RZ 3455 (# 93800) 09/09/02
Computer Science 12 pages

Research Report

BBAE – A General Protocol for Browser-based Attribute Exchange

Birgit Pfitzmann and Michael Waidner

IBM Research
Zurich Research Laboratory
8803 Rüschlikon
Switzerland
{bpf,wmi}@zurich.ibm.com

BBAE – A General Protocol for Browser-based Attribute Exchange

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

Abstract

Browser-based attribute-exchange protocols enable users of normal web browsers to conveniently send attributes, such as authentication or demographic data, to web sites. This is also called federated identity. Such protocols might become very common and almost mandatory in general consumer scenarios over the next few years. Several product and standards proposals have been made, most notably Microsoft Passport, OASIS SAML, and the Liberty Alliance V1 specifications. However, none of the current proposals – by statements of the proposers themselves – addresses the full functionality for a general consumer scenario. We propose a protocol BBAE that addresses the missing issues. It has been fully specified with existing standards elements and prototyped, and we present an initial security analysis. We also discuss how it can be used as a step forward in existing standardization processes.

1 Introduction

Electronic commerce with end consumers has not fulfilled the high expectations of a few years ago. By current consumer surveys, the main reason for reticence is lack of trust, in particular with respect to privacy, including protection from spam mail and unwanted phone calls. A second reason is that many services are not easy to use. An approach to minimize this second problem is to provide single signon or one-click features across the entire world-wide web. This means that users should be able to make transactions without remembering anything and typing anything in, except typically for one master password and some OKs. This is also called federated identity. Analysts forecast that such protocols are likely to become very common over the next years. However, unless designed very carefully, these features make the situation for trust and privacy even worse, and are thus counterproductive.

This is likely to be the reason why none of the players has yet addressed the full functionality of attribute exchange for a general consumer scenario, while all have announced to consider it in the future. By attribute exchange, we mean a generalization of single signon, which classically refers only to authentication. In an e-commerce scenario, obtaining payment and shipping data for a transaction is more important than authentication, and the sites are typically interested in additional demographic or usage data. In particular, Microsoft Passport does not yet address a truly federated scenario without control points, and, while making quite good privacy promises for the limited number of attributes it manages, does not yet have flexible privacy policies as would be needed for a larger variety of attributes [Micr_01, Micr_02]. SAML, the Security Assertion Markup Language that is currently in a voting phase as an OASIS standard, defines a very flexible message format, but its profiles, i.e., protocols using these messages, so far only address core parts of single signon [SAML_02]. The first Liberty Alliance specifications, building on SAML, are explicitly restricted to single signon and to fairly small, closed federations [Libe_02]. (We see technical reasons for these statements below; anyway, they have all been made by the respective organizations.) Similar restrictions hold for IBM's e-Community Single Signon product (eC-SSO, see [IBM_02], Ch. 10.2) and the Internet Draft [Gola_01]. (As the latter is only single signon and expired without seeming to have found support, we do not mention it again.) The Internet 2

project Shibboleth, if one extracts its protocol aspects, is closest to the full functionality, although somewhat restricted due to its concentration on applications in universities [Shib_02].

The proposals we just mentioned are mainly distinguished from related older ones like form fillers, classical wallet products, and PKIs, by two functional requirements, zero-footprint and mobility. Our protocol BBAE is in this same class. The zero-footprint requirement means that the protocols should work even if a user has nothing but an arbitrary unmodified browser. We call protocols with this feature browser-based. The reason to consider this is that market studies suggest that a large user segment is currently not willing to install additional software or even hardware for electronic commerce, neither for ease of use nor for security or privacy. Zero-footprint should include that not even active content such as JavaScript, Java or Active X is needed on the browser, e.g., for security reasons. The protocols should even work without cookies. Implementations may make use of additional features when they are available. The mobility requirement means that a user should be able to use the protocols from varying browsers, such as several personal devices or even Internet kiosks. The latter is quite dangerous for security, but as nobody is forced to use kiosks (at least in developed countries), we simply accept this as a market requirement. This excludes simple browser personalization from this protocol class.

When one moves from a protocol for closed federations and single signon, possibly with a few fixed attributes, to one for a general consumer scenario and arbitrary attributes, the main additional issues to consider are how to find a user's attributes, and flexible privacy. Further, users should be able to protect themselves against impersonation. Besides, some protocols have naming, set-up, or message-length restrictions that do not suit the general case. If one augments any existing proposal by those of these features that it does not address yet, one essentially always ends up with the same kind of protocol, which our BBAE protocol exemplifies. We show that all these additions can be made as options, so that in simpler situations the protocol automatically provides the same efficiency and ease of use as specialized protocols for these situations.

