Client-based CardSpace-OpenID Interoperation

Haitham S. Al-Sinani[*] and Chris J. Mitchell

Information Security Group
Royal Holloway, University of London
`[Haitham.Al-Sinani.2009, C.Mitchell]@rhul.ac.uk`

Abstract. We propose a novel scheme to provide interoperability between two of the most widely discussed identity management systems, namely CardSpace and OpenID. In this scheme, CardSpace users are able to obtain an assertion token from an OpenID-enabled identity provider, the contents of which can be processed by a CardSpace-enabled relying party. The scheme, based on a browser extension, is transparent to OpenID providers and to the CardSpace identity selector, and only requires minor changes to the operation of the CardSpace relying party.

1 Introduction

To mitigate identity-oriented attacks, a number of identity management systems (e.g. CardSpace, OpenID, etc.) have been proposed. An identity provider (IdP) in such systems supplies a user agent (UA) with an authentication token that can be consumed by a relying party (RP). Whilst one RP might support CardSpace, another might only support OpenID. To make these systems available to the largest possible group of users, interoperability between such systems is needed.

We consider CardSpace-OpenID interoperation because of OpenID's wide adoption; complementing this, the wide use of Windows, recent versions of which incorporate CardSpace, means that enabling interoperation between the two systems is likely to be of significance for large numbers of identity management users and service providers. CardSpace-OpenID interoperation is particularly attractive since both schemes support user attribute exchange. In addition, supporting interoperation on the client could be practically useful, because IdPs/RPs might not accept the burden of supporting two identity systems unless there is a significant financial incentive. Client-based interoperation means that the performance of the server is not affected, since the overhead is handled by the client.

2 CardSpace and OpenID

2.1 CardSpace

CardSpace is Microsoft's implementation of a digital identity metasystem, in which digital identities are represented to users as Information Cards (or InfoCards). There are two types of InfoCards: personal (self-issued) cards and

[*] This author is sponsored by the Diwan of Royal Court, Sultanate of Oman.

managed cards, issued by remote IdPs. Personal cards are created by users themselves, and the claims listed in such cards are asserted by the self-issued identity provider (SIIP) that co-exists with the CardSpace identity selector (or just the selector) on the user machine [1]. The integration scheme uses CardSpace personal cards to make information provided by OPs available to CardSpace RPs.

The personal card protocol operates as follows, in the case where the RP does not employ a security token service (STS).

1. UA $\rightarrow$ RP. HTTP/S request: GET (login page).
2. RP $\rightarrow$ UA. HTTP/S response. A login page is returned containing the CardSpace-enabling tags in which the RP security policy is embedded.
3. User $\rightarrow$ UA. The RP page offers the option to use CardSpace; selecting this option activates the selector, which is passed the RP policy. If this is the first time that this RP has been contacted, the selector will display the identity of the RP and give the user the option to either proceed or abort the protocol.
4. Selector $\rightarrow$ User. The selector highlights InfoCards matching the policy.
5. User $\rightarrow$ Selector. The user chooses (or creates) a personal card. The user can preview the card to ensure that they are willing to release the claim values.
6. Selector $\rightleftharpoons$ SIIP. The selector sends a Request Security Token (RST) to the SIIP, which responds with a Request Security Token Response (RSTR).
7. UA $\rightarrow$ RP. The RSTR is passed to the UA, which forwards it to the RP.
8. RP $\rightarrow$ User. The RP validates the token, and, if satisfied, grants access.

The PPID is an identifier linking an InfoCard to an RP. When a user first uses a personal card at a particular RP, CardSpace generates a card-site-specific PPID by combining the card ID[1] with data taken from the RP certificate, and a card-site-specific signature key pair by combining the card master key with data taken from the RP certificate. The PPID could be used on its own as a shared secret to authenticate a user to an RP. However, it is recommended that the associated (public) signature verification key should be used to verify the signed security token to provide a more robust authentication method [1].

