Privacy-enhanced Identity via Browser Extensions and Linking Services

Renato Accornero
Dipartimento di Informatica
Università degli Studi di Torino
Turin, Italy
Email: renato.accornero@di.unito.it

Daniele Rispoli
Assosecurity and
Dipartimento di Informatica
Università degli Studi di Torino
Turin, Italy
Email: rispoli@di.unito.it

Francesco Bergadano
Dipartimento di Informatica
Università degli Studi di Torino
Turin, Italy
Email: fpb@di.unito.it

***Abstract*—Identity Management systems come with a promise of simpler, centralized, and more secure handling of user data, credentials and authorizations. Service providers can thus be separated from an identity provider (IdP), and users will benefit from single sign on mechanisms. However, identity providers become single points of failure, from a security and trust perspective.**

In particular, in this paper, we address the protection of user privacy against IdPs profiling. The IdP should not be aware of which services the user will access, or about the details of such service use. This paper discusses the difficulties of this type of privacy protection problem and surveys existing solutions. We then identify a novel solution and improve an existing one: (1) moving part of the access management logic to the client, via a Web Browser extension, and (2) elaborating on previously proposed solutions based on the concept of a linking service, i.e. a server separated from the IdP and from the Service Provider, that will perform some of the authentication steps. Proof of the principle implementations of both solutions are made available to the user.

I. Introduction

Modern Identity Management (IdM) Systems [5], [13], [21], [24] provide a great opportunity to simplify the complex interaction that usually occurs between users (U) of a service and providers of that service (SP). By introducing a new entity, the Identity Provider (IdP), that hosts and guarantees a set of attributes on behalf of the user, a great deal of user-friendliness can be achieved for both parties involved in a transaction (U and SP).

The user is relieved from having to come up with a new set of credentials for each and every service she needs to use and is also spared from the tedium of having to fill endless forms with her personal data. On the other hand, the SP can rely on a trusted third party, the IdP, which, by providing and authenticating all the information needed for every authorized user, streamlines every process related with account and data management, allowing the SP to concentrate on the actual service.

The IdP is thus entrusted with a huge amount of information on any of its users: it not only knows every bit of personal data the user has supplied but also every service said user has ever been associated to. While this situation may be perfectly acceptable in certain controlled contexts such as a workplace IdM System, even providing the means with which the organization can keep track of its users' accountability, it introduces an outstanding privacy problem in a more dynamic context such as that of the Internet [11], [17], [22]. Users do not want their IdP to know every virtual step they take aggregating all of this data under the wings of a single entity that becomes too tasty a prey to be ignored by criminals, marketers and (rogue) governments alike [1], [16], [23], [26]. Several attempts to mitigate this problem can be found in the literature: [10], [12], [14], [17], [18], [19], [20], nonetheless we believe our solutions guarantee strong user privacy against this particular threat, in that the IdP does not know the SPs a user is associated to, while being minimally invasive.

In this paper we introduce two different approaches to solve this problem:

- the first is based on browser extensions ([2], [3]), in this case the association between a user and her SPs is not known by the IdP but remains only in the user's browser, as detailed in section II;
- the second uses a Linking Service (LS) (inspired by [9]) to store the redirect association thus keeping it safe from the IdP, as detailed in section III.

Overview: section II describes our user-agent mediated approach, section III explains how we use a LS to improve the privacy of an IdM system's users and section IV concludes.

II. User mediated attributes distribution

What we propose in this section is a system of User(-agent) mediated attributes distribution that keeps all the advantages illustrated above while ensuring that an IdP will be completely oblivious about the destination of the data it is authenticating on behalf of the user.

In the classic schema as shown in Figure 1 the user contacts the SP whose service she wants to use (1), the SP redirects her to the IdP of her own choosing where she can authenticate and authorize the SP to access a subset of her personal information (2). The exchange finishes, if successful, with the IdP sending the requested data to the SP along with an access token that allows the user to run the service she requested in the first place (3).

978-1-4577-0460-4/11/$26.00 ©2011 IEEE

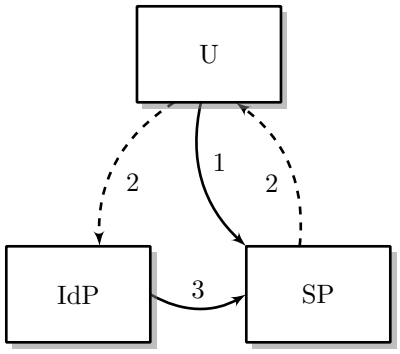

Fig. 1. The usual schema.

As noted above, since the IdP sends the data directly to the SP it can accumulate a very detailed trail of the user's activities [4]. To avoid this and protect the user's privacy against usage profiling we propose a different communication schema, as shown in Figure 2.