Overview of this paper: In Section 2, we present the general structure of attribute-exchange protocols. This serves as an introduction of notation, as an overview of our following protocol, and as a framework for presenting more details about related work. We also explain some design decisions already on this general level. In Section 3, we present our protocol BBAE in detail. Section 4 contains brief security considerations.

2 General Protocol Elements

The goal of a browser-based attribute-exchange protocol is to enable a *user* at a browser to exchange attributes with arbitrary web sites, called *destination sites*. The user and a destination site need no prior relationship, and the user typically need not type in the attributes. By *attributes* we mean arbitrary personal information, including authentication information.

For users of the zero-footprint version, the attributes must clearly be held by another party; we call it *wallet holder*. By *wallet* we mean the collection of attributes of one user at one wallet holder, including the methods to use them. We allow users who prefer better security and privacy over the zero-footprint convenience to be their own wallet holders; we call this *local wallets* and the others *remote wallets*.

Browser-based single signon protocols are therefore three-party authentication protocols with the restriction that the client application is only a browser, and general browser-based attribute-exchange protocols are extensions of this. In all such protocols, we can identify some common phases, due to the restrictions of the browser. This is shown in Figure 1. The two redirects and the user authentication (Phases C, D, G) are common to all, while different protocols have various subsets of the other elements, and not always in exactly this order.

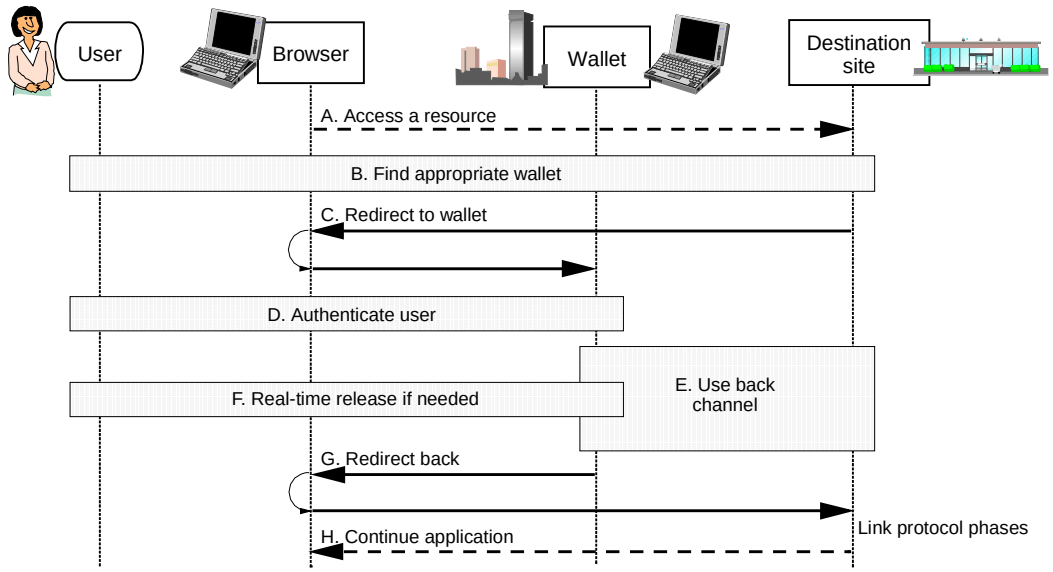

Figure 1 *Overview of browser-based attribute-exchange protocols. The normal arrows are mandatory, the dashed arrows are before and after the protocol, and the boxes are only needed in certain cases.*

Prior browsing. Before the protocol start, the user is browsing at the destination site (Phase A). This is usual in the general consumer scenario, in contrast to a portal scenario where the user already starts at the wallet. It is also assumed in the related literature, except that the SAML SSO profiles leave this start open, and thus do not specify Phases B and C.

Redirecting to wallet. If the user arrives at a resource where the destination site would like user attributes, e.g., for finalizing an order or for optional personalization, the destination site redirects the browser to the wallet (Phase C). This is usually, but not always, an HTTP redirect. If the wallet is user-chosen, some substeps may be needed to find the address of the appropriate wallet (Phase B).

Microsoft Passport does not have Phase B because all wallets are at a fixed Passport address, nor does IBM eC-SSO as a proposal for closed federations with a fixed wallet holder per federation. Liberty has this phase, but the given solution does not scale to a general consumer scenario: It assumes a common domain for each federation, and each wallet initially sets a cookie on the browser for this domain. Large common domains would also be a problem for cookie security and privacy. Shibboleth has a full version.

User authentication. Before the wallet starts transferring attributes or authentication information, it must reliably find out who the current user is. Thus user authentication follows (Phase D), except if the second half of the redirect in Phase C reused an existing secure session between the user and the wallet.

