Enhancing user authentication in claim-based identity management

Waleed Alrodhan and Chris Mitchell
Information Security Group
Royal Holloway, University of London

Agenda

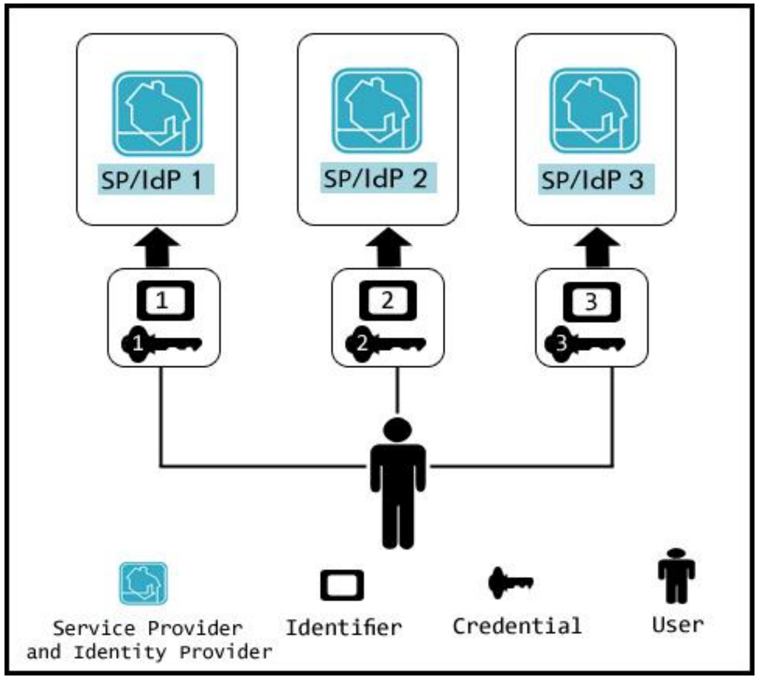
- Introduction
- Claim-based identity management
- Microsoft CardSpace
- The idea
- Enhancing user authentication
- Implementing a PoA method
- Discussion

Introduction

- Identity management is one of the fundamental building blocks for collaborative environments.
- Managing identities and protecting the corresponding credentials is a difficult problem.
- User-centric identity management provides users with more control over their identities.
- There is a risk that service providers could be deceived by untrustworthy or compromised identity providers.

Claim-based identity management I

- Many Internet ID management systems are designed to be cost effective from the perspective of service providers rather than users.
- Most such systems are *isolated*, i.e. there is no co-operation between systems for user authentication purposes.

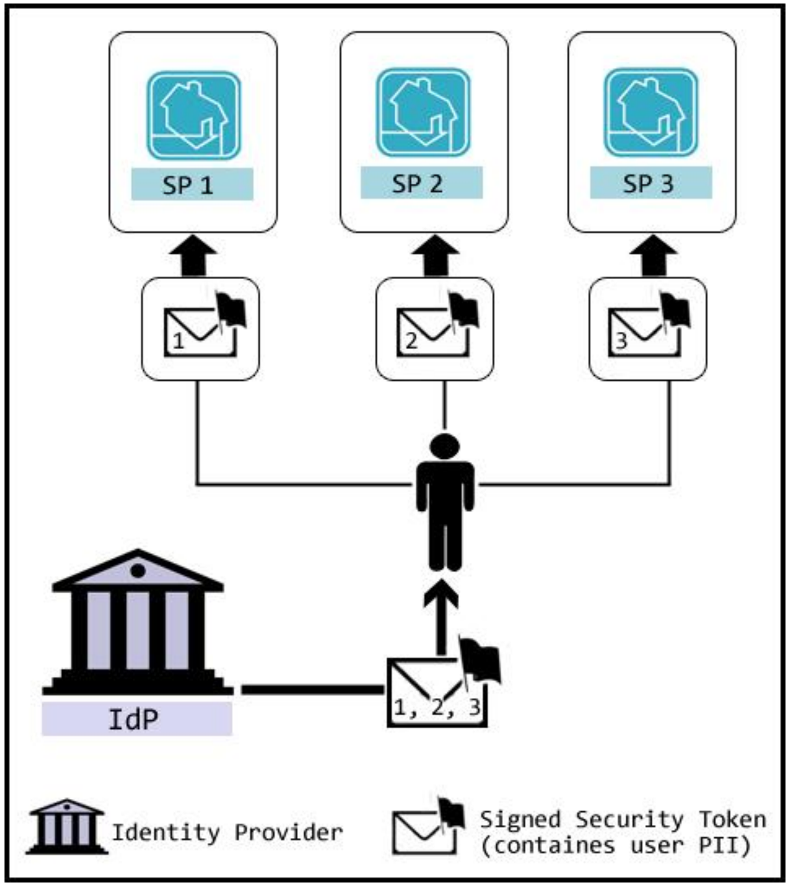


Claim-based identity management II

- Claim-based identity management (CBIM) enables users to use their Personally Identifiable Information (PII) to identify themselves to service providers, instead of using service provider specific identifiers (e.g. user names) and access credentials (e.g. passwords).
- In a CBIM system, each individual has an associated set of claims, where a claim is an assertion of the truth of some piece of PII for the associated user.
- In order to authenticate/authorise the user, the service provider can request a security token that asserts the truth of certain user PII. This security token must be signed by a trusted identity provider.

