Using CardSpace as a Password Manager

Haitham S. Al-Sinani and Chris J. Mitchell

Information Security Group
Royal Holloway, University of London
http://www.isg.rhul.ac.uk
[H.Al-Sinani, C.Mitchell]@rhul.ac.uk

Abstract. In this paper we propose a novel scheme that allows Windows CardSpace to be used as a password manager, thereby improving the usability and security of password use as well as potentially encouraging CardSpace adoption. Usernames and passwords are stored in personal cards, and these cards can be used to sign on transparently to corresponding websites. The scheme does not require any changes to login servers or to the CardSpace identity selector and, in particular, it does not require websites to support CardSpace. We describe how the scheme operates, and give details of a proof-of-concept prototype. Security and usability analyses are also provided.

1 Introduction

The most common means of user authentication remains the use of passwords, despite their well-known shortcomings. Moreover, as the number of on-line services requiring passwords continues to grow, users increasingly re-use passwords, write them down in insecure ways, and/or employ passwords which can be readily guessed. The result is an ever-increasing risk of exposure of passwords to malicious parties. Users also risk having their passwords stolen [1, 2] through key logging, phishing, sniffing, shoulder surfing, etc.

A solution that enables the use of site-unique strong passwords whilst maintaining user security and privacy is thus needed. Password managers of various types have been proposed to meet this need [3]. A password manager stores usernames and passwords and makes them available when required. Users are not required to remember any passwords apart from a single master password which can be used to lock/un-lock the password manager. Password managers can be particularly helpful when a relatively large number of passwords are required to access multiple on-line services. Password managers can be seen as potential alternatives to single sign-on systems such as Windows Live ID, formally known as Passport (`http://passport.net/`), and OpenID (`http://openid.net/`).

Windows CardSpace is a user-friendly tool supporting user authentication. Instead of providing a username and password, a CardSpace user selects a virtual card, known as an information card (InfoCard), from an intuitive user interface provided by the CardSpace identity selector (CIdS), to sign on to a website.

Despite the introduction of CardSpace (and other similar systems), the vast majority of websites still use username-password for authentication, and this is

likely to continue for at least the next few years [2]. One major problem with CardSpace, and with other similar systems providing more secure means of user authentication, is that the transition from username-password to full-blown identity management is extremely difficult to achieve. Service providers (SPs) will not wish to do the work necessary to support CardSpace if very few users employ it; equally, users are hardly likely to use CardSpace if it is only supported by a small minority of websites. The scheme we describe is designed to help overcome this barrier to change by allowing an evolutionary deployment of CardSpace, initially as a password manager and subsequently, once users are familiar with its interface, as a more sophisticated means of user authentication.

In this paper we propose a simple scheme that uses the CIdS as a password manager. The goal is to develop a simple and intuitive approach to password management, transparent to both the CIdS and SPs. The technique we describe works with existing servers without any modifications, and, in particular, SPs are not required to support CardSpace.

The remainder of the paper is organised as follows. Section 2 presents an overview of CardSpace, and Sect. 3 describes the proposed scheme. We describe a prototype implementation in Sect. 4, and, in Sect. 5, we provide security and usability analyses. Sect. 6 reviews related work and Sect. 7 concludes the paper.

2 CardSpace

This section gives a brief introduction to CardSpace, including a description of the use of CardSpace personal cards.

2.1 Introduction

Microsoft CardSpace is an identity management system that provides a secure and consistent way for users to control and manage personal data, to review personal data before sending it to a website, and to verify the identity of visited websites. It also enables websites to obtain personal information from users, e.g. to support user authentication and authorisation.

The CIdS enables users to manage digital identities issued by a variety of identity providers (IdPs), and use them to access on-line services. Digital identities are visually represented to users as InfoCards, and are implemented as XML files that list the types of claim made by one party about itself or another party.

The concept of an InfoCard is inspired by real-world cards, e.g. credit cards. Users can employ one (virtual) InfoCard to identify themselves to multiple websites. Alternatively, separate InfoCards can be used in distinct situations. Websites can request different types of cards and/or different types of claims.