Request and response. Now the wallet needs to know what attributes the destination site asks for. This request may either have been transmitted in the redirect, or a *back channel* is now set up, i.e., a direct channel between wallet and destination site. In a mere single signon protocol the request may be omitted. Microsoft Passport, with its relatively few attributes, also omits it and assumes everyone asks for everything available. In contrast, the Liberty single signon does contain a request, e.g., to ask for specific authentication strengths. Similarly, the response may be sent on the back channel or directly in the final redirect of the browser to the destination site.[1]

[1] Some protocols exchange Phases E and G, i.e., redirect the browser back to the destination site before setting up a back channel. However, a real-time release then requires two additional redirects. In both variants, someone has to

The main reason to use a back channel is to transport long information. If one performs a redirect phase by a browser redirect in the narrow sense (HTTP response 302 or 303), the only place for transporting information between wallet and destination site is the query string of the URL. In [HTTP_99], URLs are only considered safe up to 255 bytes, which is quickly exceeded, e.g., for signed responses. Cookies do not work for arbitrary domains. An alternative is to send the user a (hidden) form to POST. However, then either the user must press a submit button, which gives an unnecessary user interaction, or active content is needed. Furthermore, a back channel may have much larger bandwidth. Only Microsoft Passport and IBM eC-SSO define no back channel at all.

Real-time release. The wallet cannot always derive a response to the request on its own. For instance, it may not hold all requested attributes, or the privacy policy may not contain an a priori decision whether this release is allowed. In these cases the wallet may involve the user; we call this real-time release.[2] (If the attributes and a suitable policy are available, the user should not bothered.)

Redirect back. When the browser is finally redirected back (Phase G), but the response was sent on the back channel (Phase E), care must be taken how the destination site links the returning browser to the corresponding response, in particular for avoiding man-in-the-middle attacks.

3 The BBAE Protocol

We now present our BBAE protocol in detail. It is an all-purpose protocol with all the features needed in a general consumer scenario. It contains all elements of Figure 1 in the order given there. We also discuss how it can adapt to various simpler cases and justify some design decisions. Moreover, we mention how some details are handled in related protocols.

Our emphasis is on the protocol, i.e., the part of an overall solution that would be standardized.

3.1 Protocol with Parameters

Figure 2 shows the protocol steps with their parameters. The correspondence of steps to phases of Figure 1 is indicated by the boxes.

This figure is an abstract description. We have a fully worked-out version of the protocol as a SAML profile; this was the easiest way to obtain a formulation suitable for a standards proposal because SAML already provides a format for general attribute responses and responses. It has been prototyped by Stephen Levy in the IBM Privacy Services prototype; an earlier version of that system was presented in [BLKS_01].

The following presentation of details mostly stays on the abstract level for reasons of length and easier (at least initial) security considerations. Moreover, the abstract version shows that arbitrary existing systems and standards in this class can be extended by such a general protocol or profile.

---

prevent browser timeouts while the backchannel is used; in Figure 1 this is the wallet, otherwise the destination site before the first application-oriented response (Phase H) follows.

[2] User authentication can be postponed after the attribute query, but cannot be joined with a real-time release if that becomes necessary: Before a real-time release, a remote wallet must learn whose attributes it is looking up, and every wallet must authenticate its user before showing privacy-critical attributes on his or her screen for releasing.