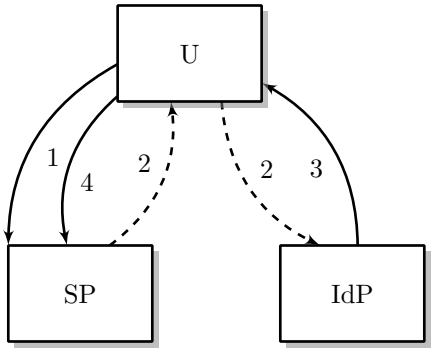

Fig. 2. Our proposed schema.

Akin to the previous usage, the user contacts the SP (1) and gets redirected to her IdP to authenticate herself (2) but, differently from what happened before, the IdP sends the access token and the personal data, signed with its own private key, back to the user (3) who will then forward it to the SP (4). By introducing this intermediate step the user can avoid to inform the IdP about the destination of her data thus enhancing her privacy.

A. Browser extension

In this approach we extend the capabilities of the browser giving it the ability to remember the association between the service a user is trying to get access to and the IdP the user chose to authenticate on. Since the extension knows where to redirect the user once she has successfully logged on, this information can be concealed from the IdP thus protecting the user's privacy against usage profiling. Both versions of the extension are used to keep this association while it is useful to the user and deleting it immediately after the authentication procedure succeeds.

1) Firefox: The first facility needed to develop our extension is a place to store the associations between a service the user is going to be redirected to once she authenticates and the identity provider the user chose to authenticate from. Since this data has to remain inside the browser we use Firefox's own components and functions[1] to retrieve a handler for an HTMLv5 localStorage object. The steps needed to accomplish this result are abstracted away in a function that takes a domain name as its only argument and uses it as the origin for the localStorage object it is requesting.

```
function get_storage(url) {
  var ios = Components.classes["@mozilla.org/network
      /io-service;1"].getService(Components.
      interfaces.nsIIOService);
  var ssm = Components.classes["@mozilla.org/
      scriptsecuritymanager;1"].getService(
      Components.interfaces.nsIScriptSecurityManager
      );
  var dsm = Components.classes["@mozilla.org/dom/
      storagemanager;1"].getService(Components.
      interfaces.nsIDOMStorageManager);

  var uri = ios.newURI(url, "", null);
  var principal = ssm.getCodebasePrincipal(uri);
  var storage = dsm.getLocalStorageForPrincipal(
      principal, "");

  return storage;
}
```

Next comes the gist of our extension: after the customary declarations

```
var idp_privacy = function () {
  var prefManager = Components.classes["@mozilla.org
      /preferences-service;1"].getService(Components
      .interfaces.nsIPrefBranch);
  return {
```

we define the listeners for the events that both the SP and IdP are going to generate as consequences of users' actions that require a change in the state of the log-in process.

The listener on the SP side, `splistener`, gets the event it was called from, `evt`, and then calls the function defined previously to obtain the storage object it needs using the `idp.privacy` as a fictitious domain name to avoid collisions with real ones.

```
    splistener : function(evt) {
      var storage = get_storage("http://idp.privacy
          ");
```

It then proceeds by storing the data it extracts from the event, a hash generated by the SP to unambiguously identify the current interaction and the address the user is going to be redirected to, into the storage object it just acquired.

```
      storage.setItem(evt.target.getAttribute("
          redirect_hash"), evt.target.getAttribute("
          sp_back_url"));
```

Finally ending this section of the protocol by redirecting the user to her chosen IdP along with the data necessary to retrieve the newly stored association once the user authenticates.

[1]https://developer.mozilla.org/en/XPCOM_Interface_Reference/NsIDOMStorageManager

```
      content.window.location = evt.target.
          getAttribute("idp_url") + "?redirect_hash
          =" + evt.target.getAttribute("
          redirect_hash");
    },
```

The other listener, `idplistener`, on the side of the IdP, performs actions that are akin to that of `splistener`: after getting its handler on the same localStorage object used previously it retrieves the address the user is going to be redirected to using the identifier it has been given from the event that triggered its execution.

```
    idplistener : function(evt) {
      var storage = get_storage("http://idp.privacy
          ");
      var sp_back_url = storage[evt.target.
          getAttribute("redirect_hash")];
```

The next section is the collection of the user's data that she chose to share with the SP and that the IdP provided. This information are simply read from the firing event and appended to the redirection address.

```
      var attrs = evt.target.attributes;
      var l = attrs.length;

      if(l > 1) {
        sp_back_url += "?" + attrs.item(0).nodeName
            + "=" + attrs.item(0).nodeValue;

        for(var i = 1; i < l; i++)
          sp_back_url += "&" + attrs.item(i).
              nodeName + "=" + attrs.item(i).
              nodeValue;
      }
```

The final steps are the redirection of the user to the SP she came from in the first place along with the data collected after the authentication

```
      content.window.location = sp_back_url;
```

and the cleanup of the association that is now no longer useful.

```
      storage.removeItem(evt.target.getAttribute("
          redirect_hash"));
    }

  };
}();
```

Finally we add two event listeners[2] that will call the functions we just defined when the need arises.

```
document.addEventListener("
    idp_privacyExtensionEvent_SP", function(e) {
    idp_privacy.splistener(e); }, false, true);
document.addEventListener("
    idp_privacyExtensionEvent_IdP", function(e) {
    idp_privacy.idplistener(e); }, false, true);
```

2) Chrome: This version of the extension works very much like to the previous one, the major difference being the usage of the localStorage object: Chrome makes it much easier for an extension to use this kind of storage without having to come up with fictitious domain names.

