	This screenname will become your Skitch.com URL: skitch.com/*screename*
	This screename will appear within all links you send, so make sure it's something you're happy to share!
Info?	☑ I want to receive occasional information about Skitch and other plasq products.
More?	☐ I want to receive occasional information from 3rd parties.
Do you agree?	☐ I have read and agree to the Terms of Service and Privacy Policy, and to receive important communications from plasq electronically.

While looking at these pictures, you might have noticed something.
Although I put in marc-seeger.de as my OpenID, I ended up at marc.seeger.myopenid.com. This is due to the fact that I didn't want to bother with actually setting up my OWN OpenID provider software and I chose to rather have myopenid.com manage my credentials. To be able to still use my personal domain marc-seeger.de as my OpenID, I had to put directions for the website that wants to verify my OpenID in my homepages header:

```
<link rel="openid.server"
 href="http://www.myopenid.com/server" />
 <link rel="openid.delegate"
 href="http://marc.seeger.myopenid.com/" />
```

These few lines of HTML code signal skitch.com that the "real" OpenID server which will handle further requests and provide identification isn't at this domain but rather at myopenid.com/server and that marc.seeger.myopenid.com wants to authenticate.

Behind the scenes

To look at what's going on behind the scenes in detail, a short look into the official specification[1] would be recommended. As someone who has not read all that much about OpenID before, it is good to understand some of the important lingo in a non-technical manner. There is also the section "Terminology" in the OpenID specs[2], but I think it might help to keep it simple to get the basic idea:

- "OpenID provider" (OP): Basically an "OpenID Server". It will respond to the OpenID protocol and verify claims of identity.
- "relying party": a web page that relies on the OP to confirm the "identity" for the user (rather: confirm the fact that the URL given by the user is actually owned by the user)
- "User-Agent": usually a user's web browser

Here is the "Protocol Overview" from said specification:

1. The end user initiates authentication (Initiation) by presenting a User-Supplied Identifier to the Relying Party via their User-Agent.
2. After normalizing (Normalization) the User-Supplied Identifier, the Relying Party performs discovery (Discovery) on it and establishes the OP Endpoint URL that the end user uses for authentication. It should be noted that the User-Supplied Identifier may be an OP Identifier, as discussed in Section 7.3.1 (Discovered Information), which allows selection of a Claimed Identifier at the OP or for the protocol to proceed without a Claimed Identifier if something else useful is being done via an extension (Extensions).

[1] *http://openid.net/specs/openid-authentication-2_0.html*
[2] *http://openid.net/specs/openid-authentication-2_0.html#terminology*

3. (optional) The Relying Party and the OP establish an association (Establishing Associations) -- a shared secret established using Diffie-Hellman Key Exchange (Rescorla, E., “Diffie-Hellman Key Agreement Method,” .) [RFC2631]. The OP uses an association to sign subsequent messages and the Relying Party to verify those messages; this removes the need for subsequent direct requests to verify the signature after each authentication request/response.
4. The Relying Party redirects the end user's User-Agent to the OP with an OpenID Authentication request (Requesting Authentication).
5. The OP establishes whether the end user is authorized to perform OpenID Authentication and wishes to do so. The manner in which the end user authenticates to their OP and any policies surrounding such authentication is out of scope for this document.
6. The OP redirects the end user's User-Agent back to the Relying Party with either an assertion that authentication is approved (Positive Assertions) or a message that authentication failed (Negative Assertions).
7. The Relying Party verifies (Verifying Assertions) the information received from the OP including checking the Return URL, verifying the discovered information, checking the nonce, and verifying the signature by using either the shared key established during the association or by sending a direct request to the OP.

This is part of the specification, so the wording might be chosen very carefully. If someone wanted to "dumb it down" a bit, this is basically the protocol:

1. You enter your OpenID at a webpage supporting it
2. The service behind the webpage itself goes to the URL you entered as your OpenID and looks in the HTML header for a link element with a rel="openid.server" attribute (e.g. <link rel="openid.server" href="http://www.myopenid.com/server" />)
3. The service establishes a shared secret with the OpenID server from the link element by using the Diffie Hellmann key exchange
4. The service redirects you to your identity provider
5. You log in to your identity provider and get redirected back to the original web page you wanted to log in to. This redirection also contains a signed message that either states that the login was successful or that it was unsuccessful.
6. The web page checks the message embedded in the redirection using the key established in step 3 and acts accordingly

The "logging in" part isn't specified. This could be a password, an IP based filter, a SecurID keyfob or even out of band things like a pass code in an SMS message.

Possible OpenID usage

By now you understand the general problem the "social internet" is currently dealing with when it comes to identities and the technical details behind a solution that is called OpenID.
A thing that is missing from the puzzle is they enormous amount of ways OpenID can be used to enrich the user experience.
Here are some examples:

- Lightweight accounts: most people don't want to sign up to simply comment to a blog post. With OpenID, they already have an account.
- OpenID friend lists: a major problem of social networks is the fact that you can't properly use your identity on different services without having to manually add all of your friends again. With OpenID, you could easily import and export friend lists based on their OpenIDs which are unique across different sites.
- OpenID proxies: there already are services that allow usage of legacy login systems like e.g. the Yahoo-ID as an OpenID. The identification they use simply implements the legacy system's login API and encapsulates it with OpenID logic
- OpenID cross connects: Why do I have to create an extra album for the pictures of last night's party on e.g. Facebook when I already have them uploaded on e.g. Flickr? OpenID would allow sites to implement a kind of "auto discovery" of public user content on other "well known" web 2.0 sites based on the user's OpenID.
- Preapproved accounts: If you know you're going to work with 4 other persons on a new project, you can already add their OpenIDs as allowed users to e.g. a wiki/forum/project site/... without them having to do anything. You simply send them the link to the e.g. wiki and they are able to log in without having to create yet another account and do the whole email confirmation thing. This resembles public key SSH authentication where you can also add public keys without the key-owners interaction
- Social Comment Spam Whitelist: As our user accounts are now unique across the web, it would be possible to build a "social" whitelist of OpenIDs that are allowed to post comments

These are just a few examples that come up when doing a little bit of brainstorming. There are a ton of other possibilities which would enrich the user experience or solve current problems, but I think the potential of the usage of OpenID technology has been made pretty clear.