2.2 OpenID

OpenID is a decentralised user authentication scheme supporting remote single sign-on to multiple websites using a single digital identity. Two 'major' OpenID versions have been released: OpenID 1.1 [2], and OpenID 2.0 [3]; fortunately v2.0 is backward compatible with v1.1. OpenID 2.0 uses two types of user identifier: URLs and XRIs. A user could adopt a self-owned URL, or register a (new) URL at an OP. OpenID operates as follows (the use of TLS/SSL is recommended).

1. UA $\rightarrow$ RP. HTTP/S request: GET (login page).
2. RP $\rightarrow$ UA. HTTP/S response. A login page is returned containing a form.
3. User $\rightarrow$ UA. The user enters their OpenID identifier into the OpenID form.
4. RP: OP Discovery. The RP uses the identifier to discover the user's OP.

[1] During card creation, a card ID and master key are created and stored.

   - **HTML-based discovery (OpenID 1.1/2.0)**. The RP requests an HTML document identified by the user's OpenID URL; such a document contains the information necessary to discover the required OP.
   - **XRDS-based discovery (OpenID 2.0)**. The RP requests an XRDS document containing the data necessary to discover the OP. If the user's identifier is an XRI, the RP will retrieve an XRDS document identified by the XRI; if it is a URL, the RP will use Yadis to retrieve an XRDS document, and, if this fails, the RP will revert to HTML-based discovery.
5. RP $\rightleftharpoons$ OP (optional). The RP and OP agree a shared secret key to be used for a specified period of time to MAC-protect exchanged messages. This (back-channel) request-response process, known as the 'association' mode, requires the two parties to be able to store the secret.
6. RP-OP Interaction. The RP and OP can communicate in either 'checkid_immediate' mode, involving direct RP-OP communications without user interaction, or 'checkid_setup' mode, where the user is interactively involved. The 'checkid_setup' mode is more commonly used; if 'checkid_immediate' mode fails, the scheme reverts to 'checkid_setup' mode. If using 'checkid_immediate' mode, the RP directly sends the OP an OpenID authentication request, and the OP directly replies with an OpenID authentication response; step 9 then takes place. However, if using 'checkid_setup' mode, the RP redirects the user to the OP with an OpenID authentication request, and step 7 follows.
7. OP $\rightleftharpoons$ User. If necessary, the OP authenticates the user. If successful, the OP constructs an OpenID assertion token, including user credentials/attributes, a freshly-generated nonce, a current time-stamp, and a MAC computed on the token. If a shared key was agreed in step 5, the OP uses it to generate the MAC; otherwise the OP employs an internally-generated MAC key. The OP requests permission to send the assertion token to the requesting RP.
8. OP $\rightarrow$ UA $\rightarrow$ RP. The OP redirects the user back to the RP with a positive or negative OpenID authentication response.
9. RP $\rightarrow$ User. The RP verifies the MAC-protected OpenID authentication response, and, if satisfied, grants access. If a shared secret was previously agreed (see step 5), the RP uses its copy to verify the MAC. If not, the RP must make an extra request to the OP to verify the MAC, typically via a TLS/SSL channel. This request-response process is known as the 'check_authentication' mode, and is adopted in the integration scheme.

3 The Integration Scheme

The parties involved are a CardSpace-enabled RP, a CardSpace-enabled UA (e.g. a suitable web browser), an OP, and a browser extension implementing the protocol described below. The scheme has the following operational requirements.

- The user must have accounts with the CardSpace RP and the OP (thus the OP can authenticate the user). The RP and the user must trust the OP.
- Prior to use, the user must create a personal card, referred to here as an IDcard, containing the following data items in specific fields (the choice of

which is implementation-specific): the user's OpenID identifier; a predefined sequence of characters (e.g. 'OpenID') used to trigger the browser extension and indicate which OpenID version to use; and the OP URL.