[2]https://developer.mozilla.org/en/DOM/element.addEventListener

Another point worth noticing is that to use this storage we have to split the extension in two parts: a JavaScript file, `contentscript.js`, for the listeners and an HTML file, `background.html`, that manages the localStorage object. The code in these two files communicate via the provided message passing infrastructure using JSON objects and a very simple protocol.

First we define the listener for the SP and the very first action it performs is to send a request to the code in `background.html` to store the data it received from the event that triggered its execution. If this operation returns a positive result, the user can be redirected to the IdP.

```
var idp_privacy = function () {
  return {
    splistener : function(evt) {
      chrome.extension.sendRequest({msg_type: 0,
          redirect_hash: evt.target.getAttribute("
          redirect_hash"), sp_back_url: evt.target.
          getAttribute("sp_back_url")},
        function(response) {
          if(!response)
            alert("Error: could not communicate with
                extension");
        });
      window.location = evt.target.getAttribute("
          idp_url") + "?redirect_hash=" + evt.target
          .getAttribute("redirect_hash");
    },
```

The other listener, like its Firefox counterpart, collects the data the user decided to share with the SP from the IdP,

```
    idplistener : function(evt) {
      var sp_back_url_opts = "";

      var attrs = evt.target.attributes;
      var l = attrs.length;

      if(l > 1) {
        sp_back_url_opts += "?" + attrs.item(0).
            nodeName + "=" + attrs.item(0).nodeValue
            ;

        for(var i = 1; i < l; i++)
          sp_back_url_opts += "&" + attrs.item(i).
              nodeName + "=" + attrs.item(i).
              nodeValue;
      }
```

and sends a request to the code in `background.html` that will perform the redirection passing along the user's data.

```
      chrome.extension.sendRequest({msg_type: 1,
          redirect_hash: evt.target.getAttribute("
          redirect_hash"), sp_back_url_opts:
          sp_back_url_opts});
    }

  };
}();
```

Finally, just like the other version, we set up the two event listeners.

```
document.addEventListener("
    idp_privacyExtensionEvent_SP", function(e) {
    idp_privacy.splistener(e); }, false, true);
document.addEventListener("
    idp_privacyExtensionEvent_IdP", function(e) {
    idp_privacy.idplistener(e); }, false, true);
```

Then we define the code that handles the localStorage object in `background.html`. As previously said the two parts of the extension communicate via JSON object: when the handler of the request receives a message of type "0" it simply stores the address the user will be redirected to once she authenticates successfully.

```
<html>
  <script>
    chrome.extension.onRequest.addListener(
      function(request, sender, sendResponse) {
        if(request.msg_type == 0) {
          localStorage[request.redirect_hash] =
              request.sp_back_url;
          sendResponse(true);
```

On the other hand, if the type of the message is "1", the action to perform is to retrieve the stored data that corresponds to the identifying hash, clean up the association since it is no longer needed and then redirect the user back to the SP she came from in the first place.

```
        } else if(request.msg_type == 1) {
          var sp_back_url = localStorage[request.
              redirect_hash];
          localStorage.removeItem(request.
              redirect_hash);
          sendResponse({});
          chrome.tabs.update(sender.tab.id, {url:
              sp_back_url + request.sp_back_url_opts
              });
      }
    });
  </script>
</html>
```

B. Server side

To prove the feasibility of this idea and show the usage of the extension we also implemented a bare-bone SP and IdP (see section II-C for a demo and the source code). The SP presents to the user a multiple choice field where she can choose her preferred IdP and doing this associates with each IdP a unique identifier in the form of a hash computed using the SP's own name, the IP address of the user, the IdP chosen and the time of generation. As soon as the user clicks on the button thus indicating her choice a JavaScript function gets called to handle the communication with the extension. This function creates a new named element in the scope of the document

```
function redirect_to_IdP() {
  var element = document.createElement("
      SPDataElement");
```

to which it adds the data that the extension uses by reading what the user chose in the selection boxes

```
  element.setAttribute("redirect_hash", document.
      getElementById('IdP').value);
  element.setAttribute("idp_url", document.
      getElementById('IdP').options[document.
      getElementById('IdP').selectedIndex].text);
  element.setAttribute("sp_back_url", "http://<?php
      echo $_SERVER['SERVER_NAME']; ?>/
      sp_after_login.php");
```

and, once all the necessary data has been retrieved, appends the element it just populated to the page's DOM.

```
  document.documentElement.appendChild(element);
```

The last step required in this interaction is to create the specific type of event that the extension awaits from the page of a SP and finally dispatch it thus triggering all of the actions explained in the preceding paragraphs.

```
  var evt = document.createEvent("Events");
  evt.initEvent("idp_privacyExtensionEvent_SP", true
      , false);
  element.dispatchEvent(evt);
}
```

Similarly, the IdP adds to its page the data that needs to be sent back to the SP via the extension and triggers the event as soon as the identity of the user is verified.