Security and OpenID

As with every identification technology, OpenID will have to withstand identity theft and phishing attacks.
A good thing about OpenID is the fact, that it is completely open. This leads to a lot of competition between providers which will speed up the development of security measures against the common forms of identity theft.

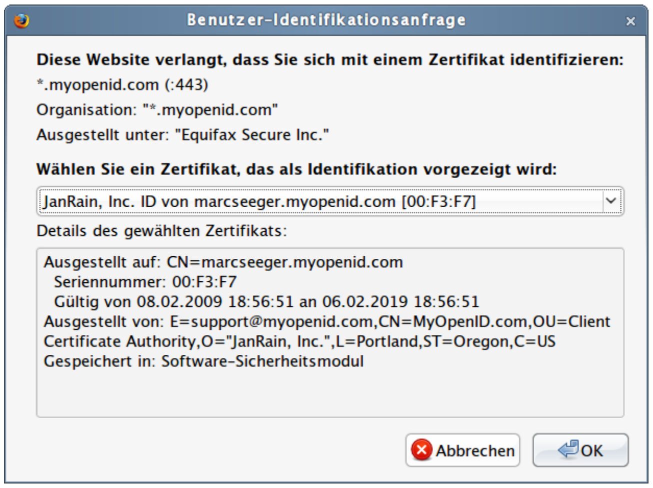
One of the usual attacks against any form of identification is stealing the user's credentials by setting up a fake website and somehow get the user to enter his credentials ("phishing").

Thanks to the fact that OpenID actually DOESN'T specify the way a user has to authenticate, there are already a number of countermeasures in place which help the user against these attempts.

Client Side SSL

The OpenID provider "myopenid" implemented an SSL client side certificate which allows the user to "enable" a computer/browser combination to log in using this certificate. Most people will never have seen the browser specific GUI for this.
The nice thing about his: for the end-user, no additional display of credentials is necessary (e.g. no password to enter). This will make it harder for phishing pages to steal password credentials as the user won't be used to put in his password as the identification is usually handled using the certificate and probably will think twice before entering his data.

The usual thing the end-user will see is something like this:


Benutzer-Identifikationsanfrage

Diese Website verlangt, dass Sie sich mit einem Zertifikat identifizieren:
*.myopenid.com (:443)
Organisation: "*.myopenid.com"
Ausgestellt unter: "Equifax Secure Inc."

Wählen Sie ein Zertifikat, das als Identifikation vorgezeigt wird:
JanRain, Inc. ID von marcseeger.myopenid.com [00:F3:F7]
Details des gewählten Zertifikats:

```
Ausgestellt auf: CN=marcseeger.myopenid.com
  Seriennummer: 00:F3:F7
  Gültig von 08.02.2009 18:56:51 an 06.02.2019 18:56:51
Ausgestellt von: E=support@myopenid.com,CN=MyOpenID.com,OU=Client
Certificate Authority,O="JanRain, Inc.",L=Portland,ST=Oregon,C=US
Gespeichert in: Software-Sicherheitsmodul
```

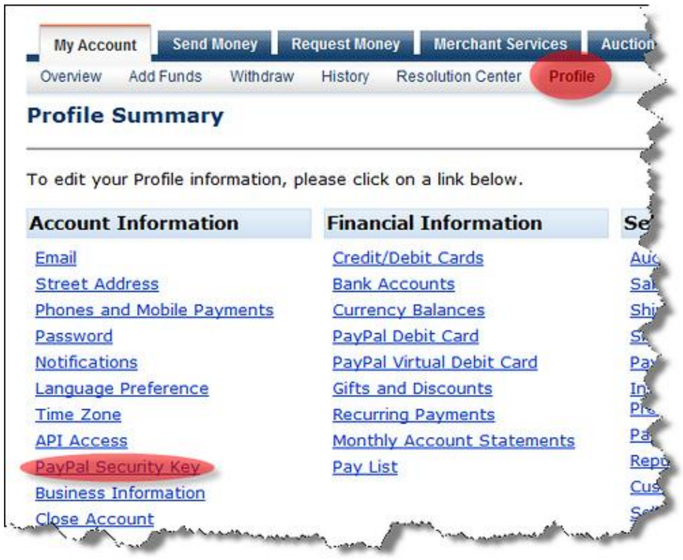
Out of band permanent cookie

Similar to the SSL certificate, a cookie which is being placed after an "out of band" authentication (e.g. by phone) can be used to identify a browser/computer combination. This has the same effect as the client side SSL cert and would it also make it a bit harder for criminals to steal user credentials. The problem with a cookie is, that it has to be transferred over the wire (unlike the client side SSL certificate which can be used to digitally sign data in a request-response scheme).

Keyfob

VeriSign labs have implemented an OpenID provider called "PIP" (Personal Identity Portal, https://pip.verisignlabs.com). Besides supporting client side SSL certificates, they also allow the user to use a "PayPal Security Key". PayPal uses these to secure their end-users online-banking and sells them at < 10 USD. Most "professional" users will have seen a similar technology when dialing in using a VPN connection using an "RSA SecurID".

The availability:



Profile Summary

To edit your Profile information, please click on a link below.