– The CardSpace RP must not employ an STS; instead, it must express its policy using HTML/XHTML, and selector-RP interactions must be based on HTTP/S via a browser (a simpler and probably more common scenario). This is because the scheme uses a (JavaScript-based) browser extension, and is incapable of managing the communications with an STS.
– The RP must accept an unsigned SAML token which includes OP-asserted attributes and the signed RSTR containing the card-RP-specific PPID.

The novel protocol operates as follows. Steps 1, 2, and 4–7 are the same as steps 1, 2, and 3–6, respectively, of the personal card protocol given in section 2.1.

3. Browser extension $\rightarrow$ UA. The extension performs the following steps.
   (a) It scans the login page to detect whether the RP website supports CardSpace; if so, it proceeds, otherwise it terminates.
   (b) It examines the RP policy to check whether the use of personal cards is acceptable. If so, it proceeds; otherwise it terminates, giving CardSpace the opportunity to operate normally.
   (c) It keeps a local copy of any RP-requested claims.
   (d) It modifies the policy to include the claim types employed in the IDcard.
8. Selector$\rightarrow$ Browser Extension. Unlike in the 'standard' case, the RSTR does not reach the RP; instead the extension intercepts it and temporarily stores it. It then determines the communication protocol (HTTP or HTTPS) in use[2]. If the RP uses HTTP, the extension uses the RSTR's contents to construct an OpenID authentication request[3], which it forwards to the appropriate OP, having discovered its address from the RSTR. On the other hand, if the RP uses HTTPS, the browser extension:
   (a) asks the user to enter his/her OpenID identifier and uses the supplied OpenID identifier to perform OP discovery (see section 2.2); and
   (b) constructs an OpenID authentication request (precisely as in the HTTP case), which it forwards to the discovered OP.

   In both cases the format of the OP authentication request will depend on the version of OpenID being used. In both cases the more commonly used OpenID 'check_setup' mode is adopted (the 'check_immediate' mode is not supported as it requires back-channel RP-IdP communication).
9. OP $\rightleftharpoons$ User. If necessary, the OP authenticates the user. If successful, the OP requests permission to send the OpenID token to the RP return-page.

[2] The protocol operates slightly differently depending on whether the RP uses HTTP or HTTPS. This is because, if HTTPS is used, the selector will encrypt the RSTR using the site's public key, and the browser extension does not have access to the corresponding private key. Hence, it will not know whether to trigger the integration protocol, and will be unable to obtain the user's OpenID identifier; such issues do not occur if HTTP is used since the selector will not encrypt the RSTR.

[3] This will indicate the RP-requested attributes which are to be asserted by the OP.

10. OP → UA → RP. The OP redirects the UA back to the RP return-page with a positive or negative OpenID authentication response[4] depending on whether or not the user granted permission in step 9.
11. Browser Extension → UA. The extension verifies the MAC-protected OpenID token by interacting with the OP using the 'check_authentication' mode via a TLS/SSL channel. If successful, it constructs a CardSpace-compatible SAML token[5], and forwards it to the RP. If unsuccessful, the extension terminates.
12. RP → User. The RP verifies the SAML token (including verifying the RSTR signature, PPID, nonce, time-stamps, etc.), and, if satisfied, grants access.

4 Security Analysis

The unsigned browser extension-generated SAML token (referred to as the 'user token') includes the PPID, the OP-asserted user attributes, the signed SIIP-issued RSTR, and (optionally) the MAC-protected OP-issued token. The RP compares the SIIP-asserted PPID (and the public key) in the user token with its stored values and verifies the digital signature. The RP can optionally verify the MAC in the OP-issued token, which is embedded unchanged in the user token; for the RP to verify the MAC, the extension must skip the verification process and the RP must interact with the OP via the 'check_authentication' mode.

It is infeasible for a malicious entity to fabricate a user token to masquerade as a legitimate party since it will not have access to three key token components: the PPID; the SIIP-signed RSTR, which is only issued if the appropriate InfoCard is selected on the correct platform; and the MAC-protected OP-issued token, which is only issued if the genuine user has been authenticated by the OP. In addition, nonces and time-stamps are used to prevent replay attacks, and RPs can also employ IP address validation. The use of SSL/TLS on the OP-client and RP-client channels is strongly recommended.