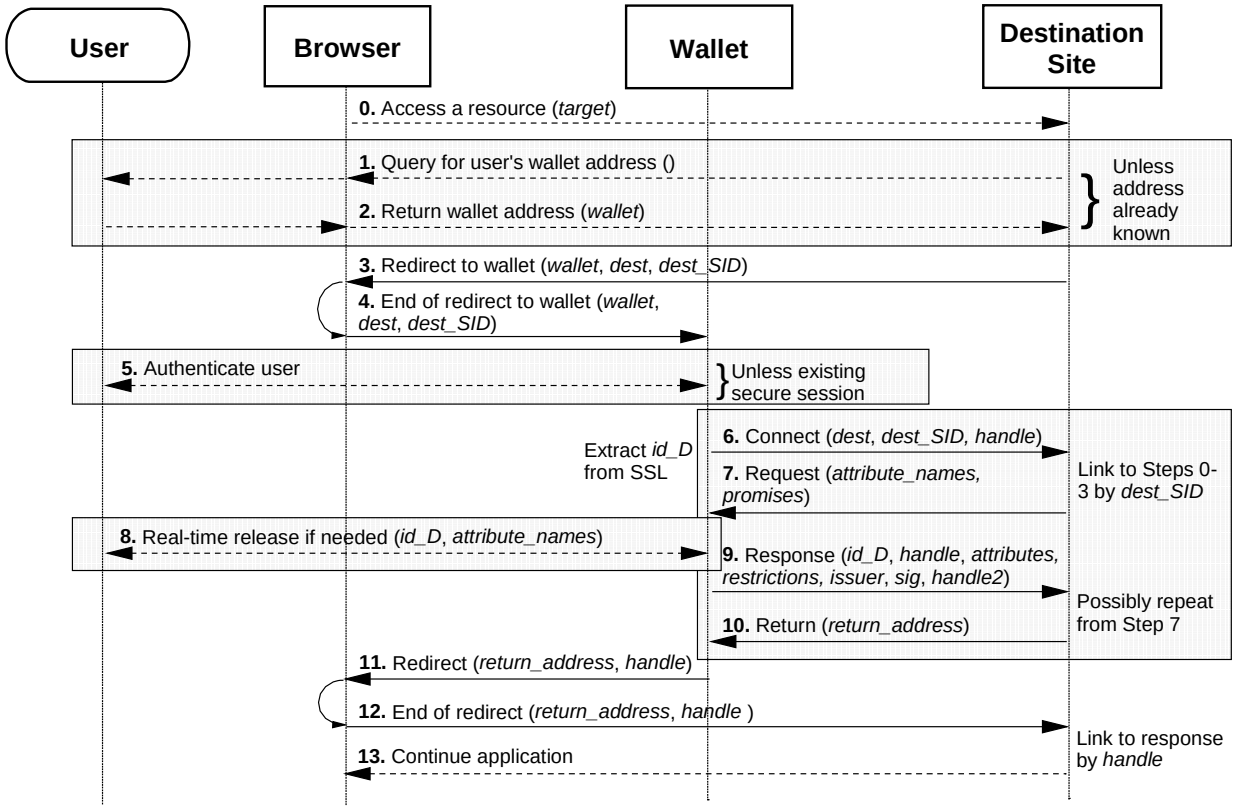

Figure 2 *BBAE protocol with parameters. Steps with dashed lines are only needed in certain cases. Secure sessions are required at least for Steps 5, 8, 11, Steps 6, 7, 9, 10, and for Step 12.*

3.2 Required Set-up

Initially, a user must register with a wallet, i.e., exchange authentication information with it. Typically this is a user name and password. If the wallet is remote, the user obtains a contact URL *wallet* of it. More precisely, this is the host part of a URL, and it is augmented by https:// and a fixed path /BBAE-wallet in the protocol.[3]

All parties must be able to use SSL/TLS channels in the form of https. Every source and destination site must be able to keep *secure sessions*. We define this to mean that message confidentiality and integrity hold with respect to the same principals for all messages of the session. The easiest implementation is to use SSL/TLS channels as secure sessions. Sites that release SSL channels after each HTTP response can use any other usual implementation, but should have at least one zero-footprint version.

A destination site should have an SSL/TLS server certificate. Anonymous destination sites are allowed, but are unlikely to get many attributes under the users' privacy policies. A wallet must be able to make XML signatures [XMLS_02], and a destination site must be able to verify them. XML signatures comprise not only digital signatures in the cryptographic sense, but also symmetric message authentication codes, and they allow many ways of certification or key distributions. We discuss typical usages and their security implications in Section 4.3.

[3] Shibboleth proposes user-memorable names for its universities and a translation scheme. For the general consumer scenario, we do not assume such additional infrastructure initially, although it may develop over time.

3.3 Find Appropriate Wallet

The goal of the wallet-finding phase, Steps 1-2, is to provide the destination site with a URL *wallet* for use in the subsequent redirection. (More precisely, *wallet* is the host part, which is then augmented with a fixed path.) It is important to allow *wallet* to be "localhost", in order to include local wallets: First, local wallets may not have fixed addresses; secondly, for privacy we do not want a fixed user address at this point.

The only truly general protocol version is to ask the user for the wallet location. A possible form for such a user query is shown in Figure 3. The radio button "it is local" leads to *wallet* = localhost. This, and the following line with the text input for the hostname *wallet* remembered from the set-up, and at least one "no" or "cancel" are mandatory in BBAE.

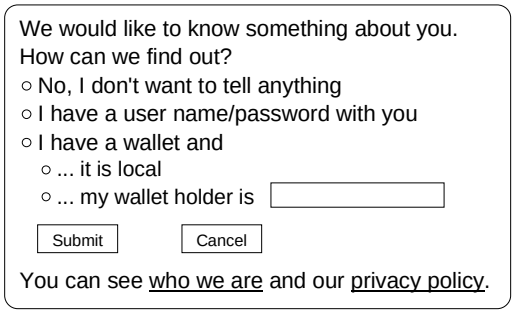

Figure 3 *Example of the form for a wallet location query (Step 1)*

Asking the user is necessary in the mobility case, where neither the destination site nor the browser know anything about the user at the beginning. It is also needed at least once per user and destination site (or group of related destination sites) if users have a free choice of wallet holders, e.g., among all the banks in the world. Further, it is needed in each interaction if a user has several wallets without clear separation of responsibilities that could be evaluated automatically, in particular if the user simply has different wallets for different roles. Multiple unlinked wallets are, e.g., Microsoft's explicit proposal for handling multiple roles; only in the current Passport they are still all at the same address. For the other cases, implementers may choose any state-of-the-art techniques of state keeping, either governed by the destination site or directly by browser extensions.

