Security, Privacy and Usability Requirements for Federated Identity

Michael Hackett and Kirstie Hawkey
Faculty of Computer Science
Dalhousie University
Halifax, Nova Scotia, Canada
e-mail: mhackett@cs.dal.ca, hawkey@cs.dal.ca

***Abstract*—Federated Identity systems promise to solve the increasingly vexing problem of password overload. However, existing systems, such as OpenID and CardSpace have failed to gain the expected levels of adoption, due in part to usability and security issues, while proprietary systems such as Facebook Connect raise serious privacy concerns over their usage of the data collected. In this paper, we examine two new contenders—BrowserID from Mozilla and WebID from the W3C WebID Community Group—and find that, while both offer significant improvements, we were still able to identify a number of important security, privacy, and usability issues that need to be addressed before beginning to widely deploy these new platforms.**

I. Introduction

Federated Identity systems attempt to solve the growing dilemma among computer users sometimes referred to as "password fatigue". According to one recent study [1], the average computer user has 25 different password-protected accounts, many of them used only infrequently, creating a situation where it is almost impossible to commit all of those usernames and passwords to memory. Coping strategies include the use of password management software, writing down of passwords (often on sticky notes attached to a computer!), and using the same password for all accounts. In addition to reducing security, these often create a dependency on a particular device (where the passwords are stored) or item (such as a notebook) or location (such as one's desk), such that the person may be unable to use their "accessible anywhere" accounts whenever he or she is away from their particular storage medium.

When using a Federated Identity, an individual need only remember one set of credentials in order to gain access to all services, without needing to share those credentials with the service providers. OpenID [2] is one well-known attempt to create such a system, but adoption has been hindered by usability and security issues. Proprietary systems, such as Facebook Connect and Microsoft Passport, have made some gains in usability, but they raise serious privacy concerns for many users, as the companies operating these systems are able to track users' activity and make use of the information they obtain, including making a user's activity public without his or her consent or knowledge.

Another concern is that, for all their convenience, these identity accounts are the keys to a great deal of information about an individual and could become the targets of hackers and identity thieves. If an identity account is compromised, the potential damage could be much more widespread and devastating for the target individual than if a password to a single site were compromised. Therefore, the security of these systems must be scrutinized carefully and the opportunities for user mishaps minimized.

Two newer proposals that are not yet widely deployed but which are garnering much attention are WebID [3], [4] and BrowserID [5], [6]. Both take a decentralized approach to identity management (where there is no one single central authority, such as Facebook), and aim to provide improved usability, security and privacy over prior decentralized systems, such as OpenID.

In this paper, we focus on analysing some of the key security and privacy claims of these new proposals, with the goal of raising alarms early in the design process, when the developers of these schemes still have the ability to solve the issues. We will begin with some further background on the problem being addressed and some useful terminology in section II, and then discuss some related projects in section III. This will be followed by two sections introducing the two proposal being examined here, BrowserID in section IV, WebID in section V. Our analysis of the protocols on a number of key issues appears in section VI, followed by our future work and conclusions.

II. Background

Although the different proposals described herein generally share many of the same concepts, their specifications often use different terminology for these concepts. For ease of comparison, this paper uses the following terms, based on those given by Maler and Reed [7], noting the corresponding term wherever one of the official specifications differs.

- A *service provider* (SP) is some web site or service that the user must sign into to use. When the SP depends on a third party for identity authentication, it is often called a *relying party* (RP), because it relies on the external authentication service.
- An *identity provider* (IdP) is another web site or service whose job is to verify or provide information to aid in verifying the identity of a user.

- A *user agent* (UA) is a web browser or other software application that communicates with a remote system on behalf of the user.

Although both WebID and BrowserID can be used for authenticating software agents (that is, automated systems) as well as human users, for simplicity we will simply use the term *user* for both cases. (The X.509 specification [8] uses the term *subject* to mean "person or system", but this word has several other meanings as well, so we find *user* to be more clear in this context.)

Also, throughout the paper, in the name of grammatical expediency, we will use the conventional user "Alice" to stand in for a general user of either system.

Note that while federated identity systems are often referred to *single sign-on* (SSO)—probably a more end-user-friendly term—this is not strictly accurate. In a true SSO system, one would not be required to sign into each site separately; once you had signed into your identity account, you would automatically be recognized at each cooperating site that you visited. This can be convenient, and might be a reasonable default within an organization, but many users would not be comfortable with applying it to the entire Internet. They may want to use different identities on some sites and have the option of anonymity on others. Furthermore, it would be inconvenient to have to remember to sign out of one identity and into another for each site. Both BrowserID and WebID give you the option to either select an identity to share with each service provider, or to not share one; they never share anything without your explicit consent. While slightly less convenient, from a privacy perspective, this is probably what most people would prefer. (That said, it would almost certainly be possible for someone to create a user agent that did immediately sign into every compatible site automatically, if that was what some users wanted.)

III. Related Work

Of course, neither BrowserID nor WebID is the first attempt to solve the digital identity problem. OpenID is perhaps the best known decentralized web identity system. It uses URLs to identify users, and users are free to choose their identity providers, or may even set up their own IdP. As OpenID "does not require any pre-established trust relationships between IdPs and [RPs]" [9], the barriers to entry for new IdPs and RPs are low. OpenID adoption by users, however, has been hindered by usability issues—particularly with respect to inconsistencies in the sign-in interface [10]—and design features that make users easy prey to phishing attacks. Adoption by RPs has greatly increased since Google and Yahoo came on board as OpenID Providers (and many users now use OpenID without even realizing it, though the provider-specific sign-on buttons present on many RP sites), however, work on improving the user experience and security issues is still ongoing.