There are two types of InfoCards: personal (self-issued) cards and managed cards. Personal cards are created by users themselves, and the claims listed in such an InfoCard are asserted by the self-issued identity provider (SIP) that co-exists with the CIdS on the user machine. Managed cards, on the other hand, are obtained from remote IdPs [4–8].

By default, CardSpace is supported in Internet Explorer (IE) from version 7 onwards. Extensions to other browsers, such as Firefox[1], and Safari[2] also exist. Microsoft has recently released an updated version of CardSpace, known as Windows CardSpace 2.0 Beta 2[3]. However, in this paper we refer throughout to the CardSpace version that is shipped by default as part of Windows Vista and Windows 7, which has also been approved as an OASIS standard under the name 'Identity Metasystem Interoperability Version 1.0' (IMI 1.0) [9].

2.2 Personal Cards

The core idea introduced here is to use CardSpace personal cards to enable users to seamlessly authenticate to websites using stored passwords. We therefore first describe how personal cards are used.

Overview. Prerequisites for use of a personal card include a CardSpace-enabled relying party (RP) and a CardSpace-enabled user agent, e.g. a web browser capable of invoking the CIdS. The CIdS enables the creation of personal cards, which currently support 14 editable claim types, namely *First Name, Last Name, Email Address, Street, City, State, Postal Code, Country/Region, Home Phone, Other Phone, Mobile Phone, Date of Birth, Gender*, and *Web Page*.

Using Personal Cards. When using personal cards, CardSpace adopts the following protocol. We assume here that the RP does not employ a security token service (STS)[4].

1. User agent → RP. HTTP/S request: GET (a login page).
2. RP → user agent. HTTP/S response. A login page is returned containing CardSpace-enabling tags in which the RP security policy is embedded.
3. The user agent offers the user the option to use CardSpace (e.g. via a button on the RP web page), and selection of this option causes the agent to invoke the CIdS and pass it the RP policy. Note that if this is the first time that this RP has been contacted, the CIdS will display the RP identity and give the user the option to either proceed or abort the protocol.
4. After evaluating the RP policy, the CIdS highlights the InfoCards matching the policy, and greys out the rest. InfoCards previously used for this particular RP are displayed in the upper half of the selector screen.
5. The user chooses a personal card. (Alternatively, the user could create and choose a new personal card). At this point the user can check the requested claim types and decide whether or not to proceed. Note that the selected InfoCard may contain several claims, but only the claims explicitly requested in the RP security policy will be passed to the requesting RP.

[1] `https://addons.mozilla.org/en-US/firefox/addon/10292`
[2] `http://www.hccp.org/safari-plug-in.html`
[3] `http://technet.microsoft.com/en-us/library/dd996657(WS.10).aspx`
[4] An STS is a software component responsible for security policy and token management within an IdP and, optionally, within an RP.

6. The CIdS creates and sends a SAML-based request security token (RST) to the SIP, which responds with a SAML-based request security token response (RSTR).
7. The RSTR is then passed to the user agent, which forwards it to the RP.
8. The RP validates the received SAML token, and, if satisfied, grants access to the user.

Details of how CardSpace managed cards are used are given in the relevant specifications [4–6, 9].

3 A CardSpace-based Password Manager (CPM)

We next give an overview of the CPM, covering relevant operational aspects.

3.1 Browser Extension

The CPM is based on a browser extension that can read and modify browser-rendered web pages. It can also read CardSpace-issued RSTR tokens. In addition it can automatically fill and submit login forms, start automatically, and be enabled or disabled by the user.

3.2 PassCards

Either prior to, or during, use of the CPM, the user must first create a personal card, referred to here as a PassCard, containing a username and password. Basic protection against phishing is provided if the URL of the target website is included in the PassCard. However, this is optional, as users may wish to use a single PassCard with multiple websites sharing the same user credentials.

The browser extension is responsible for adding the PassCard logo (see Fig. 3), a modified version of the CardSpace logo, to the SP web page, enabling the user to invoke the CIdS and to subsequently select (or create) a PassCard.

3.3 System Parties

The parties involved are as follows.

- An SP, i.e. a website that the user is currently visiting.
- A CardSpace-enabled user agent (e.g. a suitable web browser, such as IE8).
- A browser extension implementing the protocol described below.