C. Implementation

Both the extensions and the server side applications were developed on a machine running Ubuntu 10.04. The web browsers on which we tested the extensions are: Mozilla Firefox©3.6.16 and various daily checkouts of the open source branch of Google Chrome©, Chromium©, up to version 12.0.740.0 (81921).

The SP and IdP were written using the language PHP, particularly version 5.3.2-1ubuntu4.7 with Suhosin-Patch (cgi-fcgi) and were running on lighttpd 1.4.26.

The extensions developed for this paper can be downloaded from:

- http://p2pg.di.unito.it/IdP_Privacy/exts/idp_privacy.xpi for the Firefox version;
- http://p2pg.di.unito.it/IdP_Privacy/exts/idp_privacy_chrome.crx for the Chrome version.

The prototype can be tested, after a successful installation of the correct extension for the browser in use, at the address: http://p2pg.di.unito.it/IdP_Privacy/sp.php while its source code can be obtained from: http://p2pg.di.unito.it/IdP_Privacy/src/www.zip.

III. Linking Service

In this section we introduce a different approach to face the issues described previously. The idea is to prevent the IdP from finding out which services are used by the user, by introducing a new element in the system called Linking Service (LS). Our solution consists in dividing the knowledge between IdP and LS preventing an entity from being able to know both authenticator and services used.

The need to introduce a new element in the system arises from the requirement to develop a solution preventing the IdP from knowing all the services accessed by a user and compatible with the principle of location independence as expressed in [4]. In a traditional web application, the state of the interaction is managed by the web server using cookies or other technologies. In a federated identity management architecture, the requirement to assure the confidence of users' information [7] has to be met by managing the state of interaction without sharing all the data with a particular entity. To achieve this goal we have to introduce a new active element in the architecture in order to increase the capability of web

browsers. The extensions described in the previous section represent an adequate solution but forces the user to install the extension on any browser used to access her services. It is important to stress that the proposed solutions do not violate the eighth law of identity (Location Independence law) expressed in [4] as a user, through the use of extensions, should be able to access services from any device. In fact, the law shows that an IdM system should allow a user to create, manage and use her identity regardless of her current location and device. To avoid this issue, the authors recommend not to rely on any persistent data stored locally on the user's machine. There are several solutions, both commercial and academic, based on the local storage of user data that violate, partially or completely, the principle of location independence: e.g., CardSpace [15], which violates it only partially since InfoCards can be transferred to one device to another, while the protocol described in [25] stores all the information locally, rendering the identity difficult to transport from one machine to another.

The law of location independence comes from the idea that authentication should be independent from the users' location. If the access to some resources is bound by the possession of specific hardware (e.g.: personal device, specific laptop, etc.), then the risks of not being able to authenticate increase. This issue, already noted in [4], is very noteworthy because if a user is not able to authenticate, then she is precluded from accessing every service.

In literature there are many different approaches to this problem, but none of them represent an adequate solution. For example, in IdM systems such as CardSpace [15] (when not operating in non-auditing mode), the user needs to give her consent to enable the transfer of information about her identity. This approach allows the user to control the flow of information, but the IdPs are still aware of each and everyone of their users' transactions. The explicit control of the information seems to be a false solution to the problem. In fact, although it may reassure the user, the IdP is aware her actions.

Alternative solutions described in the literature are based on the exclusion of the IdP from the authentication process. For example, in [4], the authors consider a solution based on a PK infrastructure. A Certification Authority (CA) identifies and certifies the user only once, but then does not have an active role in the authentication process. After obtaining a certificate, the user can perform the authentication process using a specific software installed on her machine. The use of a CA allows to verify the user's identity in a strong way but there is a problem of managing credentials. The solution proposed by CardSpace is similar, but violates the law of location independence because the selection of which identity to use is hardware and OS specific. It also provides local storage of credentials, making it difficult to authenticate from multiple locations.

Independence from location is a crucial requirement for Federated Identity Management. In fact, the possibility to perform the authentication process from any location is a necessary condition to ensure the usability of the system. For example, a user may need to access services from machines belonging to other people. If the IdM system requires the installation of additional software or the use of a personal token, the user may not be able to perform the authentication.

