RZ 3412 (#93644) 06/10/02
Computer Science 11 pages

Research Report

Privacy in Browser-Based Attribute Exchange

Birgit Pfitzmann, Michael Waidner

IBM Research
Zurich Research Laboratory
Säumerstrasse 4
CH-8803 Rüschlikon, Switzerland
{bpf,wmi}@zurich.ibm.com
http://www.research.ibm.com/privacy

Privacy in Browser-Based Attribute Exchange

Birgit Pfitzmann, Michael Waidner
{bpf,wmi}@zurich.ibm.com, IBM Zurich Research Lab
June 10, 2002

Abstract

Browser-based attribute exchange means protocols for a user of a normal web browser to send attributes, such as authentication or demographic data, to a web site. The best-known deployed protocol of this type in the real world is Microsoft's Passport. We identify the privacy requirements on such protocols in a general consumer scenario, derive the main design decisions needed to fulfil these requirements, and present a protocol with these properties. Our emphasis lies on protocols that could be standardized and deployed short-term.

1 Introduction

Electronic commerce with end consumers has not fulfilled the high expectations of a few years ago. There are many possible explanations to that. By current consumer surveys, lack of trust, in particular with respect to privacy (here including protection from spam mail and unwanted phone calls) is the main reason for reticence. A second reason is that many services are not easy enough to use. A conclusion often drawn from this second observation is that there should be one-click or single signon features across the entire world-wide web. However, unless designed very carefully, such features can easily make the situation for trust and privacy even worse and thus be counterproductive.

Classically, single signon refers to authentication. However, in an ecommerce scenario, authentication is not of primary importance, while obtaining payment and shipping data for a transaction is, and sites are typically interested in additional demographic or usage data. Hence we will not talk of single signon, but of attribute exchange. The next market observation currently taken for granted is that a large segment of users is not willing to install additional software for electronic commerce, neither for ease of use nor for privacy. Thus attribute-exchange protocols must work for this user segment, i.e., for people using nothing but a commercial browser. We call such protocols browser-based, and the feature zero-footprint. Another currently made precondition is mobility in the sense that a person should be able to use the protocols from varying browsers, such as several personal devices or even Internet kiosks. While we must stress that single signon from unknown browsers is dangerous for security, nobody is forced to do that, and thus we simply accept that this is a market requirement.

The operational requirements we just listed exclude most existing solutions. In particular, classical form fillers rely on proxies to capture forms that query for data and try to fill them, e.g., see [Robo_99, Gato_99]. The proxies may be local or remote, i.e., on the user's machine or at a third party. While the latter is not far from the zero-footprint and mobility feature (the user only has to set the proxy), we will see that a remote proxy also has privacy disadvantages. Besides, the current goal with attribute-exchange protocols is to make the designers' life much easier by agreeing on a standard for attribute queries, while form fillers had the difficult task to analyze arbitrary forms for known attributes. Similarly, classical wallet solutions were designed either as proxies or as browser plug-ins on user machines [IBM_97, Pass_99, Zero_99]. For mere authentication, digital signatures [DiHe_76] are the technical single signon choice, but even if the corresponding PKIs were successful, signatures would not go far in our scenario because the main issue is the attribute exchange. Attributes can be transmitted in attribute certificates, e.g., as in [PKIX_01], but then still an actual exchange protocol has to be designed, i.e., in particular for transmitting such certificates with unmodified browsers. Furthermore, most of the attributes needed in general commerce do not need certification (e.g., the shipping address and user preferences).

From the long-term research side, unlinkable credentials are a related topic, as invented in [Chau_85] and currently developed furthest in [CaLy_01]. Once again, however, such protocols need a browser proxy. Moreover, as they are the privacy-enhanced version of attribute certificates, they are overkill for most of the first intended applications of browser-based attribute exchange: Most attributes are either identifying (like a shipping address) or do not need certification (like user preferences). Even payment information, which would and could theoretically need privacy and certification, is currently simply not realistic in that form. In this paper, we look at the simpler cases. Clearly, there are limits to our protocols that only unlinkable credentials and anonymous payments could solve.

This leaves us with a protocol class of which Microsoft Passport is the best-known representative (not fully published, but see [Micr_99]). Published efforts in the same design space, and with more flexible features, are Shibboleth and SAML [Shib_01, SAML_02]. A similar proposal, but only for authentication, is [Gola_01]. Another well-known player is the Liberty Alliance [Libe_01], but without detailed requirements or protocols so far. SAML (Security Assertions Markup Language) is primarily a message format for authentication and attribute assertions (and also authorization assertions), and for queries and responses about them. It allows so-called profiles, which correspond to entire attribute-exchange protocols using such messages. However, the only profiles standardized so far are core parts for single signon. Shibboleth, developed in interaction with SAML specifically for a university network, can be seen as a quite powerful additional profile if one extracts its protocol aspects. While certain privacy aspects are lacking, our proposals are quite close to it, and the quickest way to standardize them would also be as SAML profiles.