3.4 Message Flows

The protocol steps, as shown in Fig. 1, are as follows.

1. User agent → SP. HTTP request: GET (a login page).
2. SP → user agent. HTTP response: (the login page is returned).

3. The browser extension performs the following processes using the login page provided by the SP.
   (a) It scans the page for a login form containing a username and password field and a submit button.
   (b) If all of these are found, it highlights the username and password fields.
   (c) It adds CardSpace-enabling tags to the login page, setting the associated security policy to require a token asserting claims of the types in which the user credentials are stored (see Sect. 3.5 for a discussion of what occurs when interacting with SPs that already support CardSpace.)
   (d) It adds a function to the login page to intercept the RSTR that will later be returned by the CIdS.
   (e) It causes the PassCard logo to appear above the submit button, in such a way that clicking it invokes the CIdS.
4. The user clicks on the PassCard logo and the CIdS lights up. If this is the first time that this SP has been contacted, the CIdS will display the SP identity and give the user the option to either proceed or abort the protocol. On proceeding, the CIdS highlights the InfoCards that match the policy statement added by the browser extension, and greys out the rest. InfoCards previously used for this SP are displayed in the upper half of the selector screen.
5. The user selects and submits a PassCard. Alternatively, the user could create and choose a new PassCard. The CIdS creates and sends a SAML-based RST to the SIP, which responds with a SAML-based RSTR.
6. The CIdS passes the RSTR to the browser.
7. The browser extension performs the following tasks.
   (a) It intercepts and parses the RSTR.
   (b) If the RSTR contains the URL of the target website, it compares it with the URL of the visited website, and only proceeds if they match.
   (c) It extracts the username and password values.
   (d) It auto-populates and auto-submits the login form.
8. The SP verifies the credentials, and, if satisfied, grants access.

Note that the CPM user experience is precisely the same to that of 'conventional' password-based authentication except that, instead of manually entering and submitting a username and password, the CPM user selects and submits a virtual card.

3.5 CardSpace-enabled SPs

Regardless of whether or not an SP already supports CardSpace, the browser extension will always add the PassCard logo to the SP web page, as long as it detects username-password prompts on the page. This means that, if an SP supports CardSpace and simultaneously supports username-password authentication, as does the 'myOpenID' website (`https://www.myopenid.com/` [sampled on 5/1/2010]), the browser extension will still insert the PassCard logo above the submit button of the password-based login form. Informal tests on the prototype