Microsoft's CardSpace [11], [12] uses the metaphor of Information Cards that are analogous to the physical identity and membership cards most of us carry in our wallets. Each card holds either a collection of identity attributes (referred to as *claims*) or a link to an IdP that can supply such information on demand. Just as in a physical wallet, users may have cards from several different IdPs, and can choose the one the that is most appropriate for a given context, such as using a government-issued ID to vote online, or a self-issued card containing only an email address for a web discussion board. With claims-based identity, a user might be able to use any one of a number of cards to satisfy a particular RP's authentication request, provided the card includes the requested claims and is issued by an authority that the RP trusts.

Yet, despite bundling the CardSpace client with Windows Vista and Windows 7, thus ensuring an installed base of millions of potential users, claims-based identity has not gained traction with service providers, and Microsoft decided to cancel the product in February 2011. Service providers did not see any compelling advantages to implementing support, and users "were often confused by their first encounter with CardSpace" [13], finding it too complicated in comparison with the username/password forms to which they had become accustomed. In addition, Microsoft admitted that the lack of tools for service providers for creating claims-ready services was another factor in the lack of uptake [14].

Nonetheless, other groups remain interested in the Information Card concept. Higgins[1] and openinfocard[2] are two open-source projects that have also built Information Card clients. The Higgins project, for example, has experimented with several alternative client implementations, such as a Firefox-based browser extension (without the native platform component that CardSpace required) and clients for mobile phones. They have also created native client applications for Linux and Mac OS X, and introduced a web service to solve the problem of sharing Information Cards across devices, with the active clients becoming simple shells that interfaced with the web service.

However, in addition to the aforementioned lack of service provider interest in claims-based authentication, a lack of visibility and an overly technical installation process have further limited Higgins's appeal to a niche of developers and researchers. Also, some aspects of the card selection interface have been identified as "overly intrusive and annoying," particularly when returning to previously visited sites or having to select multiple cards to satisfy a request. The spotty support for various operating systems and browsers, the need (in most cases) to download an active client, and general performance issues may also have contributed to user dissatisfaction [15].

[1] http://www.eclipse.org/higgins/

[2] http://code.google.com/p/openinfocard/

IV. BROWSERID

The first of the schemes to be examined here, BrowserID[3], is a recent initiative (first made public in July 2011) of the Mozilla Identity Team[4]. It proposes the use of an email address as a user's unique identifier, rather than the URIs used in OpenID and WebID. The claim is that most computer users are already comfortable with the concept of an email address representing a particular individual [16], whereas few people have that same association with URIs. We have somehow become accustomed to email addresses so that they no longer look foreign, but URIs still look too "computery". As stated in the original specification for what was called the Verified Email Protocol (from which BrowserID was derived):

> It is understood that "alice@site.com" means that there is a person, here called "alice", who has agreed to trust "site.com" to test her identity and to act as a secure relay for messages. The fact that we use this identifier only for SMTP mail delivery is an accident of history; there is no reason we can't bootstrap from this identifier to other protocols (as recent proposals like Webfinger have made clear). [16]

Furthermore, just as many people have multiple email addresses which they use within different contexts (such as for work, school, or personal life), likewise, with BrowserID, one can have multiple identities or personas online simply by using different email addresses. This again leverages the familiarity and comfort that users have with email addresses. And when anonymity or pseudonymity is desired, one can easily create single-use "throwaway" email addresses for use with BrowserID using any one of a host of email providers.

The BrowserID sign-in flow can be seen in Figures 1–2. It consists of two distinct phases, although these may, in some situations, be executed back to back. In the first phase (as seen in Figure 1), Alice obtains an identity certificate from her IdP that asserts her control of a particular email address. She signs into her IdP, typically with her email address and password as credentials (1). Once authenticated, her UA generates a public/private key pair (2) and stores the private key in the local keyring. The public key is sent off to the IdP (3), where it is bundled into an identity certificate, along with Alice's email address and a validity interval for the certificate, and signed with the IdP's private key (4). Finally, the signed identity certificate is sent to Alice's UA (5), where it is stored locally with the corresponding private key.

Once Alice has her identity certificate, she can then use it to sign into any web site that supports BrowserID, without ever having to create a new username and password, or give her credentials to another site. When Alice wants to sign in with her BrowserID identity, she will typically click a "Sign In" button and select the identity to use (if she has more than one) in a dialog that appears. This kicks off the sequence show in Figure 2. The UA generates an *assertion* that includes the signed identity certificate (obtained earlier) and the address of the web site (the scope of the assertion), and then signs the assertion with the private key associated with the identity (1). The UA sends the signed assertion to the RP, the site she is signing into (2). The RP retrieves the IdP's public key from a well-known location at the domain in the email address (3), and uses the key to verify that the signature on the assertion and identity certificate are valid (4). It also checks that the validity intervals on both the assertion and certificate are still valid, and that the *audience* field in the assertion refers to itself. (This prevents *replay attacks*, where one RP could capture signed assertions and use them to sign in as Alice on other sites.)

Notably, the client side of these processes (apart from signing into the primary authority's account) can be standardized with native browser support or a small Javascript shim that can be used in the absence of native support. This is a significant improvement over the interface inconsistencies present in OpenID [10].