In the following, we first consider the privacy requirements that one should make for browser-based attribute-exchange protocols. We then derive individual design decisions that follow from these requirements. Finally we show that protocols exist that combine all these design decisions without significant loss in efficiency or ease of use.

2 Browser-Based Attribute Exchange

Before the discussion of privacy, we briefly summarize the functional requirements on browser-based attribute exchange protocols, sketch the structure of such protocols, and summarize security requirements and limitations.

2.1 Functional Requirements

Browser-based attribute-exchange protocols should have the following functionality, as already mentioned in the introduction:

- **Case-by-case attribute exchange:** The main functionality is that a web site wants to get information about a browsing user; the information items are called attributes. The attribute exchange should happen on a case-by-case basis. This corresponds to a cross-trust-domain scenario, where the different sites a user visits do not all need access to all attributes that some site knows about this user. (In contrast, in in-enterprise single signon solutions, all participating sites can simply use the same user registry, once a user has authenticated at a fixed entry point.)
- **Browser-based:** The protocols should be usable from all common web browsers on all common machines.
- **Zero-footprint version:** The protocols must even be usable if a user has no software specifically for attribute exchange besides the browser, and has not enabled any active content such as JavaScript, ActiveX and Java on the browser, e.g., for security reasons. They should even work for users who switch cookies off. (Implementation variants that exploit active content or cookies when available are of course allowed.)
- **Mobility:** In this context, mobility means that the user moves from device to device, e.g., by using Internet kiosks. In particular, the protocols can therefore not rely on browser personalization features.
- **Flexibility:** In addition, we would like the protocols to be flexible. In particular, they should allow multiple sites with attributes per user (which may or may not know of each other); user-chosen and certified attributes; and general-consumer and closed-group usage; all in one protocol superset.

2.2 Rough Overview of Protocols

In all known browser-based attribute-exchange protocols, a site asking for attributes redirects the browser to another site that holds those attributes. We call them destination site and wallet holder, respectively, and say wallet for the attribute collection of one user at one wallet holder. The wallet holder authenticates the user and redirects the browser back to the destination site with the attributes. Usually, the query string of the URL is used for the attribute transport, because it survives the redirect, in contrast to the message body or a cookie. A POST is also possible, but needs an additional user click or active content. If the attributes are long, the query string may only contain a handle to them, while the actual attributes are transported directly.

This scenario is shown in Figure 1. Clearly, in the zero-footprint mobile version, a person’s wallet holder must be another party. We call this a remote wallet. The security- or privacy-critical messages through the browser will be protected by SSL/TLS, a secure-channel protocol existing in all common browsers. If any authentication (under a name or a previously introduced pseudonym) is intended, the wallet holder will authenticate to the destination site; this is indicated by the seal

"W". Authentication of the destination site, indicated by the seal "D" is needed if privacy only allows the release of certain attributes to certain destination sites.

A detailed protocol, then with privacy features, is presented in Section 5.

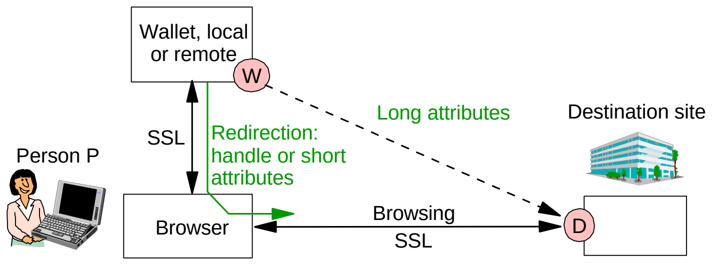


Figure 1 *Rough overview of a browser-based attribute-exchange protocol. The dashed direct contact between wallet holder and destination site is optional.*

2.3 Security Requirements and Limitations

Although we concentrate on privacy, it is useful to have the security requirements in the back of one's mind. Moreover, there are clear security limitations for browser-based attribute-exchange protocols, which should never be forgotten. The requirements are:

- **Authenticity**: If any authentication (under a name or a previously introduced pseudonym) is intended, only the correct user should be able to authenticate. In particular, man-in-the-middle attacks must be excluded.
- **Correctness of certified attributes:** If some attributes are confirmed by the wallet holder or another party, these confirmations should be unforgeable.
- **Freshness:** Where desired, it should be possible to guarantee freshness of attribute confirmations.