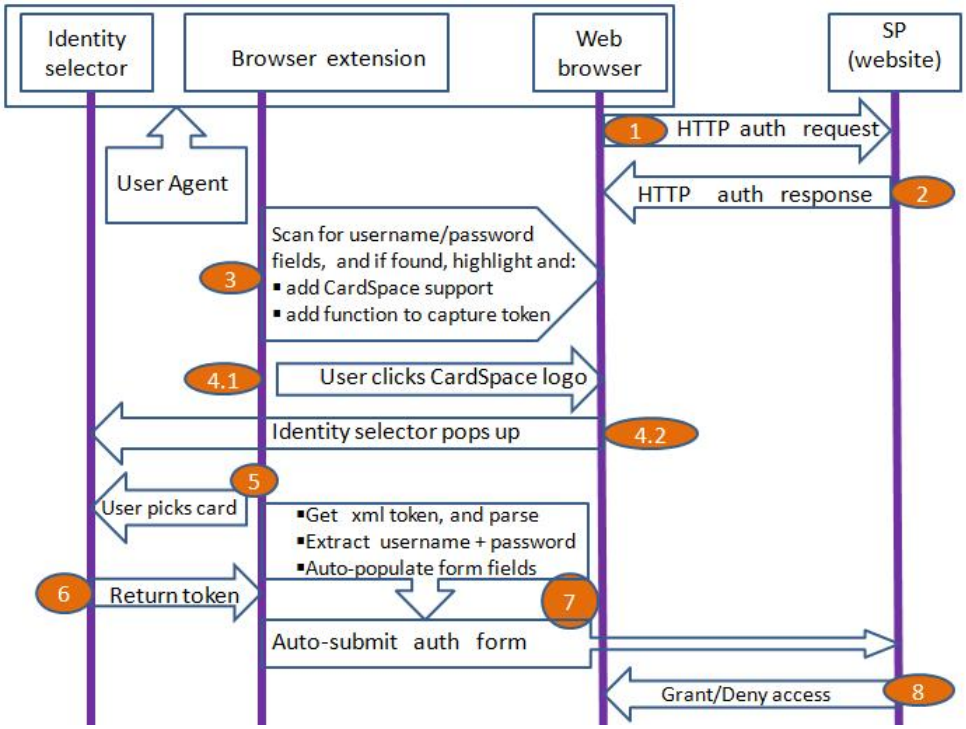

Fig. 1. CPM message exchanges

implementation (see Sect. 4) suggest that this will not disrupt the normal operation of CardSpace; indeed, the browser extension does not modify the original SP-CardSpace-enabling tags in any way.

The SP page will thus display the CardSpace and PassCard logos. In this case, users will have (at least) three login options:

1. populating the username and password fields and submitting the login form manually;
2. using the CPM to automatically populate and submit the login form; or
3. clicking the CardSpace logo to use CardSpace-based authentication.

Note that hovering the mouse over the PassCard logo results in the display of text indicating that clicking it will activate the CPM.

4 Prototype Realisation

We next give details of a prototype implementation of the scheme.

The prototype is coded as a plug-in[5] using JavaScript [10], chosen because its wide adoption should simplify the task of porting the prototype to a range of other browsers. It uses the Document Object Model (DOM) [11] to inspect and manipulate HTML pages and XML documents. We plan to publish the CPM prototype as an open source research project.

4.1 Registration

Prior to, or during, use of the CPM, the user invokes the CIdS and creates a PassCard, inserting their username in the *firstname* field and password in the *lastname* field. Optionally, the user could also insert the URL of the target website in the *webpage* field[6] (see Sect. 3.2). For ease of identification, the user can give the PassCard a meaningful name, e.g. of the corresponding website. The user can also upload an image for the PassCard, e.g. containing the logo of the intended site. Example PassCards are shown in Fig. 2.

Fig. 2. PassCards

4.2 Implementation

The prototype implements the message exchanges specified in Sect. 3.4; we refer throughout to the step numbers given there.

Prototype-specific Implementation Details. In step 3, the plug-in processes the SP web page in the following way.

[5] The term 'plug-in' is used here to refer to any client-side browser extension, such as a user script, plug-in, etc.

[6] The *web page* field was chosen to contain the URL of the target website since it seemed the logical choice; however, this is an implementation option.

1. The plug-in scans the web page for a login form containing a pair of username and password fields and a submit button. More specifically, the following procedure is used.
   (a) The plug-in scans the web page for a form tag.
   (b) If a form tag is found, it searches the form for three input tags referring to username, password, and submit, as follows:
      i. it searches for an input tag of type 'text';
      ii. if found, it searches for another input tag of type 'password'; and
      iii. if found, it searches for another input tag of type 'submit'[7].
   (c) If all three fields are detected, then the plug-in highlights the username and password fields in green for ease of identification.

   The above process involves the following detailed processing.
   - Highlighting does not take place unless both the username and password fields and the submit button have been detected in a single form, as a web page could potentially contain more than one input tag of type 'text', such as those used for searching.
   - To differentiate between registration and login web pages, the plug-in terminates if it detects more than one password field between the username and submit fields. Whereas it appears common for a login page to only have a single password field before the submit button, registration pages typically have two password fields (before the submit button): the first for the user to enter their password, and the second to confirm their password. Examples include the registration and login pages hosted by major websites, such as Google, YouTube, Yahoo, Microsoft Research, SpringerLink[8], etc.
   - When searching for the form submission button, if no submitting input tag has been found the plug-in searches for an 'image' tag. This is because, instead of a submit button, some websites display a clickable image[9] with similar functionality.
   - Whereas it appears common for a username field to be immediately followed by a password field, a submit button may not always immediately follow a password field. For example, some major websites (e.g. Google, YouTube, Yahoo, Gmail, SpringerLink) add a 'Stay signed in' or 'Remember me' check box between the password field and the submit button[10]. The plug-in addresses this issue by skipping all tags between the password field and the submit button, including those of type 'checkbox'.
2. The plug-in adds an HTML object tag that allows the user to invoke the CIdS. Within the object tag, the plug-in sets the param tags to indicate that

[7] If no input tag of type 'submit' is found, the plug-in searches for an input tag of type 'image' and, if still not found, it then searches for an event-based input tag of type 'button'.