V. WEBID

WebID is a proposal currently being developed, through an open process, by the W3C WebID Community Group[5]. While WebID is designed to provide a federated identity system, it also aims to provide a platform on which to build a distributed social network. This second goal, while also of great value, is outside the scope of this study and will largely be ignored here.

A WebID is a URI that is under the control of the user whose identity it represents. This much is similar to OpenID. In WebID, however, the resource obtained when the URI is dereferenced is a Friend-of-a-Friend (FOAF) [18] profile document that, among other items, includes a public key that is used in the authentication process to verify the user's control of the URI. (The WebID protocol was originally known as FOAF+SSL [4].)

In order to use a WebID as an identity with a service provider, the user must prove ownership (control) of the URI to the relying party. This is done by way of a certificate sent to the RP that includes the WebID URI and that has been digitally signed with the private key associated with the identity. The RP consults the profile at the given URI and verifies the certificate signature using the public key in the profile. If the signature is verified, this indicates that the RP is "talking" to the owner of both halves of the identity's

[3]As of Feb 2012, Mozilla has rebranded the user-facing portion of BrowserID as *Mozilla Persona*. However the protocol itself will still go by the name BrowserID.

[4]http://identity.mozilla.com/

[5]http://www.w3.org/community/webid/

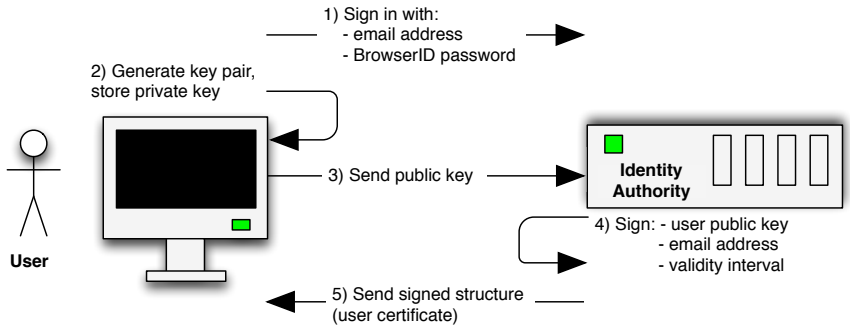


Figure 1: BrowserID identity certificate provisioning.
(Source: [17], used under Creative Commons Attribution-ShareAlike 2.5 license [CC-BY-SA 2.5])

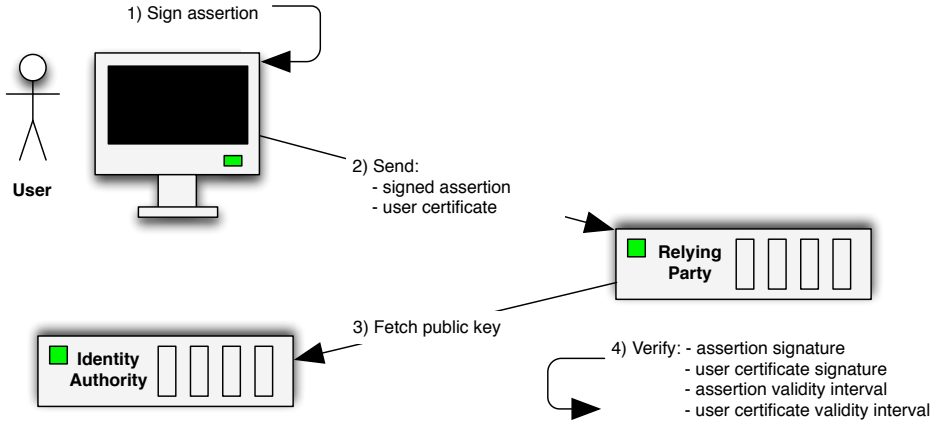


Figure 2: BrowserID assertion verification process. (Source: [17], used under CC-BY-SA 2.5)

key and that the owner of the key also controls the URI's content. Therefore, the identity is accepted as authentic.

The authentication process is executed through the existing client certificate verification mechanism present in TLS, the secure communication protocol that is a standard part of all modern web browsers (though perhaps better known by the name of its predecessor: SSL). This means that WebID works with most existing browser software—only the server ends needs to be modified, a much more manageable task.

In the past, the use of client certificates has mostly been confined to single organizations that wanted, for example, to limit site access to employees without resorting to passwords (or in addition to passwords). In this case, the certificates are digitally signed by the provider and can be verified when a user attempts to connect to the service over TLS. However, certificates can also be "self-signed" (that is, signed with the user's own private key instead of that of some central authority), which, in conjunction with some other mechanism to verify the authenticity of the signature, can be used to verify the identity of the certificate holder. In WebID, that other mechanism is the FOAF profile.