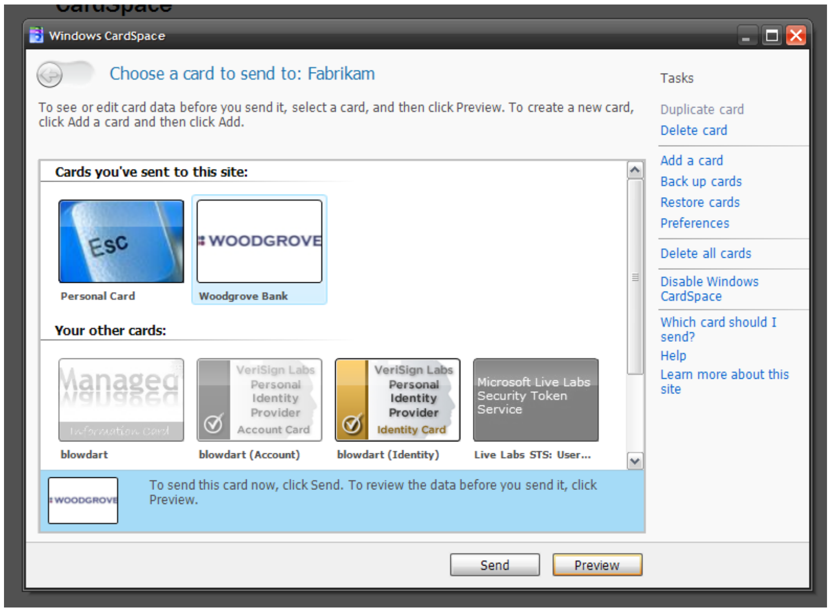
Account Information	Financial Information	Se
Email	Credit/Debit Cards	Auc
Street Address	Bank Accounts	Sal
Phones and Mobile Payments	Currency Balances	Shi
Password	PayPal Debit Card	S
Notifications	PayPal Virtual Debit Card	Pa
Language Preference	Gifts and Discounts	In
Time Zone	Recurring Payments	Pa
API Access	Monthly Account Statements	Rep
PayPal Security Key	Pay List	Cus
Business Information		
Close Account		

The hardware in question:

Cardspace-like Browser support

There are also some ideas which aren't implemented at the moment. One of the most interesting ones is the "cardspace" approach to the problem. While cardspace will be discussed in a later chapter, the main idea is pretty simple and based on the browser supporting the mechanism.
When a website wants to obtain information about the user, the CardSpace UI appears. This UI is part

of the operating system/browser and not just a JavaScript popup. In the UI, a number of user specific "cards" are presented, thus allowing the user to visually distinguish it from any spoofed phishing site.



Choose a card to send to: Fabrikam

To see or edit card data before you send it, select a card, and then click Preview. To create a new card, click Add a card and then click Add.

To send this card now, click Send. To review the data before you send it, click Preview.

Source: Wikipedia

Adding a "native" authentication support to browsers would basically eliminate the possibility of stealing a user's credentials by means of a "lookalike" authentication webpage.
As the authentication itself is not part of the OpenID standard, a lot of providers have emerged that use one of the mentioned mechanisms for authentication. A good list can be found at the "List of OpenID providers"[1] on Wikipedia:

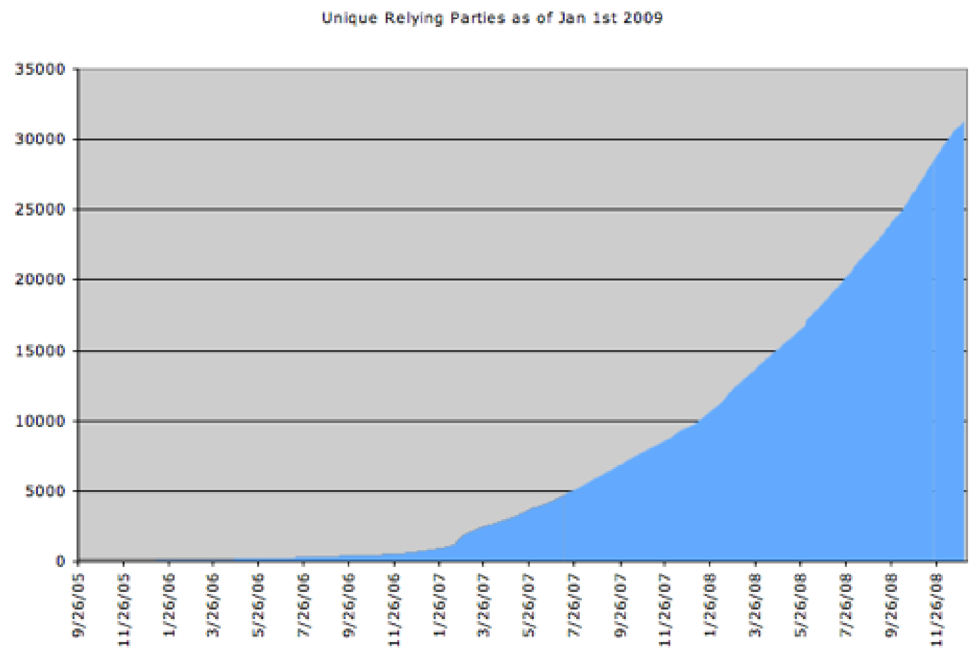
- clavid.ch: Independent Swiss OpenID-Provider supporting several authentication types such as Swiss Post Certificate, AXSionics InternetPassport, YubiKey, Username/Password or One-Time-Passwords.
- certifi.ca: Uses SSL client certificates to authenticate users.
- MyOpenID: Uses SSL certificates, InfoCards or CallVerifID.
- TrustBearer OpenID: Uses smart cards, security tokens or biometric readers to authenticate users.
- OpenID in Estonia: Uses Estonian eID smart cards and GSM SIM cards to authenticate users.
- PIP by VeriSign Labs: Uses security tokens, InfoCards or combined password + SSL certificate/one-time PIN authentication
- Beemba: Supports Information Cards as well as traditional forms-based authentication.
- WSO2 Identity Solution: Supports Information Card based login as well as username/password authentication.
- iden.tt: Strong Identity for OpenID and CardSpace with fully portable multi-factor authentication.

[1] *http://en.wikipedia.org/wiki/List_of_OpenID_providers#Strong_authentication_providers*

Usage of OpenID

A good overview of the amount of sites and companies accepting and implementing OpenID into their systems can be found on blog.janrain.com.