Accountability of the user, in particular non-repudiation, is currently not a goal. (The zero-footprint and mobility restriction would also not allow it.) This also implies that accountability of the wallet holder cannot be a goal at present. Availability of the wallets is very important; however, it is more an issue of implementation than of the protocols.

The main security limitations are the following:

- **Password security:** All authenticity finally relies on the wallet holder reliably authenticating the user before sending attributes. This is typically done with a password, called single signon password, and for users choosing the mobility option there is no real alternative. Unless the wallet is local, this makes the password available for online guessing. By trying this for enough users, an attacker is almost sure to be able to impersonate some users to their wallet holders, and thus to all destination sites using this protocol.
- **Browser and OS security:** A browser-based protocol cannot be more secure than the browser, at least in the zero-footprint version, nor than the underlying operating system. In particular, a single signon password passes through the browser and operating system. This is particularly dangerous for the mobile scenario, where the user uses unknown machines.

Other dangers are that a user is tricked into divulging the password, registration errors, and the usage of cookies, in the order of increasing amenability to design. Many details can be found in [KoRu_00, Slem_01] (see also [FSSF_01], but that only treats direct authentication).

3 Privacy Requirements

In this section, we describe the privacy requirements on browser-based attribute-exchange protocols. We concentrate on the use in a general consumer scenario, although the same protocols can also be used in more closed environments, where certain requirements can be relaxed.

3.1 Requirements

Concrete privacy requirements can systematically be derived from the following generally accepted principle:

> **Privacy principle:** Privacy is "the right of individuals to determine for themselves when, how, and to what extent information about them is communicated to others" [West_67].

Nowadays one also accepts purpose-binding of the information; that "determining" means at the minimum a visible policy of the "others" with opt-out choice for the individual; and that law may allow exceptions. The first consequence for attribute-exchange protocols is clear and universally accepted for a general consumer scenario:

1. **Standard attribute privacy:** A wallet *W* should only send attributes it explicitly holds about a person *P* to a destination site *D* if *P* consented to that.

Other consequences are more easily forgotten:

2. **Multiple roles:** The principle also applies to "identifiers" of *P*. For instance, even an identifier UID agreed upon between a wallet holder and a person *P*, with no prior real-world meaning, should not be sent to all destination sites automatically. In other words, a person should be allowed multiple roles (and not only in the user-unfriendly way of multiple wallets). It is another question which destination sites accept users without an identification under a real-world name, and for which purposes. But it should at least be accepted where one currently browses without any identification, and the sites can profit from an attribute-exchange protocol by asking for non-identifying demographic or preference data.
3. **Indirect information:** The principle also applies to indirect information, such as the person's location or visited-URL trails.
   - Clearly, wallet holders should not be allowed to mine or forward such indirect information without consent.
   - Wallet holders should not even *get* such information without consent or a technical need. It seems unavoidable that a remote wallet holder obtains the identities of the destination sites a user *P* interacts with, both for the redirect back and for security against man-in-the-middle attacks. However, it is not necessary that the wallet holder obtains the exact URLs of the pages a user visits.
4. **Privacy from wallet holders:** The principle also applies to the wallet holders as "others". (A special case was already discussed under 3.) That is, no person *P* should be forced to trust a wallet holder *W* with all their attributes. In particular, *P* should not have to trust one specific *W*, but actually, not even *any* wallet holder. In other words, people should have the choice among multiple remote wallet holders, and be allowed to use local wallets instead. The latter is substantiated further in Section 3.2.

Clearly, the zero-footprint option and the local-wallet option are mutually exclusive; a local-wallet user has at least the additional inconvenience to install and set up the local wallet. However, no additional inconvenience should result, so as not to exert an unreasonable price for privacy.

3.2 More on Trust in Wallet Holders

The argument about the need to allow local wallets may need more corroboration. (But recall that they are intended as an option, not as the only case.) One often hears three types of counter-claims, but they are not convincing. Privacy is not the only concern here: a wallet holder can also impersonate a user, which is technically more like a power of attorney than an ID card or other typical analogs. Being *forced* to allow someone impersonation would even essentially correspond to being put under guardianship.

No perfect security: The first claim is that nobody would be forced to blindly *trust* a wallet holder, because wallet holders would handle the data in a secure environment, possibly even in tamper-resistant hardware (at least whenever it is unencrypted). However, while a secure environment is certainly important, no hardware that can withstand a determined attack by a wallet holder (necessarily a rich and technology-informed organization) exists at present, and probably never will. Thus indeed one has to trust at least the organization as such. Furthermore even in the financial sector, insider-fraud is not rare, which shows both that criminal energy exists in such places and that current actual data-handling procedures are not always successful against it.