The sign-in flow can be seen in Figure 3, taken from the WebID draft specification [3]. (For this one section, we switch up names, following the conventions used in the diagram. Here, Bob is the user, trying to access a server belonging to Alice.) The following assumes that Bob has already created a WebID profile with some IdP and has provisioned a self-signed certificate to his browser keyring, with the public key stored in his WebID profile. These certificates, typically generated through a button on the profile host's site, include the WebID URI, as well as a public key. A typical sequence runs like this:

1) The user agent establishes a TLS connection with the server, which is also a WebID RP.
2) The application protocol exchange begins; in this case, the UA makes a request to access some resource on the server. A "Guard" intercepts this request and determines that authorization is required to access the resource.
3) If there is no active session between the two parties, the Guard requests that the client authenticate itself using a TLS client certificate. The UA asks Bob to select the certificate corresponding to the identity to use (unless Bob has previously selected a default for

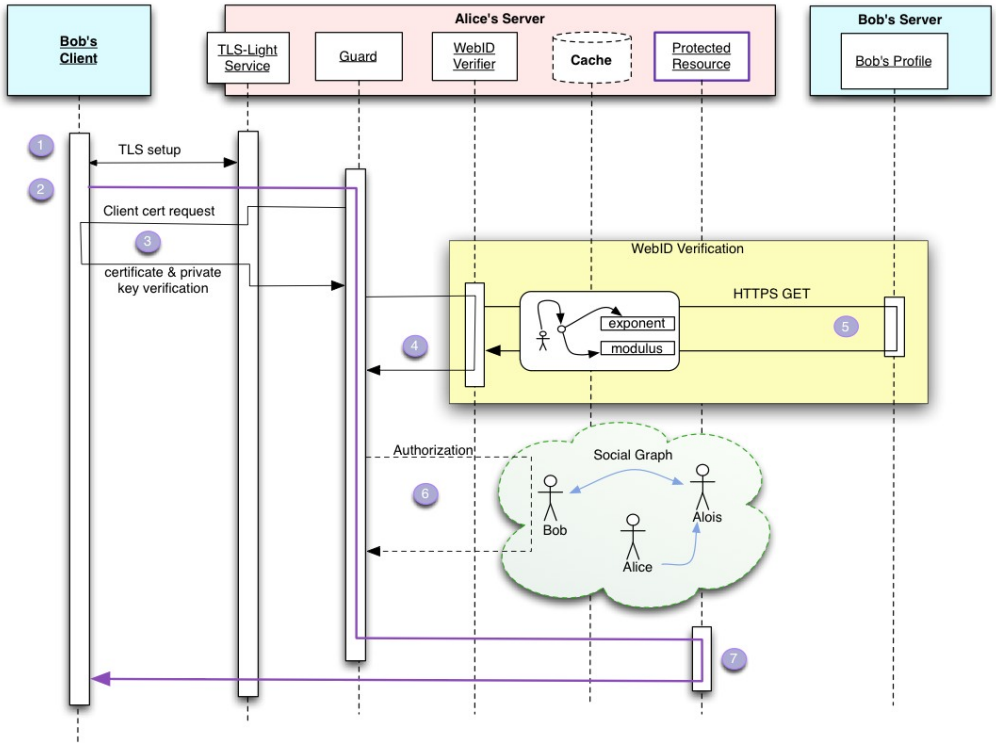


Figure 3: WebID authentication flow. (Source: [3])

the RP), and sends the certificate and a signed message to the RP. The server's TLS agent verifies the signatures on the certificate and message using the included public key, thus proving that the client is in possession of the corresponding private key.

4) The Guard extracts the WebID Profile URI from the certificate and asks the internal WebID Verifier to verify Bob's claim of ownership of the URI.
5) The Verifier dereferences the URI to obtain the profile document and checks that the public key in the profile matches the one in the certificate. If so, then the RP knows that Bob controls the given URI, and so can be identified by that string.
6) Having established Bob's identity, some other site-specific mechanism can be used to control the level of authorization granted to him, such as through the trust relationships encoded in a graph of relations (a *web of trust*), as shown here.
7) The response returned to the client depends on the level of access granted by the authorization query in the previous step.

Steps 1–3 are actually just the usual TLS mutual-authentication protocol; only the remaining steps are new to WebID. It is step 5 that links the current request to the identity URI, through the matching of the two halves of the asymmetric encryption key. Additionally, once the validity of the client's key pair has been established, it can be used to encrypt all further communications to the user agent over the TLS link, providing an additional measure of security.

VI. Analysis

In this section, we examine and compare the BrowserID and WebID proposals using six different criteria: (i) the consequences of the loss or compromise of an authentication key; (ii) resistence to phishing attacks; (iii) risks associated with recycling (reusing) identifiers; (iv) the extent to which the protocol protects user privacy and prevents tracking of activity; (v) how the system holds up in the face of a network outage or the temporary or permanent failure of an identity provider; (vi) and general usability.

A. Key Loss

Both of the proposals rely on the use of browser-held certificates to reduce the frequency with which a user needs to enter his or her credentials. And in both cases, since the certificates are stored in the browser or system keychain without any passphrase protection, simple possession of a certificate is sufficient to gain access to all of the accounts associated with the particular URI or email in the certificate. Consequently, the loss of a device with active and valid certificates represents a major security concern. However, device PINs and system passwords, with short auto-lock timeouts, could greatly mitigate the risk here (on devices owned by the user).

BrowserID

In order to try to reduce the risk associated with the loss of an identity certificate, BrowserID certificates have limited lifespans (from a few minutes to a few hours). However, there is no way to disable a certificate before it expires, so one is fairly helpless until that time period runs its course. Furthermore, the system supports automatic provisioning of certificates while an authenticated session with the IdP is active, and such a session may have a much longer lifetime, on the order of days or weeks. (Think of the session timeout of a typical webmail service, such as Gmail.)

There is a definite trade-off between security and convenience—a shorter session timeout reduces the risk of a thief obtaining an active session, but means the user will have to sign back in to their identity provider more often. Experience has shown that, given the choice, users will opt for convenience over security, and most will not want to have to sign in several times a day. Finding the right balance may present a significant challenge, and it may be wise to adjust the lifespan of certificates depending on the environment in which they are used (e.g. cell phone vs. desktop PC vs. public computer).