The use of a Linking Service allows to respect a stronger requirement: the possibility of using an IdM system on any machine with a web browser. The solution based on extensions, although it does not violate the principle of location independence [4], requires the use of properly configured browser to manage the authentication process.

For this reason we believe that the eighth law of identity, as formulated in [4], is a necessary but not sufficient requirement for the authentication process. An ideal solution should allow the use of IdM systems without installation of additional components or use of external devices. We reformulate the eight law of identity as follows:

The identity system must allow a user to create, manage, and use her identity independently of her current location, current device in use, without ad hoc device and using only standard technologies.

A solution independent from the location should be able to perform the authentication process regardless of the software or the possession of specific hardware. Further we will present a solution that can avoid this problem and allows the authentication by using a conventional web browser. This solution requires, as noted above, the use of a new service, called "Linking Service".

A. IFrame

We studied some alternative approaches to overcome this issue: one of the options we considered was a solution based on IFrame and inter-frame communication. This technique consists in an exchange of messages between different domains using the possibility of appending data to the URL. The use of cross domain messages enables the JavaScript code in an IFrame to change the URL, triggering an event defined by the main website. In our case, the IdP writes the encrypted information in the URL, thus executing an event and the corresponding call to the SP. This technique enables communication between different domains but is not compatible with all web browsers. Moreover we found security issues due to the ability of an IdP to insert malicious JavaScript code in the HTML page. For example, take in account the follow schema:

1) a user visits the web page of a Service Provider;
2) the Service Provider generates an HTML page containing an IFrame that points to the log-in page of the Identity Provider;
3) the User authenticates with Identity Provider;
4) the Identity Provider generates an encrypted payload that contains the authentication information (AI) and appends it to the URL;
5) a JavaScript event fires and the AI are read by the Service Provider.

Unfortunately, the use of IFrames introduces additional security issues. For example, the IdP could redirect the user to a malicious website, by appending a fake address to the URL. Moreover a malicious IdP could execute a phishing attack, exploiting the fact that the user is not able to verify the authenticity of the page loaded in the IFrame. The IdP could inject malicious JavaScript code in the IFrame page with the purpose of modifying the URL and redirecting the user to different services. The risk of phishing is analyzed in [4] and is considered one of the major security issues for the federated IdM systems because it may cause the unwanted diffusion of sensitive user data.

Therefore, to avoid the issues discussed above and meet the requirements listed in [6], it is necessary to define extensions (as explained in the previous section) or to introduce a new active element in the architecture. In particular, by introducing a Linking Service we build a logical bridge between IdP and SP, preventing IdPs from gathering information on the services used by the user.

B. Existing proposal based on Linking Service

The idea of introducing a third element in the architecture has been previously evaluated in [9] in order to define a service to aggregate attributes. In this paper, the authors define a federated IdM where the SP is able to acquire attributes from more IdPs. In fact, by introducing a Linking Service, it is possible to provide a transparent mechanism of aggregation that allows to retrieve attributes distributed among various IdPs.

In the system described in [9], when a user has to access a SP, she needs to select the main IdP from a predefined list. Then, once performed the authentication, the user is redirected to the Linking Service, which is responsible for querying all the other IdPs related to the user. Clearly, this solution requires an initial configuration, which is needed to notify the users' IdPs to the LS.

The LS provides transparent access to distributed attributes, but does not solve the problem of traceability of the access to the services. In fact, the IdP is aware of all services used by the user, as it must sign the user's credentials (authentication token and attributes) with the public key of the SP. However, the use of Linking Service ensures a good degree of privacy against usage profiling. In fact, each IdP is aware of only one partial identity and does not know of the existence of others identities. The Linking the service is aware of the IdPs at which a user is registered, but is unable to decipher the attributes. Furthermore, the user can directly manage the Linking Service, deciding the identity providers connected with a service provider. The IdP sends to the LS a value called LOA which states the degree of confidence in the user identity. This value is computed according to the credentials presented by the user. For example, if a user presents to the IdP a digital certificate provided by a trusted certification authority, the LOA value associated to the user will be high.

The protocol requires an initial setup stage, where the user logs into the Linking Service and manually associates IdPs with it:

1) LS shows a list of IdPs;
2) user selects an IdP from the list;
3) LS redirects the user to the selected IdP;
4) user authenticates at the IdP;
5) IdP creates a random (but permanent) identifier PIDx;
6) IdP sends the identifier, the LOA, and an authentication message to the LS;
7) LS assigns a local identifier to the user and stores locally the tuple [local identifier; PIDx; IdP address; LOA].

After the configuration, the Linking Service's association table will look like Table I.