Trust is rare: The second claim is that nobody would be *forced* to trust a wallet holder, because almost everybody would trust one by themselves. However, consumer surveys show that this is not at all the case. For instance, in a recent Gartner survey where people were asked how much they trusted different types of organizations with (rather uncritical) data like credit card number and address on a scale from 1 (low trust) to 7, no organization type got over 5 on average. Privacy surveys (e.g., [IBM_99, Harr_02]) show consistently that 80 to 90% of all people are concerned about privacy, and that 25% are willing to get it even at a considerable price (in money or inconvenience). It is also known that about half of all people at least occasionally give wrong data to web sites. Specifically for browser-based attribute exchange, Gartner recently found that less than 10 percent of online consumers would be willing to exchange personal information in order to use personalized Web services, and that only 2 percent of consumers said they chose Passport for the convenience of single sign-in compared with 16 percent six months earlier [Wilc_02].

Use may become mandatory: The third claim is that nobody would be forced to trust a *wallet holder*, because nobody would be forced to use browser-based attribute exchange. However, even one quasi-monopoly provider of an important product or service can enforce such usage. While users may then restrict potential privacy violations by minimum usage of the protocol, they automatically become vulnerable to impersonation at all other sites that accept this protocol for real-world identifiers like names or email addresses.

4 Design Decisions for Privacy

We now derive individual design decisions that follow from the requirements made in Section 3.1.

4.1 Consequences of Attribute Privacy

First we consider how to realize the standard attribute privacy requirement.

1. **Flexible privacy policies:** Clearly, a person should be given flexible choices as to which destination sites are allowed to get which attributes. Proposals for such policy languages and administration systems have, e.g., been made in [APPE_01, BLKS_01]. Not all details need to be standardized with the exchange protocol, because many are only between each person and their wallet holder or manufacturer.
2. **Authenticate destination site:** By Decision 1, a privacy policy should allow the release of certain attributes to some destination sites and not to others. Hence the destination sites must be authenticated in the exchange protocol. A minimum is authentication by name, e.g., with an SSL server certificate. For elegant privacy policies, one will even need an infrastructure for attribute-based authentication, e.g., to recognize a destination site as a bank, a doctor, or a site with a certain privacy seal, but we do not see this happening for short-term deployment.

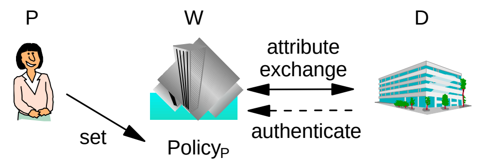


Figure 2 *Design Decisions 1 and 2: Allowing a person P to set a policy, and authenticating destination sites to apply the policy.*

3. **Integrate real-time release:** Policies are complicated. At least in the short run, we therefore expect that most people would not set a privacy policy in advance that covers a majority of usages. Therefore real-time release of attributes in the browser-based exchange protocol will play an important role. This means that the protocol allows the wallet to ask the user, via the browser, whether it allows certain attributes, e.g., a home address and a social security number, to be released to a certain destination site right now.
4. **Integrate policy data in request-response:** In privacy policies, a release is typically not unconditional, but for specific purposes and with certain obligations such as future deletion. A wallet and a destination site must negotiate this in some way. The easiest way is to allow the inclusion of privacy promises in attribute requests, and of privacy restrictions in the responses.

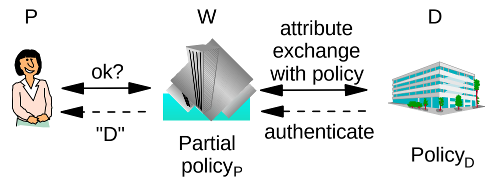


Figure 3 *Design Decisions 3 and 4: Asking the person P for her ok to release certain data to D, and exchanging policy aspects together with the attributes.*

4.2 Consequences of Multiple Roles, Indirect Information, and Local Wallets

Now we consider the consequences of the remaining requirements. In particular, we have to design the protocols such that users who choose local wallets get the full benefit of them. This means that the wallet can act independently (without allowing another party to impersonate it) and without leaking unnecessary information.