WebID

WebID public keys are stored in the user's profile, and retrieved from the profile by RPs during certificate verification. This means that a compromised private key can be disabled simply by removing its corresponding public key from the profile. Also, because the WebID profile-editing interface will typically be password-protected, the loss of a device does not necessarily compromise the identity. (Although, if the profile host allows password resets by email, all bets are off.)

B. Phishing

The ease with which users of a system can distinguish between a genuine and a fake credential prompt is key to preserving the security of their data. If users can be easily fooled into giving their access credentials away to the bad guys, no level of data protection is going to help.

The authentication flow of these systems will require a certain amount of adjustment for users, and it is important that all camps work to provide sufficient education to new users so that they understand the dangers and how to protect themselves. There is a risk that users, who are now conditioned to entering password information when signing into a site, will not realize that, in a federated system, they should never be giving out their identity password to any site except their IdP. As a result, some may be easily tricked into typing their username and password into sign-in forms on other sites. (This is one of the points we will be examining further in an upcoming user study, as described under Future Work.)

BrowserID

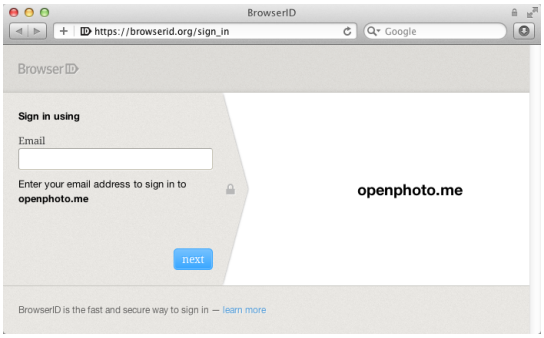


Figure 4: BrowserID sign-in dialog.

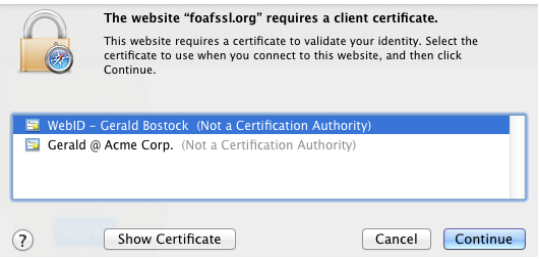
In order to bootstrap the system and generate sufficient user interest to drive RP adoption, the BrowserID user interface is initially being deployed as a JavaScript module, to be downloaded by the web browser as part of a site's pages. BrowserID windows (such as the one shown in Figure 4) will look like any browser windows and will require careful scrutiny to verify that they are genuine. The user's only means of detecting fakes will be the URL and lock icon in the address bar, which many users do not know how to interpret [19], or even bother to look at [20].

Additionally, it would hardly be surprising to see phishing versions of these windows simply hide the address bar, again hoping that most users will not notice its absence, or will not be technically sophisticated enough to realize the implications.

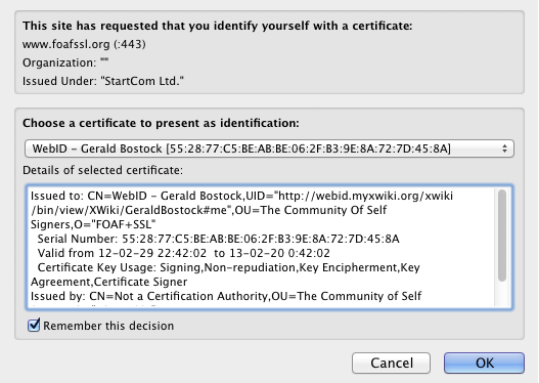
One can argue that this is a short-term issue, and that the goal (if BrowserID gains traction) is to have most browsers provide native support for the protocol. This would mean that the provisioning and sign-in steps can be presented in platform-specific windows that are more easily distinguishable from web page windows. However, again, it has been demonstrated that users will often not examine the window "chrome" that closely and can easily be fooled by web-page-generated displays that are designed to mimic the look of a system window on popular platforms (e.g. Windows) [21]. A more secure solution would be to adopt a technique such as that proposed by Sun et al. [10], where the desktop background is blacked out and the browser window shown shrunk down, behind the sign-in window. This is an effect that cannot be replicated by JavaScript code running in a browser window, and makes it very easy for the user to verify the authenticity of the window.

WebID

Certificate provisioning in WebID is usually a one-time activity per device, well separated from the RP sign-in process. When actually interacting with RPs, one uses the browser's own client certificate selection interface, which has a visually distinct appearance vis-a-vis browser content windows. (See Figure 5 for two examples. The user-friendliness of these interfaces will be discussed in sec-



(a)



(b)

Figure 5: WebID certificate selection: (a) Safari 5, (b) Firefox 10.

tion VI-F.)

With better browser integration for profile management, WebID users might never need to enter the password into a web form—all changes to their profile might be done through native platform windows with recognizable chrome. Nonetheless, education will still be required to get users to unlearn some habits of password usage.

C. Recycled Identifiers

A known issue with OpenID is that if a user abandons or loses control of his or her OpenID identifier, "a new registrant of that URL can gain access to the same private resources as the previous registrant" [22]. While the OpenID Authentication 2.0 specification addresses recycling of identifiers by OpenID providers (see [2, section 11.5.1]), this does not protect against the case where a user delegates a URI under his or her own control to another provider, and then gives up the domain from which the delegation occurred.