A short quote from the blog entry Relying Party Stats as of Jan 1st, 2009[1]:
31,185 -- That's how many unique web sites are OpenID enabled (as seen from MyOpenID.com) as 2008 ended. The actual number of destination sites is probably higher since we count umbrella sites like blogger & livejournal as single sites, even though they each have thousands unique blogs that each accept OpenID. 31K is a healthy increase over the 9.6K sites we started the year with. I predict we'll leave 31k in the dust in 2009.



[1] http://blog.janrain.com/2009/01/relying-party-stats-as-of-jan-1st-2008.html

Important Websites

There are basically two ways, websites can make use of OpenID:

1. Become an OpenID Provider (OP)
2. Accept OpenID as a means of authentication (thus becoming a "Relying Party")

To throw in some of the big names that already do support OpenID in one way or another:
Relying Party:

- dailymotion.com
- mapquest.com
- SourceForge.net
- technorati.com
- ma.gnolia.com
- wikitravel.org
- identi.ca
- ustream.tv
- smugmug.com
- stackoverflow.com

OpenID provider:

- Google (in part: http://code.google.com/intl/de-DE/apis/accounts/docs/OpenID.html)
- MySpace (http://www.techcrunch.com/2008/07/22/myspace-confirms-openid-support-launches-data-availability-on-flixster-and-eventful/)
- Windows Live ID (http://winliveid.spaces.live.com/Blog/cns!AEE1BB0D86E23AAC!1745.entry)
- Yahoo! (me.yahoo.com/<username>)
- Blogger (<username>.blogspot.com)
- Live journal (<username>.livejournal.com)
- AOL/AIM (http://openid.aol.com/<screenname>)
- Flickr (http://flickr.com/photos/<username>)
- VeriSign (https://pip.verisignlabs.com)

Support from the industry

To get a small impression of which companies actually are involved and interested in the development of OpenID, one can simply take a look at the current (February 2009) board of the OpenID Foundation[1]:

[1] *http://openid.net/foundation/*

Community Board Members

- Brian Kissel (JanRain) - Chair
- Chris Messina (Vidoop)
- David Recordon (Six Apart) - Committee Liaison
- Eric Sachs (Google)
- Joseph Smarr (Plaxo)
- Nat Sakimura (NRI) - International Liaison
- Scott Kveton (Vidoop) - Vice Chair
- Snorri Giorgetti

Corporate Board Members

- Facebook - Luke Shepard
- Google - DeWitt Clinton
- IBM - Tony Nadalin
- Microsoft - Michael B. Jones - Secretary
- PayPal - Andrew Nash
- VeriSign - Gary Krall
- Yahoo! - Raj Mata - Treasurer

Estonia

One particularly interesting OpenID provider is OpenID.ee.
A short quote from their webpage[1]
OpenID.ee is an OpenID provider that uses Estonian eID smart cards and Mobile-ID SIM cards[1] for user identification and authentication. It gives users control over their private data and allows to remain partially anonymous on the net.

The service uses the "ID-card" which is an official document in Estonia. There is also the possibility to use something called "Mobiil-ID" which is basically a GSM SIM card which can also be used to digitally sign documents. Here are some quotes from the id.ee website describing the "ID-card":

- The ID-card is valid everywhere in Estonia. According to the law, no other document can be required for the purposes of personal identification in lieu of the ID-card.
- The ID-card can be used for using Internet-based services provided by the state as well as by several private enterprises.
- The ID-card can be used for issuing digital signatures. The use of digital signatures and electronic documents is more convenient and less expensive than paper-based transactions.

[1] *http://www.openid.ee/*

- The ID-card provides you with a personal @eesti.ee e-mail address that constitutes an easy and convenient way for state institutions, companies and other interested parties to communicate with you.
- The ID-card is the most straightforward, convenient and safe way of self defense for users of Internet banking systems and other web-based services.

The OpenIDs that openid.ee allows to use are built using a number of different schemes. Here's a quote from the opend.ee developer section:

Integrators wishing to build upon OpenID.ee should check for the following URL types:

- https://openid.ee/e/[your].[name](.number) e.g. https://openid.ee/e/martin.paljak or https://openid.ee/e/joe.doe.1. If there are more Estonian ID-card holders with the same name, an index is appended to their name. This form is useful on OpenID 1.1 sites where the user has to type in the used OpenID. Every user has only one such URL.
- https://openid.ee/u/[countrycode]:[personalidcode] e.g. https://openid.ee/u/EE:38207162722. This scheme allows us to extend our service into other countries. These URLs are meant for government services that rely on the personal ID code as the unique identifier.
- https://openid.ee/a/[random token] e.g. https://openid.ee/a/29dd7a86e7020ab672731508e07b71ddde05a7e2. These URLs contain random data, and every person can have several such OpenIDs. But on one website the random token remains unique and persistent.

1

http://emt.ee/wwwmain?screenId=content.private&componentId=ContentProviderComponent&pageId=1141&menuId=&language=ENG

Competition

Microsoft Passport

While Passport isn't really 100% comparable to OpenID as the data is centralized at Microsoft, it is the authentication system with the biggest user base.

Microsoft rolled out Passport in its original form in 1999. Over the years, names changed to ".NET Passport" and "Windows Live ID".

In the beginning, many industry players feared that Microsoft might start controlling an important part of the internet and tried to create an alternative called "Liberty Alliance Project".

Neither Passport nor Liberty was really accepted by the internet community.

Dick Hardt summed up the disadvantages of Passport in his blog post "Identity 2.0 Passport vs. OpenID vs. Facebook Connect"[1]. Here are, according to Dick Hardt, the main disadvantages:

- Cost: Quoted price was $10,000 per site. Out of range for small sites.
- Installation: Proprietary code supplied by Microsoft. Big sites using open source were not all that keen to put some proprietary code into a critical part of their infrastructure. UNIX code was problematic to install and get running.
- Functionality: SSO, minimal profile at times.
- Centralization: although Microsoft announced in 2001 that enterprises could run their own federated identity servers, it was not an open environment and the perception that the system was Microsoft controlled was firmly entrenched in the market.