5. **Minimum information by URLs:** To prevent remote wallet holders from getting precise trails of a user's browsing behavior, destination sites should use fixed URLs for the second redirection, fixed names, and random session IDs in the protocol. (In closed environments, this can be given up without endangering interoperability.)
6. **Allow public-key cryptography:** The protocol must support public-key cryptography, because local wallets cannot a priori exchange symmetric keys with all potential destination sites, and using a key- or ticket-distribution center (e.g., as in Kerberos) allows the center to impersonate the wallet and gives it an unnecessary usage trail. We recommend that remote wallet holders also use public-key systems, so that they can also act independently.
7. **Allow nameless statements:** It must be possible to transfer demographic data about an unidentified user. First this means that the statement format for transferring attributes should not require a fixed user identifier. (This is invisible in the protocol design as such; but one has to avoid unnecessary naming requirements in standards.) Additionally, if a local wallet is used and authentication in a role is desired, the wallet's role key must not identify the wallet or its user. In other words, a wallet must be allowed different, unlinkable keys for a user's different roles.
8. **Local-wallet compatible certificates:** First, Decision 7 implies that self-certificates (or no certificates at all) must be allowed for role keys of local wallets. Secondly, for named statements, local wallets need rooted certificates. This means that the protocols must allow at least two-level certificate chains, or generally one level more than without local wallets. Typical protocols do not fix this anyway, but it may be important when deciding what statements can be transferred in a query string.
9. **Allow relative addressing:** A local wallet, and thus its user, can also be identified by its address. We saw in Section 2.2 that all known protocols need a redirect to a wallet. We must allow this to be to a path at localhost. (Once again, this just means that a standard should not make unnecessary requirements on the address format; redirects automatically also work for such addresses.)
10. **Direct connections opened by wallet:** In protocols where attributes are transferred directly between the wallet and the destination site, typically the destination site contacts the wallet [SAML_02, Shib_01], and thus needs an absolute wallet address. This should be turned around, i.e., the wallet should get an address of the destination site in the first redirect and open the direct connection.

Hiding the wallet address (Decisions 9 and 10) in this protocol layer is useful even though lower layers may identify a user: First, lower-layer information may be concentrated in fewer modules in the destination site and is thus less likely to be divulged accidentally. Secondly, if the lower layers provide some anonymization (from simple dynamic addresses assigned by the IP provider to complicated mixes [Chau_81]), anonymous addressing is not always implemented, while anonymous responses over a given connection usually are.

5 A Protocol with All these Properties

So far, no protocol has been proposed that fulfils all our requirements. While even Microsoft endorses many such requirements in press releases, no sketch of a compatible federated protocol has been presented yet. Hence Shibboleth is the only detailed, flexible protocol we can consider. It is actually quite well designed for privacy, although a bit special in places given its specific university purposes (e.g., with specific URL-based privacy policies). In particular, real-time release has been considered, although with a message flow with unnecessarily many steps. For our more general purpose, mainly local wallets are missing, and thus Design Decisions 7-10.

5.1 Overview

A protocol that combines all our design decisions is shown in Figure 4. To allow for long attributes, this protocol transfers the actual attributes directly. (In [HTTP_99], URLs are only considered safe up to 255 bytes, which is quickly exceeded for attributes with signatures.)

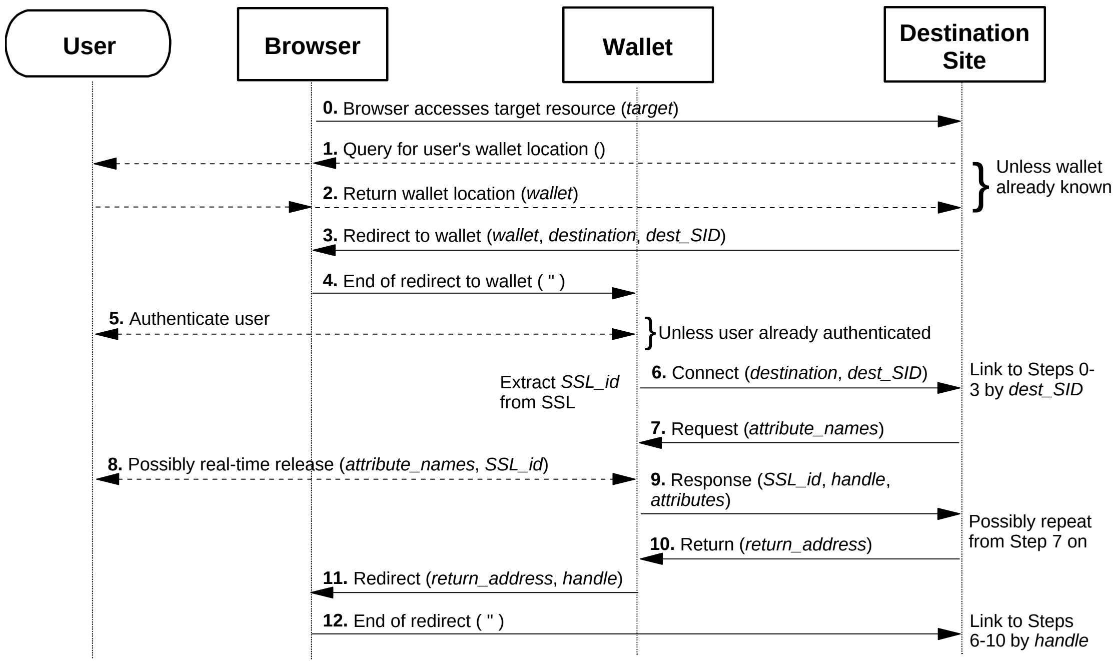
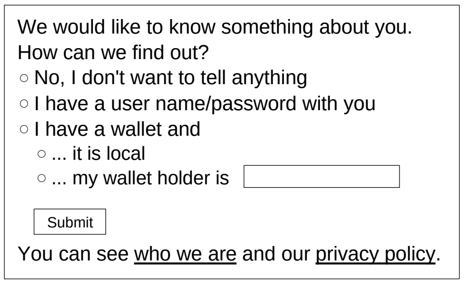