[8] Websites most recently checked on 5/1/2010.

[9] This includes an image tag embedded in a hyperlink (anchor) tag, an image tag on its own, an image tag embedded inside a button tag, an event-based button tag, etc.

[10] Websites most recently checked on 5/1/2010.

the SP security policy requires PassCards to contain two fields: the *firstname* and the *lastname* fields, (or three fields if protection against phishing is required, in which case the third field would be the *web page* field). Alternatively, the security policy could be configured so that the *web page* field is optional.

3. The plug-in adds a function to the head section of the SP login page to intercept the XML-based RSTR message returned by the CIdS.
4. The plug-in inserts the PassCard logo in the login web page, causing it to appear just before the 'login' button, as shown in Fig. 3. The logo is associated with an 'on-click' event so that, if clicked, the CIdS is invoked (after calling the added function). To cater for users with sight difficulties or web browsers configured not to display images, a text field can replace the logo. This text is also displayed when the mouse is held over the PassCard logo, indicating that the CPM can be used to sign on.

After step 6, the plug-in performs the following steps.

1. It intercepts the XML-based RSTR using the added function.
2. It parses the intercepted token and extracts the values of the *firstname* and *lastname* fields. The plug-in uses an XML parser built into the browser to read and manipulate the intercepted XML token. The plug-in passes the token to the parser, which reads it and converts it into an XML DOM object that can be accessed and manipulated by JavaScript.
3. If a URL is stored in the PassCard, it compares the stored URL with the URL of the visited website, and only proceeds if they match.
4. It automatically fills in the username and password fields with the *firstname* and *lastname* values, respectively.
5. It auto-submits the login form using the JavaScript 'click()' method.

Fig. 3. PassCard logo

Coding Environment. The JavaScript-driven plug-in was built using IE7PRO, an IE extension, chosen to simplify prototype implementation. Users of the prototype must therefore install IE7PRO, freely available at the IE7PRO website (`http://ie7pro.com`), prior to installing the CPM plug-in. To enable or disable the prototype, a user can simply tick or un-tick the appropriate entry in the 'IE7PRO Preferences' interface, thus meeting the final objective listed in Sect. 3.1.

An IE7PRO-free version has also been produced. In this latest version, the JavaScript code is executed using a C#-driven browser helper object (BHO), a Dynamic-link library (DLL) module designed as a plug-in for IE that, once installed, attaches itself to IE, thus gaining access to the current page's DOM. The CPM prototype can readily be enabled or disabled using the add-on manager in the IE *Tools* menu.

5 Discussion and Analysis

We now consider a number of CPM features, and also review certain limitations.

5.1 CPM Features

Security. The CPM takes advantage of the CIdS, and is supported by its built-in security features. For example, when invoked, the CIdS runs in a separate private desktop session, preventing other applications, e.g. malware, from interacting or interfering with it. In addition, all values inserted in the fields of a PassCard are stored in encrypted form on the user machine.

Furthermore, in protocol step 4 of the CPM, the CIdS identifies the SP to the user and indicates whether or not they have visited that particular SP before; if the user is visiting this SP for the first time, CardSpace requests the user's permission to proceed[11] (see section 3.4). This helps to support mutual authentication since the user and the SP are both identified to each other.

As with any local password manager, the CPM avoids the need for trusted third parties. In addition, the automatic form-filling feature reduces exposure to shoulder-surfing attacks and also helps to thwart key loggers.

The CPM reduces the threat of phishing attacks involving impersonation of legitimate sites by comparing the URL in the PassCard with that of the visited website. The CPM also supports the use of strong per-site passwords, since users no longer need to memorise or write down passwords.