TABLE I
AN EXAMPLE OF LINKING SERVICE TABLE

Local User ID	PID	IdP	LOA
Bob	PID=unito123	di.unito.it	1
Bob	AuthUser=ax101	bank.com	2
Bob	ClientID=bb5567	www.ex.org	1

As shown in Figure 3, the communication protocol from [9] unfolds thusly: when a user wants to access a service she contacts the SP and chooses her preferred IdP and LS from a predefined list (1). Alternatively, the choice of IdP can be made using a "Where Are You From" service. The SP redirects the user to the selected IdP (2) where she authenticates. Then the IdP sends to the SP the authentication credentials and the URL of the users' Linking Service (3). The SP asks the Linking Service to retrieve, if necessary, a user's attributes managed by other IdPs (4). While doing this, the SP may indicate a minimum level of LOA. The Linking Service checks its association table (see Table I for an example) and contacts the IdPs (5). Each IdP signs the users' credentials with its own private key and encrypts the message with the public key of the SP. Finally, the IdP sends the encrypted messages to the Linking Service (6) which sends all the attributes to the SP (7).

C. Overview of LS protocol

In this section we describe a new protocol based on a Linking Service able to prevent the diffusion of the knowledge about which services a user employs.

The protocol is transparent to the user and does not require the installation of any additional component. Unlike the solution described in [8], the IdP is not aware of the SPs visited by the user and does not need to encrypt the data with the public key of the SP. In fact, the users' data are protected using a symmetric key known only to the SP and the IdP preventing the Linking Service from deciphering the content.

The idea is to use a Linking Service to distribute users' information across multiple architectural elements instead of aggregating the attributes distributed among IdPs. In this way, the IdP knows the users' attributes but is not aware of the services used. In this way the IdP cannot analyze the behavior and usage patterns of its users.

The Linking Service is aware of the services accessed, but does not know the identity of the user. In fact, users' data are

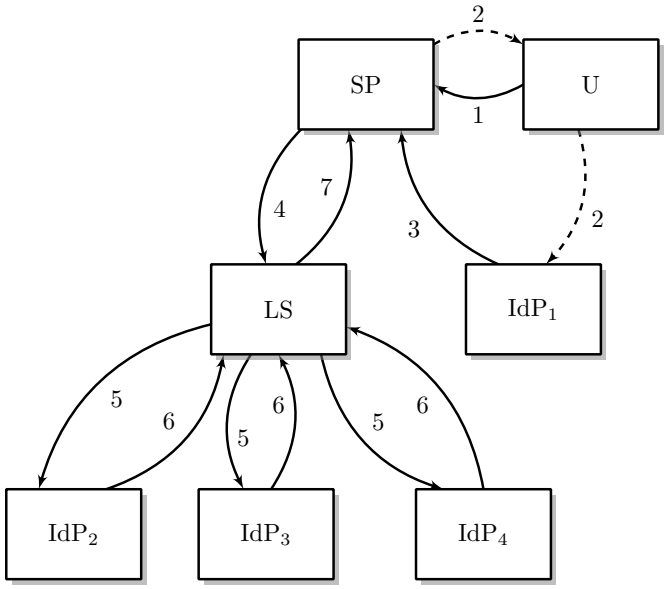

Fig. 3. Original Linking Service protocol.

transmitted encrypted and only the SP knows the symmetric key that allows to read data. The only element used by the Linking Service to redirect the user to the requested service is a random code generated by the SP which can't be related to the user's identity.

The protocol does not require an initial setup as the Linking Service's association table gets built by interacting with it.

The table stores <Nonce; URL_BACK> and the data is kept only for the time necessary to grant access to the user.

The SP trusts the source of the the attributes as the IdP signs them using its private key. In this way, the overall trust level of the system is similar to the one that can be obtained using asymmetric keys as described in [9].

The introduction of the Linking Service solves the problem of the direct connection between IdPs and SPs, and consequently reduces the risks caused by the knowledge of the services employed by a user. As shown in the previous section, because of the limited capabilities of the web browser, the use of additional services is the only way to solve this problem. A detailed description of the LS protocol follows.

As shown in Figure 4, the interaction is started by the user, who accesses the SP (1), selects the name of his IdP and a trusted LS. The SP generates a Nonce used to uniquely identify the interaction. The nonce is necessary in order to avoid replay attacks, in which a malicious user could eavesdrop the message and use the credentials to make unauthorized access. Then the SP redirects the user to the selected IdP sending a message containing Nonce, the address of the LS, and a temporary symmetric key TK (2). In addition, through an asynchronous XMLHttpRequest call, the SP communicates Nonce and URL_BACK to the LS (3)[3], which will store it locally in a temporary entry of its database. The data mapping <Nonce; URL_BACK> should be deleted or invalidated after the first use to avoid hijacking attacks. To successfully conclude the process, the user performs the authentication with the IdP. Once completed, she is redirected to the LS (4), which acts as a bridge and redirects the user to the URL provided previously by the SP (5). The IdP encrypts all user data with the temporary symmetric key TK.