User
Browser
Wallet
Destination Site
0. Browser accesses target resource (target)
1. Query for user's wallet location ()
2. Return wallet location (wallet)
Unless wallet already known
3. Redirect to wallet (wallet, destination, dest_SID)
4. End of redirect to wallet (")
5. Authenticate user
Unless user already authenticated
Extract SSL_id from SSL
6. Connect (destination, dest_SID)
Link to Steps 0-3 by dest_SID
7. Request (attribute_names)
8. Possibly real-time release (attribute_names, SSL_id)
9. Response (SSL_id, handle, attributes)
Possibly repeat from Step 7 on
10. Return (return_address)
11. Redirect (return_address, handle)
12. End of redirect (")
Link to Steps 6-10 by handle

Figure 4 *Privacy-enhanced browser-based attribute exchange. Steps with dashed lines are only needed in certain cases.*

The basic structure is as described in Section 2.2: In Steps 3-4, the destination site redirects the browser to the wallet; in Step 5, the wallet authenticates the user; in Step 9, the wallet sends the attributes directly to the destination site; and in Steps 11-12 the wallet redirects the browser back to the destination site.

5.2 The Protocol Steps

Steps 1-2, a query and response for the user's wallet location, are necessary for allowing the user free choice among multiple remote wallet holders. In particular in the mobility case, neither the destination site nor the browser know anything about the user at this time, and thus the user must be asked. A possible form is shown in Figure 5. The radio button "it is local" leads to a redirect to localhost (Decision 9).



We would like to know something about you.
How can we find out?

○ No, I don't want to tell anything
○ I have a user name/password with you
○ I have a wallet and
  ○ ... it is local
  ○ ... my wallet holder is

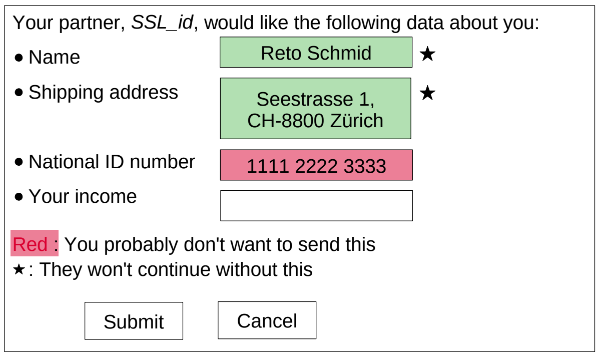
You can see who we are and our privacy policy.

Figure 5 *Example of the form for a wallet location query (Step 1)*

The redirect in Steps 3-4 uses the address *wallet* that was obtained. It transports a contact address *destination* and a session identifier *dest_SID* in its query string. By Decision 5, we do not typically transport the user's original URL *target* here as in some other protocols because usually the wallet need not know it.

Upon receiving such a request, the wallet first authenticates the user of this browser (Step 5). Then (Step 6) it contacts the destination site at the given address, including *dest_SID* so that the destination site can look up *target* and other parameters on which the attribute request may depend. This step must build up a secure channel, to guarantee privacy against outsiders for the following attribute request and response, and to ensure authentication of the destination site (Decision 2). We assume SSL/TLS for this, and let *SSL_id* be the name in the server certificate that the destination site uses.

The destination site sends its request over this secure channel in Step 7. The request format must be standardized at least for the general consumer scenario; a possibility is to use P3P and its base data schema [P3P_02] directly as a query language (recall Decision 4). The wallet may happen to know all attributes that the destination site would like, and also have a privacy policy (Decision 1) that allows their release to this site. Otherwise Step 8, the real-time release, is necessary (Decision 3). A simple form for such a release is shown in Figure 6. Where the wallet knows the data, it distinguishes whether the user has already pre-authorized the release in a policy (green, vertical stripes) and where he has not (red, diagonal stripes); in our example the name and shipping address are pre-authorized, while the ID number is not. All fields are editable so that the user is not bound to a-priori choices. In our example, a user might delete the ID number and not add an income, because the fields are not mandatory.