Unfortunately, as both BrowserID and WebID are based on non-permanent email addresses or URIs as identifiers, they are both prone to this issue as well.

BrowserID

With BrowserID, this recycling problem seems even more likely to occur because, as any longtime Internet user can attest, one's email address is likely to change several times: as one changes ISPs, switches to webmail, registers one's own domain, and so on. Depending on the email provider (including some high-profile sites such as Hotmail and Yahoo [23]), once an account becomes inactive, the address may become available to others. If an RP site records the email address as the sole means of identifying a user, anyone who claims the old address will be able to access any of the previous owner's data, without having to know the previous owner's password. A BrowserID RP currently has no way of knowing that the person behind the email address has changed.

Admittedly, this problem exists with traditional accounts as well—most sites permit an account's password to be reset through a link sent to the email address associated with an account. The difference is that, in many cases, the password reset procedure would require the intervention of a human (for example, to complete a CAPTCHA) and would be a different procedure on each site. In the BrowserID case, though, no reset process is required, making the process much faster and easy to automate. Additionally, the reduced friction in account creation means that it is likely that users will create many more accounts than before, many of which they will soon forget about.

While there is some discussion underway about creating a consistent process for changing the email address registered with an RP, poor Alice is still left to remember all of the places where she has created an account and visit each of those sites to make the switch. Any that she may forget (perhaps some site that she hasn't signed into for quite a while, but which might still have important information about her) could be targets for identity thieves who could use tools to scan for disused email accounts. And because the BrowserID IdP does not know which sites a user has signed into (see the subsection below on Privacy/Tracking), it cannot offer to track and automatically update RPs. However, the browser itself is able to track user activity, which, along with a syncing feature for multiple devices, could help the user manage such updates.

WebID

A similar problem can arise with WebID, particularly if one is using a third-party profile hosting service. WebID RPs identify users strictly by their profile URIs, and if Alice's profile host recycles her identifier after she closes her account, all Bob needs to do is create a new profile (with at least one public key) at the same URI to gain access to any web accounts still associated with that WebID. It is not necessary to have Alice's private key, because he has replaced the public keys in the profile with his own.

This issue is amplified if an organization is hosting its own profile server under its own domain, and for one reason or another lets the domain go. In that case, the domain will almost certainly be snapped up by someone else, who now effectively controls *all* of the identities formerly associated with the domain.

For both BrowserID and WebID, the use of permanent URNs (Uniform Resource Names) [24] or XRI (Extensible Resource Identifier) i-numbers [25], with a means of mapping from the human-friendly email address or URI, would allow users to change IdPs without the aforementioned issues with recycling of the canonical identifier. However, there remain questions regarding how to pass the permanent identifier, how to prove ownership, and the implications of an account compromise that taints a permanent identifier. For now, using a domain that is under one's control (and remains so for life) is probably the best defense against the risk of account hijacking through identifier recycling.

D. Privacy/Tracking

Users want control over the dissemination of their personal information and many are rankled by the insistence of nearly every web site to collect personal details there are not really necessary to use the service.

Equally important to many is the ability to maintain some level of anonymity online, at least some of the time. Very few people want to broadcast their every move online, yet that it is a very real danger with many federated identity systems. With systems such as Facebook Connect, every time you use your identity to sign into a web site, that activity is reported back to the identity provider (e.g. Facebook), who may very well make use of that data for marketing purposes, or share it with other users of their service.

BrowserID

BrowserID uses a mechanism where users can obtain a certificate from their IdP, and then subsequently use that certificate to generate a token that can be passed to an RP and verified by the RP using only the IdP's global public key. The benefit of this two-stage process is that Alice's identity provider is not informed about which sites she signs into and thus cannot track her activities. The only contact between a service provider and the identity provider is to request the latter's public keys, which reveals no information about the identity of the user for which the request is being made.

There is one caveat to this claim, however: At the time of writing, Mozilla was encouraging the use of a centralized verification service hosted at browserid.org [26], which would give Mozilla access to all sign-in requests that are verified through this service. Even with a stringent data privacy protection policy in place, some will view this as contrary to one of BrowserID's key principles (about not leaking tracking data back to identity authorities), and we worry that it sets a precedent that could lead to the creation of competing verification services, some of whom might try to monetize the data. In this situation, users would not be able to determine when this information was being shared with other parties; and in most cases, given BrowserID's marketing material, they might probably not even be aware that such sharing was taking place.

WebID

When using WebID, on the other hand, when Alice tries to sign into to a site, the RP must make a request to Alice's IdP to retrieve her profile, thereby giving the IdP enough information to connect Alice and the site. Of course, not all of the requests for profile information will correspond to sign-in requests—search engines may crawl the public portions of the profile, amid requests from individuals following social graphs, for example—but some simple rules and heuristics would likely be sufficient to fairly reliably distinguish between different types of traffic.

Nonetheless, the advocates of WebID argue that this sort of trackability is a *good* thing, as they are working under the assumption that users concerned about privacy would host their WebID profile only on a system that they themselves control (such as a server in one's home). In that case, having all identification requests recorded makes it much easier for a user to monitor for suspicious activity, such as sign-in requests to sites she has never visited, or activity at unusual times of the day. Software running on the user's profile server could easily alert the owner when it identifies some suspicious pattern of activity.