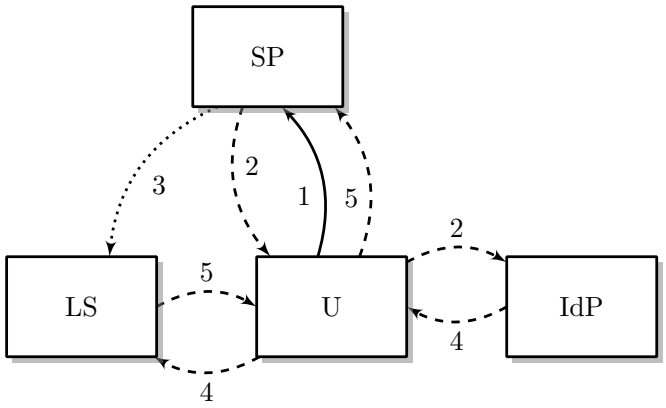

Fig. 4. Linking Service protocol.

It's important to stress that this solution splits the knowledge of the information in two parts and doesn't allow a single component of the architecture to get an overview of users' behavior.

The main contribution of our protocol is to solve an open problem in the field of federated IdM systems, ensuring that the IdPs are not aware of their users' actions and conforming to the principle of location independence. Moreover, this solution respects a stronger principle, since it allows to perform the authentication process without having to install extra components.

Our conceptual model is based on the belief that no element of the system should have a global view of the users' activities, intended as a set of static and dynamic components. The dynamic components are represented by the users' actions, such as the access to a SP or the authentication at an IdP. The static components are represented by attributes and authentication credentials. If knowledge is distributed among a sufficient number of elements, then none of them have the ability to compromise the whole of the users' privacy. The problem of ensuring privacy against usage profiling can be solved by moving part of the knowledge on a third service. The main idea behind our solution is to introduce a third element, the Linking Service, in order to ensure the users'

[3]Since, in the general case, these two entities would not be hosted on the same domain, an XMLHttpRequest would violate the browser's same origin policy. This problem can be solved using XMLHttpRequest Level 2 (http://www.w3.org/TR/XMLHttpRequest2/) which includes a feature to communicate with other domains according to the "Cross-Origin Resource Sharing" specification (http://dev.w3.org/2006/waf/access-control/).

privacy against usage profiling. The damage caused by the presence of a malicious IdP are contained as it holds less information. In particular, the IdP is not able to correlate users to used services.

It's important to note that none of the three elements of our architecture has a global vision of both static and dynamic components. Unlike what happens in the protocol described in [7], the IdP is not aware of the SP used and is able to encrypt authentication credentials and attributes using a symmetric encryption system.

The preservation of privacy against usage profiling and the validity of our model requires a minimal level of trust, no more than necessary in [8]. The user and the SP trust the Linking Service to safely keep the information about the mapping between the user and the SP. In addition, the user trusts the IdP for the management of confidential credential, but is assured that the IdP cannot trace her actions.

D. Implementation

Almost the same set of hardware and software described in section II-C has been used to develop this solution, the only notable difference being the inclusion of SQLite 3.6.22.

The prototype can be tested at the address: http://p2pg.di.unito.it/IdP_Privacy/ls/sp.php while its source code can be obtained from: http://p2pg.di.unito.it/IdP_Privacy/src/www.zip, in the folder `ls`.

IV. Conclusions

In this paper we presented two different solutions to a very important privacy problem in the context of Identity Management: the IdP knows every service a user chooses to employ and this can lead to very extreme consequences that range from profiling a user's behavior up to, in case of a breach in the IdP's infrastructure, the complete violation of all the services a user is associated to. To solve this issue our first proposed solution is to extend the browser allowing it to hide from the IdP which SP the user is trying to log-in to while remembering this association locally and acting as an intermediary between those two entities redirecting the user and transferring her data when necessary. Our second proposal introduced another service, the Linking Service (LS), which performs the same function the extension did but without requiring additional software on the user's computer.

Acknowledgment

We would like to thank Assosecurity[4] for financing part of this research. We would also like to thank the anonymous reviewers whose comments improved this article.

References

[1] R. Adhikari, "Can an Act of Congress Give the US the Cybersecurity It Needs?" http://www.technewsworld.com/story/69390.html?wlc=1266881734, February 2010.

[2] H. Al-Sinani and C. Mitchell, *Client-based CardSpace-OpenID Interoperation*, ser. Lecture Notes in Electrical Engineering. Springer-Verlag, 2011.

[3] H. S. Al-Sinani, W. A. Alrodhan, and C. J. Mitchell, "CardSpace-liberty integration for CardSpace users," in *IDtrust*, K. Klingenstein and C. M. Ellison, Eds. ACM, 2010, pp. 12–25.

[4] G. Alpár, J.-H. Hoepman, and J. Siljee, "The Identity Crisis. Security, Privacy and Usability Issues in Identity Management," *CoRR*, vol. abs/1101.0427, 2011.