Your partner, *SSL_id*, would like the following data about you:

• Name	Reto Schmid	★
• Shipping address	Seestrasse 1, CH-8800 Zürich	★
• National ID number	1111 2222 3333	
• Your income		

Red : You probably don't want to send this
★: They won't continue without this

Figure 6 *Example form for a real-time release (Step 8).*

In Step 9, the wallet sends a response over the secure channel. Here Decisions 6-8 come in, i.e., if the response needs a signature, public-key signatures should be allowed, there should be no required name formats or fixed user IDs, and no fixed certification should be prescribed. For instance, all this is possible within the SAML format. The signed part of the response contains a handle (also called nonce or artifact), which is a freshly generated random number of sufficient length. It ensures the link between the correct browser and these attributes. It also contains the identity *SSL_id* of the destination site to counter man-in-the-middle attacks by dishonest destination sites.

If the destination site is not satisfied with the answer, it may repeat from Step 7. Otherwise it tells the wallet the address *return_address* to which the wallet should return (Step 10). By Decision 5, this address should be fixed, while the typical redirect to *target* should later be done internally at the destination site. The wallet carries out this redirect, including the handle in the query string. It is essential that this redirect is also via SSL.

One easily sees that even without the privacy requirements, no drastic improvements in efficiency would be possible, in particular with respect to the number of messages and thus the main part of the latency.

To improve the credibility of this protocol, we have worked a refinement of this protocol out as a SAML profile and integrated it into a research prototype.

5.3 Security Considerations

It is hopefully fairly clear by the design decisions that our attribute-exchange protocol as such does not violate the privacy requirements. In particular, no impostor can see attributes of a user by getting the real-time release request because the user is already authenticated in Step 5 (and not within Step 8, which would otherwise have been nice to decrease the number of user interactions) and because the same secure connection is used for the real-time release request. Similarly, an impostor cannot give an answer in the real-time release. Note, however, that some requirements are only fully fulfilled by higher layers, in particular by the details of the privacy policies used, the format for the attribute payload, and whether wallets and destination sites actually keep their privacy promises. Other requirements may in many situations be violated by lower layers, in particular the location secrecy.

Authenticity may actually be less clear, and thus we treat it in a bit more detail. The basic idea is that if a destination site requires authenticity, it only accepts a Step-9 response signed by the wallet, and with its own name *SSL_id* and a name or identity id_P of the person to be authenticated included. We also assume that the namespaces are such that no wrong wallet can confirm identity id_P to an honest destination site. When the wallet signed this response, it only sent it over a secure connection to exactly this destination site *SSL_id*. Here we assume SSL security and safe handling of the destination site's certificates. Thus an impersonation attack could only happen if a wrong user gets the nonce *handle* for this response. First, an honest wallet must have authenticated the user for id_P in Step 5. Here we assume sufficiently good registration if id_P is a pre-existing identity, and that the authentication is correct. In particular, the single signon password must not have been guessed, or the user tricked into revealing it to anyone else, or the browser or operating system been hacked. The wallet sends the handle to the browser (in the redirect) still within a secure session with this user. If we assume that the browser does nothing with the handle except forwarding it in the redirect (or possibly showing the redirect URL to the user, who should then also do nothing with it), the only remaining way of impersonation would be that the redirect goes to an attacker. However, the address *return_address* for this came from the same destination site with name *SSL_id*, and we required that SSL is used, hence indeed the handle only goes to the intended destination site.

A real proof of both privacy and security would actually be quite complicated, in particular because one has to formalize the assumptions about the browser and the user. This is in contrast to normal proofs of security protocols, where (for honest parties) one simply assumes protocol machines doing nothing but what is prescribed in the protocol. The main browser assumption, beyond HTTP and SSL conformance, is secrecy of the information from specific secure connections, here in particular the handle. This is a particular problem for the mobility case, where it depends on well-designed (non-)caching and that SSL connections do not survive when one user leaves the browser and another user comes, not even after crashes.

6 Summary

Browser-based attribute-exchange protocols can offer convenience to web users and seem widely desired by companies. We have investigated the privacy requirements on such protocols, when used in a general consumer scenario, that follow from general, accepted privacy principles. In particular, local and remote wallets must be able to coexist in one protocol. The requirements imply certain design decisions, in particular that public-key cryptography must be included and relative addressing of local wallets must be sufficient. We presented a protocol that fulfils all the requirements, while remaining very efficient, user-friendly, and deployable with standard components in short term.