3.4 Redirect to Wallet

In Steps 3-4, the destination site redirects the browser to the obtained address *wallet*. In the query string, it includes a contact address *dest* and a session identifier *dest_SID*. For privacy, these parameters should not include any (unencrypted) information about the user's browsing behavior. We therefore recommend that *dest* is fixed and *dest_SID* a nonce, i.e., a freshly generated random number. (In contrast, all other proposals recommend to use the user's original URL *target* in the place of *dest_SID*.)

By the redirect, the wallet therefore obtains the parameters *dest* and *dest_SID*.

3.5 Authenticate User

Upon receiving a Step-4 message at the appropriate URL, the wallet first authenticates the user of this browser (Step 5) with the method agreed upon during registration. While we also recommend HTTPS for the previous steps, it now becomes essential that a secure session in the sense of Section 3.2 is used and maintained for the (potential) Step 8 and Step 11. If such a session already exists, e.g., from a prior

execution of the BBAE protocol, it can be reused; however, in particular if cookies are used and in the mobility scenario, this is quite dangerous, compare attacks on Microsoft Passport [Slem_01].

3.6 Request

Now the wallet sets up the back channel by contacting the destination site at the obtained address *dest* (Step 6).[4] This step builds up a secure session for use until Step 10, to guarantee privacy of the attribute request and response against outsiders, and to ensure authentication of the destination site. In BBAE, the wallet uses HTTPS, and extracts the name *id_D* of the destination site from the server certificate.

In the query string of the HTTPS request, the wallet includes the destination site's session identifier *dest_SID*. Due to a SAML idiosyncrasy, we also include a nonce *handle* that is mainly needed in Step 9.

Using *dest_SID*, the destination site can look up *target* and other parameters on which its attribute request depends. It then sends this request in the secure session (Step 7). In Figure 1, *attribute_names* abstractly indicates the attribute names requested. Our concrete protocol version uses a SAML request that contains exactly one SAML attribute query. This message format can ask for attributes from arbitrary name spaces. Nevertheless, for the general consumer scenario one or more specific name spaces should be fixed for interoperability. Our initial recommendation is to use P3P and its base data schema [P3P_02] directly as a request language. This contains all the typical attributes that e-commerce sites ask for, and allows privacy promises to be included in the request. This is important for combining user convenience and privacy, because the wallet can then compare the user policy with the promises. No other current proposal includes privacy promises in requests yet.

3.7 Deriving a Response

The wallet now derives a response to the request. This should be based on a privacy policy that specifies who is allowed to obtain which attributes. Typically, the wallet needs the identity of the requestor as input to the policy evaluation; in BBAE it uses the name *id_D* of the destination site.

The protocol is independent of the privacy-policy language used, but for interoperability with the possible promises in the request, common languages should be fixed. For wallets acting primarily as user agents, a standards proposal is APPEL [APPE_01]. It allows to specify preferences with respect to data recipients and their P3P promises. Wallet holders who primarily store data for their own purposes and offer the wallet functionality as an addition might have an enterprise privacy management system such as in [KaSW_02].[5] In addition to decisions, the policy evaluation may output usage restrictions. They are then included in the response. This is shown as *restrictions* in addition to *attributes* in Figure 2.

If the wallet is missing attributes, or the user made no prior privacy decision covering the given situation, Step 8, the real-time release, is necessary. This presupposes three-valued privacy policies, i.e., with "allow" and "deny" for some situations (where the user is not bothered with a real-time release), and "ask me" for situations that the user does not want to think about before. In APPEL, this is the prompt attribute. Such an approach corresponds well to user studies that people do not want to start with abstract concepts, like policies, but with examples. (How real-time releases can be extended to build up policies

[4] BBAE is the first protocol where the wallet, and not the destination site, sets up the back channel. This is important for including local wallets, just like allowing redirects to localhost, because local wallets may not have a fixed address and even if they do, we do not want to use a fixed user address here for privacy.

[5] Among the prior proposals, Passport has a simple fixed privacy policy with two data categories, demographics and purchase. The former data, if present, are released to all destination sites who ask, while the latter are always handled with real-time releases. Shibboleth allows quite a detailed policy in its own language, supporting arbitrary data categories and data users by name or subtree, but no purposes or obligations.

over time is shown by Susan Spraragen and Stephen Levy in a successor of [BLKS_01], the prototype that was also extended by BBAE.)

A simple form for a real-time release is shown in Figure 4. Where the wallet knows the attribute, it distinguishes whether the user has pre-authorized the release in a policy (green, vertical stripes) or not (red, diagonal stripes). In our example the name and shipping address are pre-authorized, while the ID number is not. The user might delete the ID number and not add an income, because the fields are not mandatory. If they were, he might cancel.