[5] J. Camenisch and E. V. Herreweghen, "Design and implementation of the *idemix* anonymous credential system," in *ACM Conference on Computer and Communications Security*, V. Atluri, Ed. ACM, 2002, pp. 21–30.

[6] K. Cameron, "The Laws of Identity," http://www.identityblog.com/?p=352, May 2005.

[7] D. W. Chadwick, "Federated Identity Management," in *FOSAD*, ser. Lecture Notes in Computer Science, A. Aldini, G. Barthe, and R. Gorrieri, Eds., vol. 5705. Springer, 2008, pp. 96–120.

[8] D. W. Chadwick and G. Inman, "Attribute Aggregation in Federated Identity Management," *IEEE Computer*, vol. 42, no. 5, pp. 33–40, 2009.

[9] D. W. Chadwick, G. Inman, and N. Klingenstein, "A conceptual model for attribute aggregation," *Future Generation Comp. Syst.*, vol. 26, no. 7, pp. 1043–1052, 2010.

[10] A. Dey and S. Weis, "PseudoID: Enhancing Privacy for Federated Login," http://www.pseudoid.net/static/pseudoid.pdf.

[11] R. Dhamija and L. Dusseault, "The Seven Flaws of Identity Management: Usability and Security Challenges," *IEEE Security & Privacy*, vol. 6, no. 2, pp. 24–29, 2008.

[12] Eclipse Foundation, "Higgins: Open Source Identity Framework," http://www.eclipse.org/higgins/.

[13] ForgeRock, "OpenAM," http://www.forgerock.com/openam.html, April 2011.

[14] S. Fujiwara, T. Komura, and Y. Okabe, "A Privacy Oriented Extension of Attribute Exchange in Shibboleth," in *SAINT Workshops*. IEEE Computer Society, 2007, p. 28.

[15] S. Gajek, J. Schwenk, M. Steiner, and C. Xuan, "Risks of the CardSpace Protocol," in *ISC*, ser. Lecture Notes in Computer Science, P. Samarati, M. Yung, F. Martinelli, and C. A. Ardagna, Eds., vol. 5735. Springer, 2009, pp. 278–293.

[16] J. Granick, "Federal Authority Over the Internet? The Cybersecurity Act of 2009," https://www.eff.org/deeplinks/2009/04/cybersecurity-act, April 2009.

[17] M. Hansen, A. Schwartz, and A. Cooper, "Privacy and Identity Management," *IEEE Security & Privacy*, vol. 6, no. 2, pp. 38–45, 2008.

[18] A. Jøsang, M. A. Zomai, and S. Suriadi, "Usability and Privacy in Identity Management Architectures," in *ACSW Frontiers*, ser. CRPIT, L. Brankovic, P. D. Coddington, J. F. Roddick, C. Steketee, J. R. Warren, and A. L. Wendelborn, Eds., vol. 68. Australian Computer Society, 2007, pp. 143–152.

[19] S. Landau, H. L. V. Gông, and R. Wilton, "Achieving Privacy in a Federated Identity Management System," in *Financial Cryptography*, ser. Lecture Notes in Computer Science, R. Dingledine and P. Golle, Eds., vol. 5628. Springer, 2009, pp. 51–70.

[20] R. Marx, H. S. Fhom, D. Scheuermann, K. M. Bayarou, and A. Perez, "Increasing Security and Privacy in User-centric Identity Management: The IdM Card Approach," in *3PGCIC*, F. Xhafa, L. Barolli, H. Nishino, and M. Aleksy, Eds. IEEE Computer Society, 2010, pp. 459–464.

[21] R. L. B. Morgan, S. Cantor, S. Carmody, W. Hoehn, and K. Klingenstein, "Federated Security: The Shibboleth Approach," *EDUCAUSE Quarterly (EQ)*, vol. 27, no. 4, 2004.

[22] S. Motahari, S. G. Ziavras, R. P. Schuler, and Q. Jones, "Identity Inference as a Privacy Risk in Computer-Mediated Communication," in *HICSS*. IEEE Computer Society, 2009, pp. 1–10.

[23] Privacy Protection Study Cmssn United States, "Personal Privacy in an Information Society," http://www.ncjrs.gov/App/Publications/abstract.aspx?ID=49602, July 1977.

[24] D. Recordon and D. Reed, "OpenID 2.0: a platform for user-centric identity management," in *Digital Identity Management*, A. Juels, M. Winslett, and A. Goto, Eds. ACM, 2006, pp. 11–16.

[25] S. Suriadi, E. Foo, and A. Jøsang, "A user-centric federated single sign-on system," *J. Network and Computer Applications*, vol. 32, no. 2, pp. 388–401, 2009.

[26] L. Tien and S. Schoen, "Real ID Online? New Federal Online Identity Plan Raises Privacy and Free Speech Concerns," https://www.eff.org/deeplinks/2010/07/real-id-online-new-federal-online-identity-plan, July 2010.

[4] http://www.assosecurity.it/