The system is still by far the largest authentication system in the world, yet few of the people who own a passport account will actually know it (or use it for that matter). A lot of the accounts are created using one of the following ways:

- The Windows XP registration wizard asks the user to create a passport account
- MSN Hotmail doubles as a passport account
- The "Xbox Live ID" is connected to a Windows Live ID

How Does Windows Live ID Relate to Passport?

Microsoft explains this on the "Introduction to Windows Live ID"[2] page:

[1] *http://identity20.com/?p=153*
[2] *http://msdn.microsoft.com/en-us/library/bb288408.aspx*

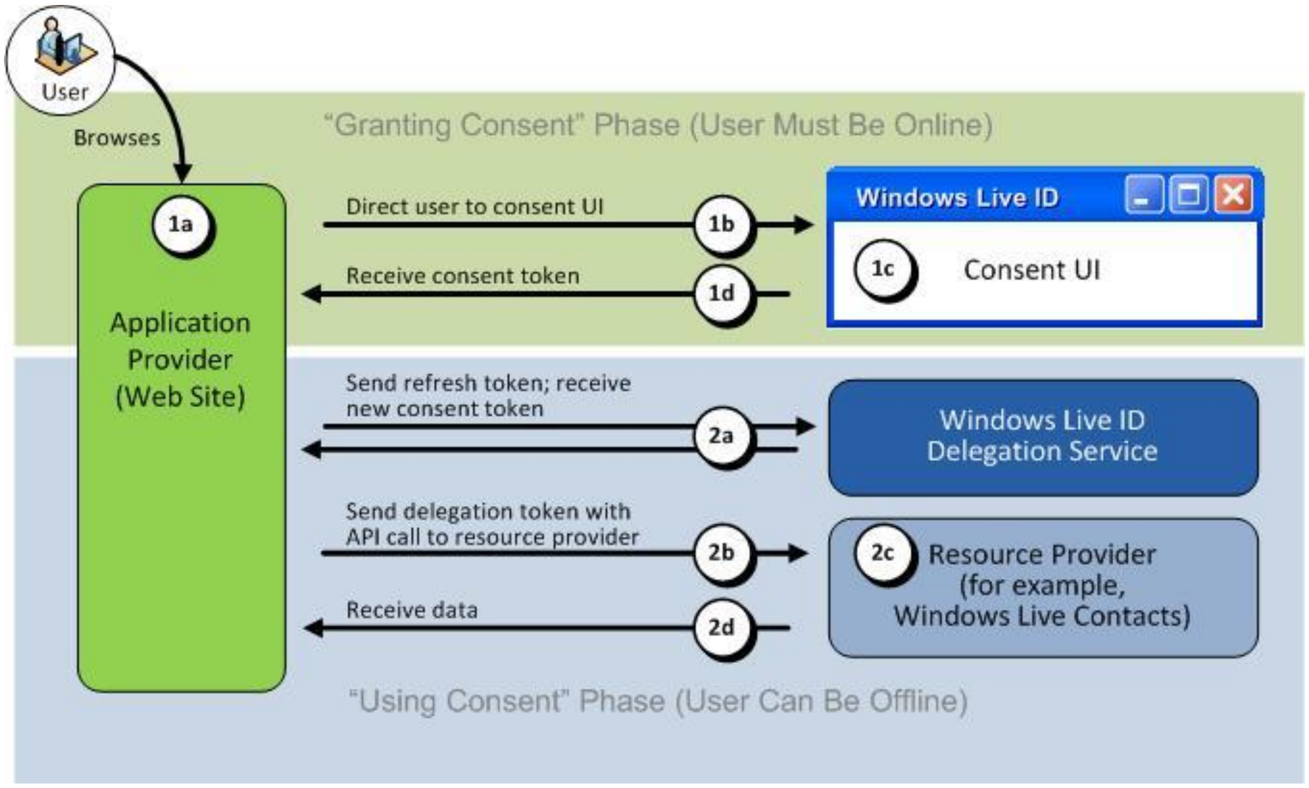
The Windows Live ID service represents the evolution of Microsoft Passport into a world based on federation. Windows Live ID will be the authentication system for all existing and future Microsoft online services. Relying parties (Microsoft properties and those of close partners) who have implemented Passport will be compatible with the Windows Live ID service.

Another area of evolution is towards support for "rich clients" using Web services. By supporting WS-Trust and Windows CardSpace, Windows Live ID will extend its single sign-in framework to the Windows Communication Framework (WCF) employed in many emerging applications.

Technical Overview

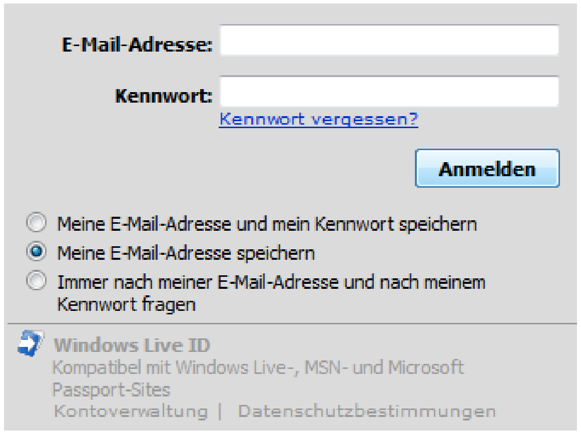
As it would go too far for this paper to describe the live id architecture in detail, an illustration from the "Understanding Windows Live Delegated Authentication"[1] whitepaper describes the basic process:

[1] *http://msdn.microsoft.com/en-us/library/cc287613.aspx*


User
Browses
"Granting Consent" Phase (User Must Be Online)
1a
Application Provider (Web Site)
Direct user to consent UI
1b
Receive consent token
1d
Windows Live ID
1c
Consent UI
Send refresh token; receive new consent token
2a
Windows Live ID Delegation Service
Send delegation token with API call to resource provider
2b
2c
Resource Provider (for example, Windows Live Contacts)
Receive data
2d
"Using Consent" Phase (User Can Be Offline)

User Experience

The Passport user experience is reasonably straight forward and similar to the usual OpenID approach.