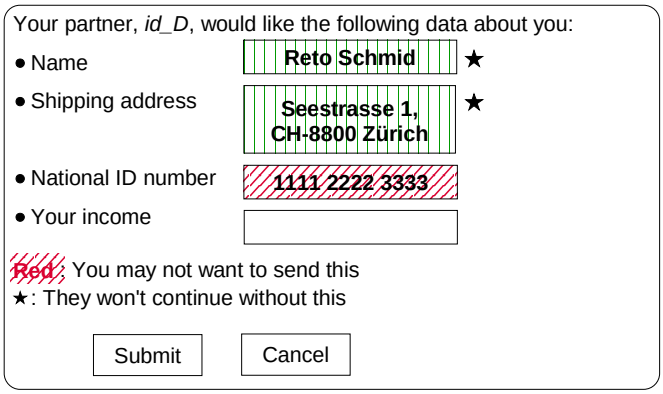

Figure 4 *Example form for a real-time release (Step 8).*

This step may also involve a second wallet that holds or confirms certain attributes. We sketch the main options, although we have not specified this in a SAML profile yet. (Nor does any other proposal contain confirmed attributes yet.) If the second wallet has to confirm a permanent or slow-changing attribute for a certain name, e.g., the year of birth, it may give the first wallet a confirmation token, e.g., a SAML attribute assertion, with an appropriate validity period. This simply becomes an attribute in the first wallet. If the confirmation must be fresh and the second wallet's policy allows release to the first wallet, the first wallet can fetch the confirmation on a back channel. For a real-time release at the second wallet, the first wallet must redirect the browser to it. If the wallets are not linked, the first wallet only answers what it can, and the destination site has to restart the protocol, asking the user for the appropriate wallet for the remaining attributes.

3.8 Response

In Step 9, the wallet sends the derived response in the secure session. The response contains the name *id_D* of the destination site (this is important against man-in-the-middle attacks), a nonce *handle*, and if authentication is desired, an XML signature *sig* by the wallet over the whole response under a name *issuer*. The destination site verifies these elements. (Recall that [XMLS_02] allows a variety of instantiations and will be discussed in Section 4.3.)

In our SAML instantiation, Step 9 is a POST of a SAML response. The nonce *handle* is a SAML "artifact", and included in the SAML subject element for purposes like ours under "subject confirmation". We put *id_D* in the SAML recipient element. For the same reason as we included *handle* in Step 6, we include a new handle *handle2* in case Steps 7 and 9 are repeated.

3.9 Redirect Back

If the destination site is not satisfied with the response, it may repeat from Step 7. Otherwise it tells the wallet the address *return_address* for the browser (Step 10). For privacy of the user's browsing behavior towards the wallet, this address should be fixed. The wallet now initiates the redirect back (Step 11), still in the secure session with the user authenticated in Step 5. It includes the nonce *handle* in the query string. The second part of the redirect (Step 12) must also be via SSL, i.e., *return_address* must be HTTPS. The destination site, upon receiving a Step-12 message, uses *handle* to retrieve the corresponding attribute response from Step 9 and the session identifier *dest_SID* from Step 6.

Typically it then internally redirects the user to the original *target* with parameters from the response.

4 Security Considerations

We now discuss the security and privacy properties of BBAE. Under security, we only discuss authenticity, because non-repudiation is not a goal of browser-based single signon or attribute exchange. (Technically, this is because no user signatures on messages are made, and those cannot be added at least in the zero-footprint version.)

4.1 Authenticity

Often one of the attributes that the destination site requests is a name or role name (pseudonym) id_U of the user *U*. No other user, nor a dishonest destination site, should then be able to impersonate this (honest) user under id_U to an honest destination site *D*. In addition, we have to assume that all wallet holders from whom *D* accepts a confirmation of id_U are honest. This depends on name spaces and certification, see Section 4.3. More precisely, we show that the destination site has a secure session to this user *U* in Step 12 (which it typically retains for following actions), and all the remaining attributes from the response belong to id_U as far as the wallet knows.

The basic idea why BBAE guarantees this under certain operational assumptions (mentioned in the following) is that if a destination site requires authenticity, it only accepts a Step-9 response *R* signed by one of the wallets we just assumed to be honest, with this name id_U as an attribute, and containing its own name *id_D* and a value *handle*. When the wallet signed this response *R*, it only sent *R* in a secure session to exactly this destination site *D* whose name *id_D* it included. Here we assume SSL security and safe handling of the destination site's certificates (in contrast to recent attacks on some implementations). Thus an impersonation attack can only happen if an adversary gets the nonce *handle*, so that it can carry out Step 12 with this destination site. We now show that this cannot happen.