Finally, the CPM browser extension does not require any changes to default IE security settings, thus avoiding potential vulnerabilities resulting from lowered browser security settings.

[11] This enhances security by comparison with 'conventional' password-based authentication, where the SP is not identified to the user.

Usability. The CPM provides a simple, intuitive user experience through its use of the CIdS interface. At the same time, it familiarises users with CardSpace, thereby potentially facilitating future adoption of more secure means of authentication. Unlike other password managers which represent credentials in text form, CPM credentials are stored in PassCards which can be equipped with a readily recognisable image, e.g. an SP logo.

The CPM operates completely transparently to external parties, and hence does not require any changes to SPs, identity selectors or to default browser security settings. The scheme is also highly flexible, since users can choose whether or not to use it simply by electing to click the PassCard logo (or not).

Finally, by making use of CardSpace features, the CPM supports a degree of roaming. A user can transfer PassCards from one PC to another using the CardSpace backup facilities. Indeed, if the CardSpace backup file (which holds data in encrypted form) is stored on a portable storage medium (e.g. a USB drive) then full mobility is provided, as well as robustness in the form of protection against loss of credential data.

5.2 Limitations and Countermeasures

Perhaps the most obvious limitation of the CPM is that anyone with access to a Windows user account can access the PassCards and use the stored credentials. This is a fundamental limitation of CardSpace which, by default, does not impose any additional password protection on the use of the CIdS. To address this issue, we observe that CardSpace allows individual InfoCards to be PIN-protected, which should be seriously considered for PassCards stored on a machine which is not physically secure. In addition, it may be possible to cause CardSpace to run under User Account Control (UAC), so that running CardSpace causes Windows to prompt the user for an admin password.

The version of the CPM prototype described in this paper does not work as intended with websites employing TLS/SSL encryption because, if the SP has a certificate, then the CIdS will, by default, encrypt the SAML-based RSTR message using the public key of the requesting SP. The plug-in does not have access to the SP's private key, and hence will be unable to decrypt the token. As a result, the CPM will not be able to perform automatic form-filling as it cannot obtain the username/password values. However, because the CPM (if clicked) automatically invokes the CIdS, users could manually copy and paste the required credentials from the relevant PassCard into the login form. This is particularly simple if only a password is needed.

A more satisfactory solution to this issue would be to configure the CPM to redirect the user to an (arbitrary) HTTP-based website whenever an HTTPS-based website is visited. In this case, the CPM could read the SAML token returned from the CIdS as it would not be encrypted. The CPM could then submit the credentials automatically to the target HTTPS-based site. The transfer could be achieved using hidden HTML form variables or URL query parameters. We are currently developing a prototype to realise this.

The browser extension must scan every browser-rendered web page to detect whether it supports username-password authentication, and this may affect system performance. However, informal tests on the CPM prototype suggest that this is not a serious issue. In addition, the browser extension can be configured so that it only operates with certain websites.

The current CPM prototype has not yet been tested with the recently released CardSpace 2.0[12]. We are thus unable to provide precise operational details for this version.

Finally, older browsers (or browsers with scripting disabled) may not be able to run the CPM, as it was built using JavaScript. However, most modern browsers support JavaScript (or ECMAscript), and so this is not a major usability obstacle.

6 Related Work

Password managers, which store passwords in a secure location either on the user PC or remotely, are now widely available. They typically store passwords in encrypted form and, unlike the CPM, require users to use a single master password to access the password store. Some are also capable of masking passwords, and others, much like the CPM, provide automatic password entry. Examples of password managers include open source schemes such as Password Safe (`http://passwordsafe.sourceforge.net/`), KeePass (`http://KeePass.info/`), Qubliette (`http://tranglos.com/free/oubliette.html`), Password Gorilla (`http://fpx.de/fp/Software/Gorilla/`) and PINs (`mirekw.com/winfreeware/pins.html`) as well as commercial products such as RoboForm (`http://roboform.com/`), Any Password (`http://anypassword.com/`) and Turbopasswords (`http://chapura.com/passwordmanager.php`).