Acknowledgments

We have benefited from discussions with many colleagues, in particular Peter Buhler, Peter Capek, Chris Giblin, John Hind, Stephen Levy, Matthias Schunter, and Jay Unger.

References

APPE_01 A P3P Preference Exchange Language 1.0 (APPEL1.0); W3C Working Draft 26 February 2001, http://www.w3.org/TR/P3P-preferences.html

BLKS_01 Kathy Bohrer, Xuan Liu, Dogan Kesdogan, Edith Schonberg, Moninder Singh, Susan L. Spraragen: Personal Information Management and Distribution, 4th International Conference on Electronic Commerce Research (ICECR-4); Dallas, TX, USA, Nov. 8-11, 2001

CaLy_00 Jan Camenisch, Anna Lysyanskaya: An efficient system for non-transferable anonymous credentials with optional anonymity revocation; Eurocrypt 2001, LNCS 2045, Springer-Verlag, Berlin, 93-117

Chau_81 David Chaum: Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms; Communications of the ACM 24/2 (1981) 84-88

Chau_85 David Chaum: Security without Identification: Transaction Systems to make Big Brother Obsolete; Communications of the ACM 28/10 (1985) 1030-1044

DiHe_76 Whitfield Diffie, Martin E. Hellman: New Directions in Cryptography; IEEE Transactions on Information Theory 22/6 (1976) 644-654

FaHo_02 S. Farrell, R. Housley (ed.): An Internet Attribute Certificate Profile for Authorization; Internet Request for Comments 3281, 2002, http://www.ietf.org/rfc/rfc3281.txt

FSSF_01 Kevin Fu, Emil Sit, Kendra Smith, Nick Feamster: Dos and Don'ts of Client Authentication on the Web; Proceedings of the 10th USENIX Security Symposium, 2001

Gato_99 Gator: The Smart Online Companion; first release 1999, http://www.gator.com/

Gola_01 Y. Y. Goland: Zero Install Single Sign On Solution for a HTTP Browser; Internet Draft, 2001, http://www.ietf.cnri.reston.va.us/internet-drafts/draft-goland-sso-human-00.txt

Harr_02 First Major Post-9/11 Privacy Survey Finds Consumers Demanding Companies Do More To Protect Privacy; Harris Interactive, Rochester, Feb. 20, 2002, http://www.harrisinteractive.com/news/allnewsbydate.asp?NewsID=429

HTTP_99 Hypertext Transfer Protocol -- HTTP/1.1; Internet RFC 2616, 1999

IBM_97 IBM Consumer Wallet; first release 1997, White Paper 1999 at http://www-3.ibm.com/software/webservers/commerce/payment/wallet.pdf

IBM_99 IBM Multi-National Consumer Privacy Survey (Conducted by Louis Harris & Associates, Inc.); IBM Global Services, October 1999

KoRu_00 David P. Kormann, Aviel D. Rubin: Risks of the Passport Single Signon Protocol, Computer Networks, Elsevier Science Press 33 (2001) 51-58

Libe_01 Liberty Alliance Project, http://www.projectliberty.org/, founded 2001

Micr_99 Microsoft Corporation: Various .NET Passport documentation, in particular Technical Overview, September 2001, and SDK 2.1 Documentation, http://www.passport.com and http://msdn.microsoft.com/downloads

P3P_02 The Platform for Privacy Preferences 1.0 (P3P1.0) Specification; W3C Recommendation, April 16, 2002, http://www.w3.org/TR/2002/REC-P3P-20020416/

Robo_99 Roboform: Free Web Form Filler and Password Manager; first release 1999, http://www.siber.com/roboform/

Pass_99 v-Go Single Signon; first release 1999, White Paper 2000, http://www.passlogix.com/media/pdfs/usable_security.pdf

SAML_02 OASIS Security Assertion Markup Language (SAML); Committee specification, April 19, 2002, http://www.oasis-open.org/committees/security/docs

Shib_01 Shibboleth-Architecture DRAFT v04; Nov 21, 2001, http://middleware.internet2.edu/shibboleth/docs/draft-internet2-shibboleth-arch-v04.pdf

Slem_01 Marc Slemko: Microsoft Passport to Trouble; Rev. 1.18, Nov. 5, 2001 http://alive.znep.com/~marcs/ passport/

West_67 Alan F. Westin: Privacy and Freedom; Atheneum, New York NY, 1967

Wilc_02 Joe Wilcox: Customers wary of online IDs and Survey: Passport required--not appealing, CNET News.com, April 2002, http://news.com.com/2100-1001-892808.html and http://news.com.com/2100-1001-884730.html

Zero_99	Zeroknowledge: Freedom Personal Firewall; first release 1999, http://www.freedom.net/products/firewall/index.html