First, the wallet has authenticated the user for id_U in Step 5. Here we assume sufficiently good registration if id_U is a pre-existing identity, and that the authentication is correct. In particular, if a password is used with the wallet it must not have been guessed or the user tricked into revealing it to anyone else,[6] and the browser and operating system must not have been hacked. All these are operational limitations to the security achievable with browser-based protocols; for more details see [KoRu_00, Slem_01] (and [FSSF_01], but that only treats direct authentication to a destination site).

The wallet sends *handle* to the browser (in the redirect) still in a secure session with the same user as in Step 5. If we assume that the browser does nothing with *handle* except forwarding it in the redirect (or possibly showing the redirect URL to the user, who then also does nothing with it), the only remaining

[6] A particularly dangerous addition in Passport and Liberty is "inline single signon", where the wallet does not set up an extra browser window for the input of the single signon password, but takes a part of the destination site's window. This annihilates all user education on checking certificates before inputting critical data.

way of impersonation is that the redirect (Step 12) goes to the adversary. However, the address *return_address* for this came from the same destination site *D* in a secure session, and we required it to be HTTPS and assumed secure SSL. Hence indeed *handle* only goes to the intended destination site *D*.

4.2 Privacy

We now summarize the privacy properties that BBAE enables. The resulting actual privacy also depends on higher layers, in particular the privacy policies used and whether wallets and destination sites keep their privacy promises. Other requirements may be violated by lower layers, in particular location secrecy.

- Attribute privacy. All attributes in a wallet can be placed under arbitrary privacy policies, and the request derivation evaluates those. Further, policies can be negotiated in the request and response steps. Flexibility for users who do not know their entire preferences in advance is given by allowing real-time release. Correct policy enforcement is ensured by authentication of all parties to whom (restricted) data are released: First, the destination site is identified using *id_D*. Secondly, no impostor can see attributes of a user by getting the real-time release request because that is sent in a secure session to the user authenticated in Step 5. Similarly, an impostor cannot give an answer (e.g., releasing too many data) in the real-time release.
- Multiple roles. User identifiers are handled like other attributes and can thus be placed under privacy policies. This enables multiple roles, conveniently handled in one wallet. The most important initial case is to send demographics or preferences without any identifier. This needs no additional administration. In contrast, reusing certain roles in certain situations and assigning different attributes to them needs administration and user interfaces.
- Privacy of traffic data. The only information that a (remote) wallet automatically learns about user behavior in BBAE is fixed addresses of the destination sites that ask for attributes. This seems unavoidable both for the redirect back and for man-in-the-middle security. BBAE avoids unnecessary additional information such as exact target URLs.
- Wallet choice. BBAE can handle user-chosen remote wallets, multiple wallets per person, and local wallets. In particular, the local-wallet case is compatible with multiple roles (no other current proposal fulfils this) because no wallet identifier is given automatically to each destination site. We saw already that no wallet address is needed on this protocol layer. Section 4.3 also shows that issuing and signing attribute responses (Step 9) can be compatible with the desired degree of identification.

(A much more detailed discussion of privacy, from user surveys over more detailed requirements to limitations of browser-based protocols compared with unlinkable credentials as in [CaVa_02], is contained in an unpublished companion manuscript.)

4.3 On Identifiers, Keys, and Certificates

We now recommend how to use identifiers, keys, and certificates of wallets in the most important scenarios. Our use of SAML responses with an issuer name *issuer* (mandatory, but arbitrary string) and XML signature *sig* (optional) left that open. We concentrate on the general consumer scenario with many wallets and destination sites. In addition, a wallet can use *id_D* to first look up whether it has a closer relationship with the destination site, e.g., a joint secret key, for a more efficient element *sig*.

Form-filling case. The most usual case for initial e-commerce does not require authentication, because it replaces current forms that the user fills in without confirmation, e.g., with demographics and preferences. Even sending addresses and payment information belongs to this case, because the payment information is

verified with existing payment systems. Then no issuer name and signature are necessary. With SAML and for a really anonymous response, a local wallet should choose a nonce as *issuer* and omit *sig*.

Real identity. For authenticity under a name id_U with prior meaning, e.g., a legally identifying name or an email address, this name must be verified in registration. Authenticity cannot get better than this verification, and all parties trusted by a destination site to confirm this name must be honest. (This is the same tradeoff between security and convenience as with PKIs; it has nothing to do with whether only a password is exchanged in registration or a public key.) Users with a local wallet have to register a public key *pk* of the wallet and get it certified for id_U. The local wallet thus becomes a leaf in a tree of certification authorities, where it may only confirm one or a few names from this externally managed name space.