1. You see an icon ( ) signaling that the site uses MS Passport, you click on it.
2. You type in your Passport credentials if you don't have an active session
3. 

   E-Mail-Adresse:

   Kennwort:

   Kennwort vergessen?

   Anmelden

   Meine E-Mail-Adresse und mein Kennwort speichern
   [✓ Meine E-Mail-Adresse speichern]
   Immer nach meiner E-Mail-Adresse und nach meinem Kennwort fragen

   Windows Live ID
   Kompatibel mit Windows Live-, MSN- und Microsoft Passport-Sites
   Kontoverwaltung | Datenschutzbestimmungen
4. You are logged in.

Services supporting Passport

Currently, Microsoft itself is the biggest user of Passport. They have it in several of their web pages and use it in their mail service (hotmail) and their XBOX gaming console.

There are also sites owned by Microsoft such as expedia.com which also use it, but other than that, there isn't a broad use of Passport.

One of the bigger sites that used passport was eBay, but in a press release back in 2005, they announced that "As of 24 January, 2005, eBay will no longer support Microsoft Passport as a means to sign in".

Passport and OpenID

A post on the Windows Live ID blog dated 27.10.08 announced the plan to make Windows Live ID double as an OpenID provider:

Beginning today, Windows Live™ ID is publicly committing to support the OpenID digital identity framework with the announcement of the public availability of a Community Technology Preview (CTP) of the Windows Live ID OpenID Provider.

You will soon be able to use your Windows Live ID account to sign in to any OpenID Web site!

The Windows Live ID OpenID Provider (OP) enables anyone with a Windows Live ID account to set up an OpenID alias and to use that alias for identification at an increasing number of OpenID 2.0 relying party sites

Microsoft Cardspace

Microsoft Cardspace (previously called "Infocard") is Microsoft's next try to establish an identity framework after Passport pretty much failed.
While Passport was basically an Identity Provider, Cardspace actually implements a kind of "identity selector" using the "identification card" metaphor I talked about in the first chapter as a central design motive.
This allows shielding the user from all the complexities of what's going on behind the scenes and simply selecting the matching identity for the service in question.
According to Microsoft Engineer Vittorio Bercotti, cardspace is meant to be part of a "bigger picture" which changes the current systems which are only a means of authentication using a username and a password to a more natural way of reasoning about your identity.
Cardspace is what makes this change visible to the windows and takes care, under the hood, that the protocol works.
While cardspace still can go "back down" to the old username and password approach, it is on a cryptographically more sound basis. Cardspace uses cryptographically signed tokens build on top of the WS* Infrastructure (WS-Security, WS-Trust, ...).
The cardspace system distinguishes between managed cards which are given to you by a third party and self-issued cards (also known as: self-asserted cards) which themselves can be accessed using different levels of authentication, ranging from a simple password to a complete smartcard-based web service approach.
While the technological foundation of Cardspace is a pretty interesting one, it would go over the scope of this paper.
En excellent source of information for further information is a video which features Vittorio Bertocci and Caleb Baker called "Understanding CardSpace and the complexities of identity" (http://channel9.msdn.com/shows/Going+Deep/Vittorio-Bertocci-and-Caleb-Baker-Understanding-CardSpace-and-the-Complexities-of-Identity/) and the other "Infocard" videos on channel 9.

Windows Live ID and Cardspace

In August of 07, Microsoft released this press release:

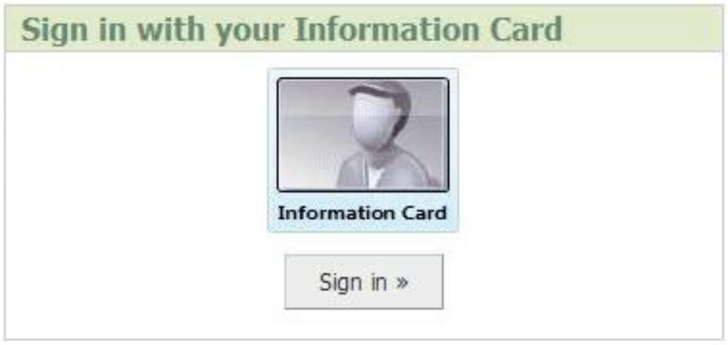
Windows CardSpace is a new way to sign in securely and conveniently into websites. And now you can use CardSpace with your Windows Live ID account! Using CardSpace with Windows Live ID means you don't use a password to sign-in. Instead, just send your Information Card to Live ID to identify you and get signed into Hotmail, Windows Live Spaces or any other site that accepts Windows Live ID.
[...]

Nayna Mutha, Program Manager - LiveID
Rob Franco, Lead Program Manager - Windows CardSpace

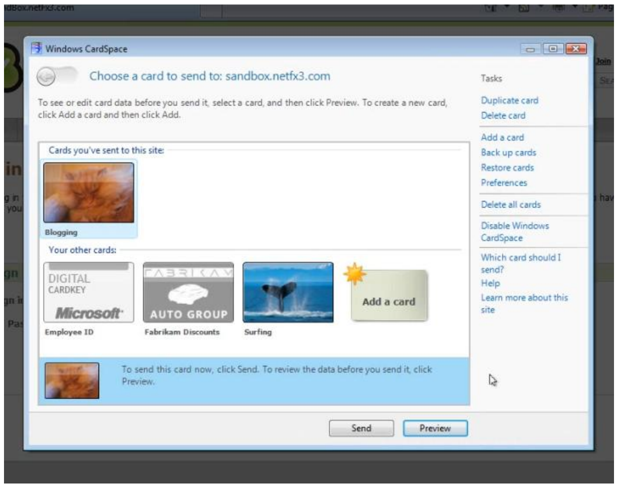
User Experience

This is what the usual user will see when using cardspace:

1. an icon displaying the availability of cardspace



2. upon clicking the sign in button, the identity selector appears and asks for the identity to send to the site



Choose a card to send to: sandbox.netfx3.com

To see or edit card data before you send it, select a card, and then click Preview. To create a new card, click Add a card and then click Add.