The selector identifies the RP to the user and indicates whether or not they have visited that particular RP before; if this RP is being visited for the first time, CardSpace requests the user's permission to proceed. This helps to support mutual authentication since the user and the RP are both identified to each other (a security gain over 'native' OpenID, which does not identify the RP).

The scheme mitigates the risk of phishing, because the redirect to the OP[6] is initiated by the browser extension and not by the RP, i.e. the RP cannot redirect the user to an OP of its choosing. By contrast, in OpenID a malicious RP could redirect a user to a fake OP, which might capture the user credentials.

Finally, the scheme allows the user attributes to be remotely stored at the OP; this has potential security advantages over storing the attributes locally on the user machine, as is currently the case with CardSpace SIIP-issued attributes.

[4] The RP will receive the OP-issued token unchanged (embedded in the URL); however it is assumed that the RP will ignore it because of its inability to process the token.

[5] Observe that this (unsigned) SAML token will contain the user attributes as asserted by the OP and the digitally-signed SIIP-issued RSTR (which contains the PPID).

[6] In HTTP mode the OP address is retrieved from the IDcard as entered by the user.

5 Related Work

A somewhat similar scheme [4] has been proposed to support CardSpace-Liberty interoperation. However, unlike the scheme proposed here, the CardSpace-Liberty integration scheme is not transparent to the IdPs, does not support the exchange of identity attributes, and does not support HTTPS-enabled websites.

Kim et al. [5] have proposed an OpenID authentication method using an identity selector. The scheme uses a specially modified identity selector to enable OpenID authentication, unlike our scheme which uses an unmodified selector.

Microsoft and OpenID have announced plans (`http://www.guardian.co.uk/technology/blog/2007/feb/07/openidgetsab`) to enable a level of interoperation. A stated aim of this effort is to reduce the risk of phishing in OpenID by enabling an OpenID user to employ CardSpace when authenticating to an OP. The scheme proposed here inherently protects against phishing (see section 4), and also supports the use of CardSpace to authenticate to OPs.

6 Conclusions

We have proposed a means of interoperation between two leading identity systems, namely CardSpace and OpenID. CardSpace users are able to obtain an assertion token from an OpenID provider, the contents of which can be processed by a CardSpace-enabled relying party. The scheme is transparent to OpenID providers and identity selectors, uses a browser extension, and requires only minor changes to a CardSpace relying party. It uses the selector interface to integrate OpenID providers with CardSpace relying parties, using personal cards.

The integration scheme takes advantage of the similarity between the OpenID and the CardSpace frameworks, and this should help to reduce the effort required for full system integration. A full version of this paper, including a description of a prototype implementation, is available [6].

References

1. Mercuri, M.: Beginning Information Cards and CardSpace: From Novice to Professional. Apress, New York (2007)
2. Recordon, D., Fitzpatrick, B.: OpenID Authentication 1.1. (2006) `http://openid.net/specs/openid-authentication-1_1.html`.
3. OpenID Community: OpenID Authentication 2.0 — Final. (2007) `http://openid.net/specs/openid-authentication-2_0.html`.
4. Al-Sinani, H.S., Alrodhan, W.A., Mitchell, C.J.: CardSpace-Liberty integration for CardSpace users. In Klingenstein, K., Ellison, C.M., eds.: Proceedings of the 9th Symposium on Identity and Trust on the Internet, (IDtrust'10), Gaithersburg, Maryland, USA, April 13–15, 2010, ACM, New York, NY, 12–25 (2010)
5. Kim, S.H., et al.: OpenID Authentication Method Using Identity Selector. United States, Patent Application Publication, Pub. No. US 2009/0249078 A1. (2009)
6. Al-Sinani, H.S., Mitchell, C.J.: CardSpace-OpenID Integration for CardSpace Users. Technical Report: RHUL–MA–2011–12 (Department of Mathematics, Royal Holloway, University of London). (2011)