Long-term role. If a wallet generates a role name *role* for repeated use (e.g., with a certain federation of enterprises), but without verified prior meaning or even with the explicit desire to keep this role unlinkable to other roles, it generates a new name. (SAML explicitly allows this.) This is like SPKI/SDSI compared with classical PKIs. A remote wallet holder can issue responses for *role* under its own fixed name *issuer* and with a fixed key *pk* without disturbing the potential unlinkability if its user community is large enough to offer anonymity. If the wallet is local and the role should really be unlinkable, it should generate a fresh name $issuer_{role}$ and key pk_{role} for this role. For XML signatures, the wallet can simply include the key in the "KeyValue" element when first using it with a destination site. The name *role* should be considered to be in a name space governed by *pk* or pk_{role}, respectively. Then only the wallet that issued *role* must be trusted for authenticity in this role.

To distinguish these cases precisely, one could apply authentication context definitions as in [Libe_02] to individual attributes in a response. For instance, one could represent the Passport case by stating that names and payment information have not been verified, while the email address has been a little.

4.4 Security Outlook

A real proof of authenticity and privacy would 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 that do nothing but what is prescribed in the protocol. The main browser assumption, beyond HTTP and SSL conformance, is secrecy of the information from specific SSL channels, here in particular *handle*. This is a particular problem for the mobility case, where it depends on well-designed (non-)caching and that SSL channels do not survive when one user leaves the browser and another user comes, not even after crashes.

5 Summary

We have presented the first browser-based attribute-exchange protocol, BBAE, which covers the full functionality required in a general consumer scenario. In particular, it allows an arbitrary number of wallet holders that can interact with all interested destination sites without special set-up and central control points, while also allowing smaller federations with closer trust relationships. It allows multiple wallets per user (e.g., for financial versus employment information) and multiple roles in a wallet (e.g., in a "primary" wallet of a person). No privacy sensitive information is released automatically except the names of visited destination sites to the wallet, in particular no global identifiers, detailed browsing histories, or location information. Thus all this information can be governed by privacy policies. These can be set arbitrarily by users in situations where the users act for themselves, while other applications, e.g., in an employment relationship, can have predetermined policies. A prototype with the protocol exists. Even

without the privacy requirements, no serious efficiency improvements over BBAE are possible, in particular with respect to the number of messages and thus the main part of the latency.

BBAE is also the first browser-based attribute-exchange or single-signon protocol with at least a sketch of an authenticity proof. It is certainly not easier to construct flawless browser-based protocols than classical authentication protocols like Needham-Schroeder, which have quite a bad track record.

We believe that all the main current proposals, like Microsoft Passport and the Liberty Alliance's, will need a BBAE-like protocol to fulfill the promises made for their next phases.

Acknowledgments

We have benefited from discussions with many colleagues, in particular Peter Buhler, Peter Capek, Chris Giblin, Thomas Groß, 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, Nov. 2001

CaVa_02 Jan Camenisch, Els Van Herreweghen: Design and Implementation of the Idemix Anonymous Credential System; accepted for ACM CCS 2002, Washington, Nov. 2002

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

Gola_01 Y. Y. Goland: Zero Install Single Sign On Solution for a HTTP Browser; Internet Draft, Nov. 2001, http://www.ietf.cnri.reston.va.us/internet-drafts/draft-goland-sso-human-00.txt [BPf6]

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

IBM_02 IBM: Enterprise Security Architecture using IBM Tivoli Security Solutions; April 2002, http://www.redbooks.ibm.com/abstracts/sg246014.html

KaSW_02 Günter Karjoth, Matthias Schunter, Michael Waidner: Platform for Enterprise Privacy Practices; to appear in 2nd Workshop on Privacy Enhancing Technologies (PET), LNCS, Springer-Verlag, 2002.

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_02 Liberty Alliance Project (founded 2001): Specifications Version 1.0 (6 parts), July 2002, http://www.projectliberty.org/specs/liberty-specifications-v1.0.zip

Micr_01 Microsoft Corporation: Various .NET Passport documentation (started 1999), in particular Technical Overview, Sept. 2001, and SDK 2.1 Documentation; http://www.passport.com and http://msdn.microsoft.com/downloads

Micr_02 Microsoft Corporation: Microsoft Federated Security and Identity Roadmap, June 2002, http://msdn.microsoft.com/library/default.asp?url=/library/enus/dnwebsrv/html/wsfederate.asp?frame=true

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

SAML_02 OASIS Security Assertion Markup Language (SAML); Committee specification 01, May 2002 (started Jan. 2001), http://www.oasis-open.org/committees/security/docs[BPf7]

Shib_02 Shibboleth-Architecture DRAFT v05; May 2002 (v1 in 2001) http://middleware.internet2.edu/shibboleth/docs/draft-internet2-shibboleth-arch-v05.pdf

Slem_01 Marc Slemko: Microsoft Passport to Trouble; V1.18, Nov. 2001, http://alive.znep.com/~marcs/passport/

XMLS_02 XML-Signature Syntax and Processing; W3C Recommendation, Feb. 2002, http://www.w3.org/TR/xmldsig-core/