Cards you've sent to this site:

Blogging

Your other cards:

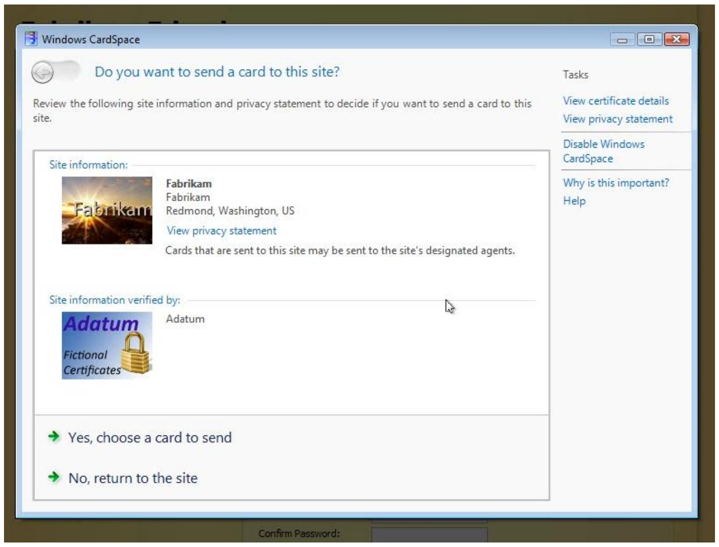
Employee ID

Fabrikam Discounts

Surfing

To send this card now, click Send. To review the data before you send it, click Preview.

3. if the user hasn't been to the site before, a bit of information, comparable to SSL certificate popups might appear



Windows CardSpace

Do you want to send a card to this site?

Review the following site information and privacy statement to decide if you want to send a card to this site.

Site information:

Fabrikam
Fabrikam
Redmond, Washington, US
View privacy statement
Cards that are sent to this site may be sent to the site's designated agents.

Site information verified by:

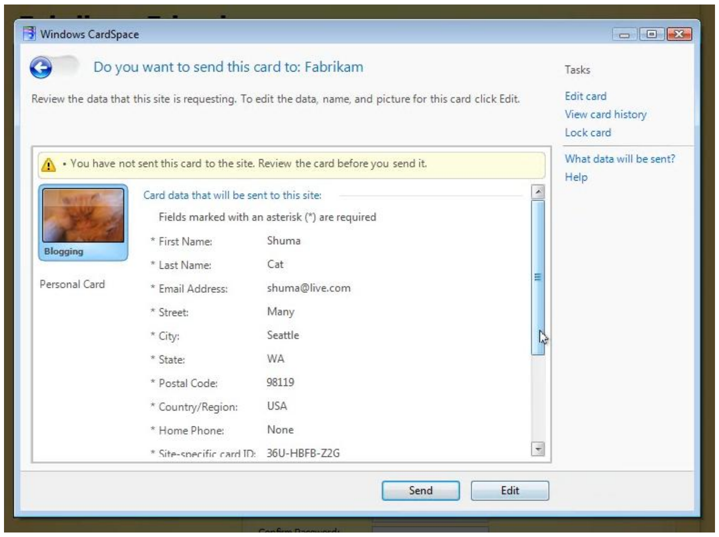
Adatum

Yes, choose a card to send

No, return to the site

Tasks
View certificate details
View privacy statement
Disable Windows CardSpace
Why is this important?
Help

4. before actually sending a card, a user can preview the data that will be transmitted



Windows CardSpace

Do you want to send this card to: Fabrikam

Review the data that this site is requesting. To edit the data, name, and picture for this card click Edit.

• You have not sent this card to the site. Review the card before you send it.

Personal Card

Card data that will be sent to this site:

Fields marked with an asterisk (*) are required

* First Name:	Shuma
* Last Name:	Cat
* Email Address:	shuma@live.com
* Street:	Many
* City:	Seattle
* State:	WA
* Postal Code:	98119
* Country/Region:	USA
* Home Phone:	None
* Site-specific card ID:	36U-HBFB-Z2G

Tasks
Edit card
View card history
Lock card
What data will be sent?
Help

Usage on the web

Cardspace is basically not used by any large non-Microsoft sites.
Windows Live users can use Cardspace to log-in to Windows live using
https://login.live.com/beta/managecards.srf?wa=wsignin1.0
A nice quote from one of the architects was that "Cardspace is one of those technologies where a LOT of people line up to be the second one who adopts it"

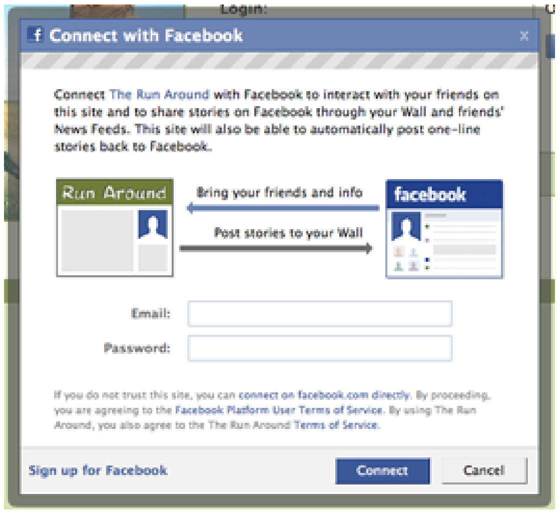
Facebook Connect

While Facebook Connect is comparable to Microsoft Passport, it has one major advantage:
Facebook tries to only have accounts that are "real" identities. Behind every Facebook Account, there is a real-life person and Facebook deletes accounts that are not real people.
As those people are usually connected to other people on Facebook, it is possible to transfer those connections to other sites.
From the Facebook connect developer site:
[...] Developers will be able to add rich social context to their websites. Developers will even be able to dynamically show which of their Facebook friends already have accounts on their sites. [...]
Apparently those connections are even updated on participating sites once the connection status on Facebook changes:
[...] if a user changes their profile picture, or removes a friend connection, this will be automatically updated in the external site. And the users can control who can see what pieces of their information - the same rules that they set on Facebook can be applied through your site too using our dynamic privacy controls.[...]