The counter-argument, of course, is that very few average users have the technical knowledge or desire to set up and run their own servers, and many will not have the ability to easily do so, due to terms-of-use limitations in the ISP contracts. As a result, it would seem that widespread mainstream adoption of WebID is likely going to have to rely on hosting service providers, and in that case, the trackability baked into the WebID protocol can be seen as a potential privacy concern.

E. Robustness

What happens if a identity provider goes down, even for a short time? What if it goes down for good? Suddenly, you are cut off from not just one account, but *all* of your accounts at once. This will obviously be a serious concern for potential users of these systems.

BrowserID

Mozilla is in the process of putting into place the capacity and redundancy to support a massive level of usage of their secondary authority service, intended as a temporary measure to fill the gap until various email providers have the support in place to vouch for their own users (called a *primary authority*). However, it is unlikely that smaller primaries are going to be able to scale to the same level of service, so larger providers may have a substantial advantage in this critical area.

The two-stage provisioning system used by BrowserID also provides a significant benefit here in that only the primary's public key is required to verify the signature on a BrowserID identity certificate. And since those public keys are relatively static and few in number (compared to the

	BrowserID	WebID
Key loss	Short lifetimes limit window of opportunity; should be protected with device PIN/password.	Keys long-lived, but easy to disable from password-protected profile, even if device is lost/stolen.
Phishing	Requires careful scrutiny of address bar and security icons. Fairly easy to spoof until browsers support natively.	Uses browser's client cert selection UI; low risk of phishing. No danger in giving out cert to wrong site.
Recycled identifiers	Both systems allow access to a previous owner's accounts if email/URI is given up.	
Privacy	IdP does not know where identity has been used—not trackable.	IdP knows every site on which identity was used—allows tracking. (Could be positive, if self-hosting one's profile.)
Robustness	Cached certificates and IdP public keys may permit the system to work for a while, even if the primary is unreachable.	Profile must be accessible to verify a certificate, so accounts cannot be accessed while profile host is unreachable.
Usability	Much attention being paid good flow and ease of understanding.	Uses inconsistent and often cryptic client certificate selection UI.

Table I: Comparison: BrowserID vs. WebID

number of users they serve), they can be easily cached. As a result, a user may still be able to sign in even if her identity provider is temporarily unreachable, provided the IdP's public keys are in the RP's cache.

WebID

In the WebID case, an RP must be able to access the user's profile in order to verify the signature on the user's identity certificate. Thus, even though she has a valid identity certificate cached in her browser, it would not be possible for Alice to sign in to any sites while the profile host was unreachable from the RP site.

There is, however, a mechanism in the WebID specification to allow a certificate to contain multiple WebID URIs, which would allow redundant profile hosts to be used. Unfortunately, the current specification also introduces a security hole that would permit an attacker to easily hijack someone's accounts by creating a certificate that points to both the original site and a second site that the attacker controls. This needs to be addressed by having the specification spell out how service providers should correctly deal with multiple WebIDs.

F. Usability

In truth, the lack of uptake with security and privacy tools, such as PKI, stems not from questions about their security or utility, but from the perceived complexity from the perspective of the average user [27]. Regardless of the potential benefits, it is user experience that is going to make or break any new security-related technology.

BrowserID

The Mozilla Identity Team recognizes that they are going to have to provide an experience that is as good or better than that provided by the likes of Facebook if they are going to woo users away to their system. As such, they are putting a great deal of their efforts into getting the BrowserID user experience right, making it as smooth and simple as possible. This is still a rapidly evolving system, and they are still making major changes, but they appear to already be well ahead of WebID in terms of usability. (Compare Figure 4 with Figure 5, for starters.)

WebID

WebID takes advantage of a feature that is already built into virtually all modern web browsers: support for client certificates. Unfortunately, the user interface for this feature varies widely between browsers, and in many cases is cryptic and ugly, overloading the user with technical details that few understand. So, although the support is technically present, WebID's developers are entirely dependent on browser vendors improving the usability of this interface, which in the past has been considered an advanced feature, used only in private corporate networks, with support from the local IT department. On the other hand, the amount of work required to redesign this one dialog should be significantly less than will be required to integrate native support for BrowserID, and developers can draw on the usability research being done by the BrowserID team and others.

VII. Conclusion and Future Work

A summary of the above analysis is given in Table I. We have noted a number of security and usability issues that remain in both of the authentication systems examined in this paper, although BrowserID, in particular, seems to made significant improvements in usability over OpenID. Additionally, its privacy focus may have greater appeal as online privacy awareness and advocacy increases. Concerns about risks related to key loss and spoofing attacks remain, but native browser support will help with the latter issue.

WebID's approach trades privacy for security; placing public keys in a public profile makes it is much easier to deactivate a compromised identity certificate, but it requires that relying parties contact the server to verify certificates, leaking information about the identity owner's activities to the identity provider. WebID's choice to base their system on SSL client certificates allows their system to work on nearly

all browsers immediately, but the user interfaces for certificate selection are inconsistent and much less straightforward for users than BrowserID's custom tailored solution.

Our future work in this area includes user studies to test the usability of these two systems, and to see how well users of various levels of technical background can comprehend the security and privacy trade-offs in these systems. We will also be closely monitoring the development of both platforms during this critical pre-launch period, and on into wider deployment to see whether they will have a greater impact than did their predecessors.