Perhaps the most distinctive feature of the CPM is its dependence on CardSpace, whereas other password managers are independent applications. As a result, the CPM can benefit significantly from the CardSpace security features. The CPM's use of CardSpace may also give users greater confidence in its security features. Most importantly, it is hoped that the introduction of a scheme like the CPM, with immediate practical benefits to the end user, will help encourage the adoption of more sophisticated identity management systems like CardSpace. Such identity management schemes offer the potential for a step forward in the practice of user authentication and authorisation, with potential benefits for all legitimate parties operating via the Internet. Indeed, without simple paths to adoption for schemes like CardSpace, there is a danger that it and all the other identity initiatives will fail.

[12] CardSpace 2.0 has not yet been standardised; in fact, at the time of writing, CardSpace 2.0 has only been released as a *Beta* prototype.

7 Conclusions and Future Work

In this paper we have proposed a novel scheme that enables CardSpace to be used as a password manager. Users store their usernames and passwords in CardSpace personal cards, and use such cards to transparently sign on to corresponding websites. The scheme is based on a browser extension, and requires no changes to login servers; in particular, it does not require them to support CardSpace. Neither does the scheme require any changes to the current CIdS.

The scheme uses the CIdS interface to seamlessly authenticate users to websites. It extends the use of personal cards to allow for transparent password management. Such an approach could help to extend the applicability of CardSpace, as well as encouraging its adoption.

Planned future work includes building a portable version of the CPM to support users who do not have installation privileges or are forced to use untrusted machines when travelling. In addition, we plan to investigate the possibility of using CardSpace as a password-based single sign-on system.

Acknowledgements

The first author is sponsored by the Diwan of Royal Court, Sultanate of Oman. The helpful remarks provided by anonymous referees are gratefully acknowledged. Finally, we would like to thank Cormac Herley for his guidance and support.

References

1. Conklin, A., Dietrich, G., Walz, D.: Password-based authentication: a system perspective. In: Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS 04) — Track 7, IEEE Computer Society, Los Alamitos, CA, USA, 70170b (2004)
2. Herley, C., van Oorschot, P.C., Patrick, A.S.: Passwords: If we're so smart, why are we still using them? In Dingledine, R., Golle, P., eds.: Financial Cryptography and Data Security. Volume 5628 of Lecture Notes in Computer Science., Springer-Verlag, Berlin, Heidelberg, Germany, 230–237 (2009)
3. De Luca, M.: Password Management for Distributed Environments. VDM Verlag, Saarbrücken, Germany (2008)
4. Jones, M.B.: A Guide to Using the Identity Selector Interoperability Profile V1.5 within Web Applications and Browsers. Microsoft Corporation. (2008)
5. Bertocci, V., Serack, G., Baker, C.: Understanding Windows CardSpace: An Introduction to the Concepts and Challenges of Digital Identities. Addison-Wesley, Reading, Massachusetts, USA (2008)
6. Mercuri, M.: Beginning Information Cards and CardSpace: From Novice to Professional. Apress, New York, USA (2007)
7. Oppliger, R., Gajek, S., Hauser, R.: Security of Microsoft's identity metasystem and CardSpace. In: Proceedings of the Kommunikation in Verteilten Systemen (KiVS 07), VDE Publishing House, Berlin, Germany, 63–74 (2007)

8. Al-Sinani, H.S., Alrodhan, W.A., Mitchell, C.J.: CardSpace-Liberty integration for CardSpace users. In: Proceedings of the 9th Symposium on Identity and Trust on the Internet (IDtrust 10), ACM, New York, NY, USA, 12–25 (2010)
9. Jones, M.B., McIntosh, M., (editors): Identity Metasystem Interoperability Version 1.0 (IMI 1.0). OASIS Standard. (2009) http://docs.oasis-open.org/imi/identity/v1.0/identity.html.
10. Powell, T.A., Schneider, F.: Javascript: The Complete Reference. 2nd edn. McGraw-Hill Osborne Media, Berkeley, CA, USA (2004)
11. Hors, A.L., Hégaret, P.L., Wood, L., Nicol, G., Robie, J., Champion, M., Byrne, S., (editors): Document Object Model (DOM) Level 2 Core Specification. W3C Recommendation. (2000) http://www.w3.org/TR/DOM-Level-2-Core/.