User Experience

As usual, the user is signaled that he is allowed to log in using Facebook connect by a button that is assigned by Facebook: Connect

Once the user has clicked on the button, a popup opens and asks him to confirm his selection. If the user isn't logged in, the user has to give his Facebook username + password.



Connect with Facebook

Connect The Run Around with Facebook to interact with your friends on this site and to share stories on Facebook through your Wall and friends' News Feeds. This site will also be able to automatically post one-line stories back to Facebook.

Email:

Password:

If you do not trust this site, you can connect on facebook.com directly. By proceeding, you are agreeing to the Facebook Platform User Terms of Service. By using The Run Around, you also agree to the The Run Around Terms of Service.

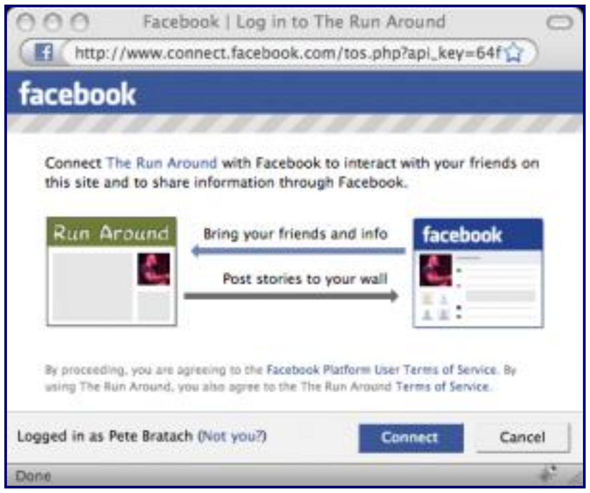
The confirmation window also shows the user what powers he is granting to the webpage in question (e.g. receiving the list of friends, posting images on the user's profile, …)

The Connection Process

The Facebook Developer Wiki[1] describes the process as follows:

1. Client requests page, server sends an HTML response that includes <fb:login-button></fb:login-button> somewhere as a placeholder for the Facebook Connect login button.
2. Browser receives response, executes Facebook JavaScript initialization code. FB.Facebook.init("YOUR_API_KEY_HERE", "xd_receiver.php");
3. Browser pings Facebook for login status. See Cross Domain Communication for details
4. JavaScript library renders the button:

5. User clicks the button. An onclick handler is triggered: FB.Connect.requireSession();
   If the user is logged into Facebook and has authorized the app previously, then the session is returned immediately, and skip to step #7.
6. A popup window opens on Facebook.com, asking the user for permission. If they are not logged into Facebook, then it will ask for email/password as well.


Facebook | Log in to The Run Around
http://www.connect.facebook.com/tos.php?api_key=64f
facebook
Run Around
Bring your friends and info
Post stories to your wall
facebook
Connect
Cancel
Done

Connect The Run Around with Facebook to interact with your friends on this site and to share information through Facebook.

By proceeding, you are agreeing to the Facebook Platform User Terms of Service. By using The Run Around, you also agree to the The Run Around Terms of Service.

Logged in as Pete Bratach (Not you?)

7. Session is returned. The Facebook JavaScript library does two things:

   1. Set some cookies on the application domain to store the session info.
   2. Trigger the callback defined by :

   FB.Facebook.get_sessionState().waitUntilReady(function() { ... });

   The callback typically will just refresh, although in a more advanced site it might just do some ajaxy updates to avoid refreshing the whole page.

[1] *http://wiki.developers.facebook.com/*

8. On the next request to the app server, cookies will be sent that contain signed session information. Server should:

   1. Verify the signature to make sure the data came from Facebook
   2. Retrieve the Facebook user ID
   3. Look up the user's account and "log them in". If they have no account, then create one behind-the-scenes

Security

According to the Facebook Developer Wiki[1], the above flow keeps information secure through several means:

- When the session comes back, it is signed using your secret key. You can use the server-side library to verify that the information came from Facebook and not some hacker.
- All communication is mediated by the browser. A Facebook user ID is not given unless the user sitting at the computer has authenticated to Facebook.
- No information about the application is passed to Facebook (aside from the API key) in the authentication step.

Usage on the web

Among the sites implementing Facebook connect, there are some sites that might be considered to be one of the "big players":

- Cnet
- Gawker
- Gizmodo
- Meebo
- Vimeo
- Joost

A full list of sites is available at the Facebook developers wiki[2]

[1] *http://wiki.developers.facebook.com/index.php/How_Connect_Authentication_Works#Security*
[2] *http://wiki.developers.facebook.com/index.php/Facebook_Connect_Live_Sites*

Summary

To sum up the paper with a small personal touch, I'd like to give a short summary of my thoughts on the services while having written this paper:

OpenID

I personally like the "lightweight" accounts that OpenID allows people to have on sites they don't necessarily want to register for and the fact that I can build up a reputation across multiple sites. The openness of OpenID is also a big plus as competition tends to make products (in this case: the providers authentication methods) better.
Although I think OpenID will continue to find support on "regular" sites, I don't see it gaining any support on "critical" systems (e.g. banking) as this kind of identification would have to come from a trusted open-id provider which might reduce OpenID back to a "one provider per page" situation.

Facebook Connect

I think Facebook connect is especially interesting because of the connections between its user base. I don't think it would have any advantages over a social network which is based on OpenIDs. As Facebook seems to be interested about integrating OpenID into their solution, they could become an interesting OpenID provider though.

MS Passport

I personally think it is dead. It doesn't offer any advantages and follows a "closed" model.

Cardspace

Microsoft has, in my opinion, the most solid underlying technology (WS*) and the integration within Windows and the "card" metaphor will make it a good choice for the average user. As cardspace is more of an identity selector with support for "managed cards", it might even advance into the previously mentioned "critical" systems such as banking sites. Something that might keep cardspace from gaining popularity is its “solid foundation”. The WS* technology is a pretty complicated standard which might scare some developers into implementing something easy and lightweight like opened.