ACKNOWLEDGEMENTS

The authors would like to thank the members of the BrowserID and WebID teams who answered many of our questions and offered their feedback on this paper, with particular thanks to Lloyd Hilaiel and Dominik Tomaszuk for their reviews of a draft of this paper.

REFERENCES

[1] D. Florêncio and C. Herley, "A large-scale study of web password habits," in *Proc. 16th Int'l Conf. World Wide Web* (WWW '07). ACM, 2007, pp. 657–666.

[2] *OpenID Authentication 2.0*, OpenID Foundation, Dec. 2007; openid.net/specs/openid-authentication-2_0.html.

[3] M. Sporny, T. Inkster, H. Story, B. Harbulot, and R. Bachmann-Gmür, *WebID 1.0 - Web Identification and Discovery*, World Wide Web Consortium (W3C) editor's draft, work in progress, Dec. 2011; www.w3.org/2005/Incubator/webid/spec/drafts/ED-webid-20111212.

[4] H. Story, B. Harbulot, I. Jacobi, and M. Jones, "FOAF+SSL: RESTful authentication for the social web," in *Proc. European Semantic Web Conf.*, 2009.

[5] *BrowserID Specification (Draft)*, Mozilla Foundation draft, Feb. 2012; wiki.mozilla.org/Identity/BrowserID.

[6] L. Hilaiel, "How BrowserID works," blog, Jul. 2011; lloyd.io/how-browserid-works.

[7] E. Maler and D. Reed, "The venn of identity: Options and issues in federated identity management," *IEEE Security & Privacy*, vol. 6, no. 2, pp. 16–23, Apr. 2008.

[8] R. Housley, W. Ford, T. Polk, and D. Solo, *Internet X.509 Public Key Infrastructure—Certificate and CRL Profile*, IETF RFC 2459, Jan. 1999; www.ietf.org/rfc/rfc2459.

[9] S. Sun, Y. Boshmaf, K. Hawkey, and K. Beznosov, "A billion keys, but few locks: The crisis of web single Sign-On," in *Proc. New Security Paradigms Workshop*, Jul. 2010.

[10] S. Sun, E. Pospisil, I. Muslukhov, N. Dindar, K. Hawkey, and K. Beznosov, "What makes users refuse web single Sign-On? an empirical investigation of OpenID," in *Proc. Symp. on Usable Privacy and Security* (SOUPS), 2011.

[11] D. Chappell, "Introducing windows CardSpace," Apr. 2006; msdn.microsoft.com/en-us/library/aa480189.aspx.

[12] V. Bertocci, G. Serack, and C. Baker, *Understanding Windows CardSpace*. Upper Saddle River, NJ: Addison-Wesley, Dec. 2007.

[13] M. Jones, "Personal reflections on the CardSpace journey," blog, Feb. 2011; self-issued.info/?p=458.

[14] Microsoft Identity and Access Team, "Beyond windows CardSpace," blog, Feb. 2011; blogs.msdn.com/b/card/archive/2011/02/15/beyond-windows-cardspace.aspx.

[15] P. Trevithick, "Identity in the browser at 5. lessons learned." blog, May 2011; www.incontextblog.com/?p=728.

[16] *Verified Email Protocol*, Mozilla Foundation, Jul. 2011; wiki.mozilla.org/Labs/Identity/VerifiedEmailProtocol.

[17] Mozilla Foundation, "BrowserID protocol overview," Jan. 2012; developer.mozilla.org/en/BrowserID/Protocol_Overview.

[18] D. Brickley and L. Miller, *FOAF Vocabulary Specification*, The FOAF Project, Aug. 2010; xmlns.com/foaf/spec/.

[19] R. Dhamija, J. D. Tygar, and M. Hearst, "Why phishing works," in *Proc. SIGCHI Conf. on Human Factors in Computing Systems* (CHI '06). ACM, 2006, pp. 581–590.

[20] T. Whalen and K. M. Inkpen, "Gathering evidence: use of visual security cues in web browsers," in *Proc. of Graphics Interface 2005*. Canadian Human-Computer Communications Society, 2005, pp. 137–144.

[21] D. Sharek, C. Swofford, and M. Wogalter, "Failure to recognize fake internet popup warning messages," *Proc. Human Factors and Ergonomics Society Ann. Meeting*, vol. 52, no. 6, pp. 557–560, Sep. 2008.

[22] D. Reed, L. Chasen, and W. Tan, "OpenID identity discovery with XRI and XRDS," in *Proc. 7th Symp. Identity and Trust on the Internet* (IDtrust '08). ACM, 2008, pp. 19–25.

[23] Buzz Web Tips, "They recycled your deleted email account? oh the risks!" blog, Aug. 2011; www.buzzwebtips.com/archives/165.

[24] R. Moats, *URN Syntax*, Internet Engineering Task Force Proposed Standard RFC 2141, May 1997; tools.ietf.org/html/rfc2141.

[25] OASIS XRI Technical Committee, *Extensible Resource Identifier (XRI) Syntax V2.0*, OASIS Committee Specification xri-syntax-V2.0-cs, Nov. 2005; www.oasis-open.org/committees/download.php/15377/xri-syntax-V2.0-cs.pdf.

[26] Mozilla, "How to use BrowserID on your site," Jan. 2012; github.com/mozilla/browserid.

[27] A. Whitten and J. D. Tygar, "Why johnny can't encrypt: a usability evaluation of PGP 5.0," in *Proc. 8th Conf. USENIX Security Symp.* USENIX Association, 1999.