Claim-based identity management III

- A CBIM system is a type of user-centric ID management system.



CardSpace – background

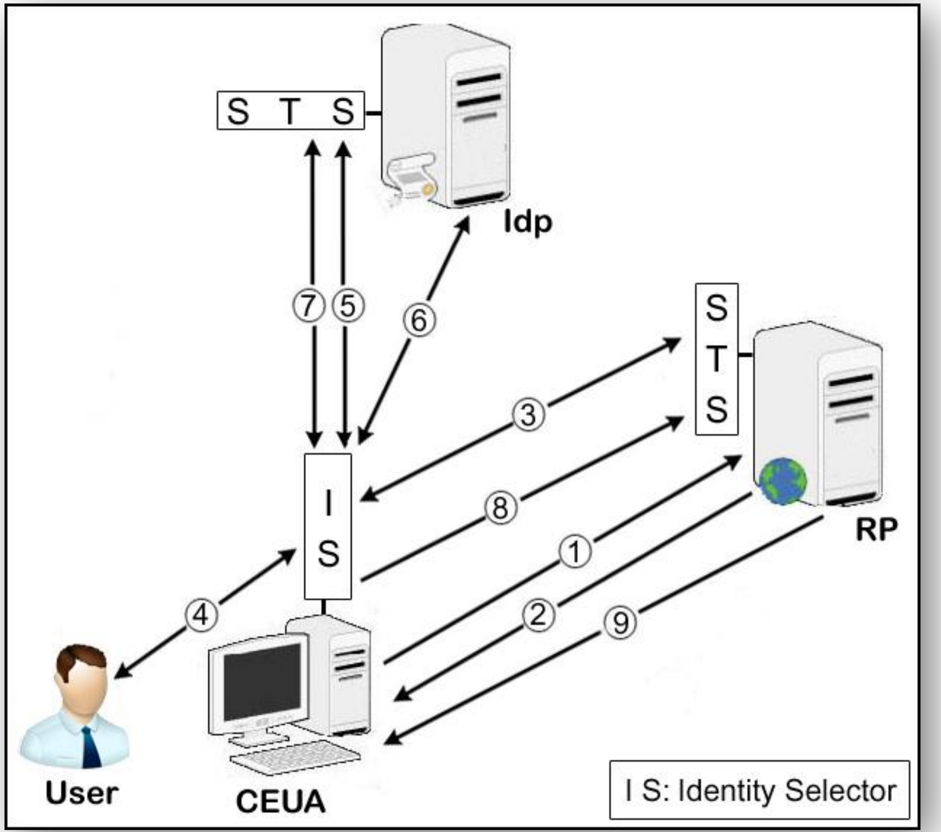
- Most widely discussed example of a CBIM system is Microsoft CardSpace.
- Currently deployed with Windows Vista and Windows 7 (works with multiple browsers).
- Based on identification process we experience in the real world when using physical identification cards.

CardSpace – system roles

- **User**;
- **CEUA** – CardSpace-enabled user agent (i.e. a web browser);
- **RP** – Replying Party, i.e. the service provider who wishes to authenticate the user;
- **IdP** – Identity Provider, i.e. the entity providing the security token;
- **STS** – Security Token Service, i.e. a service provided by the RP and/or the IdP for generating and consuming security tokens.

CardSpace – protocol

- The message flow



1. **CEUA** → **RP** : User clicks on the CardSpace logo on the RP log-in web page
2. **RP** → **CEUA** : InfoCard Tags (XHTML or HTML object tags), to trigger the Identity Selector
3. **Identity Selector** ↔ **RP-STS** : Identity Selector retrieves the RP security policy via WS-MetadataExchange
4. **Identity Selector** ↔ **User** : User picks an InfoCard
5. **Identity Selector** ↔ **IdP-STS** : Identity Selector retrieves the IdP security policy
6. **Identity Selector** ↔ **IdP** : User Authentication
7. **Identity Selector** ↔ **IdP-STS** : Identity Selector retrieves security token via WS-MetadataExchange
8. **Identity Selector** → **RP-STS** : Identity Selector forwards the security token (after, optionally, showing its contents to the user)
9. **RP** → **CEUA** : Welcome, you are now logged in!

CardSpace – the threat

- This system passes all the power/control to the IdP.
- User can choose the IdP, but what if the IdP is compromised or corrupt?
- Also, what if the user authentication method to the IdP is compromised?
- We have introduced a new single point of failure ...

The idea

- Add an extra (simple) direct user-RP authentication technique.
- **Objection!** Why use identity management if we go back to direct user-RP authentication?
- **Answers**:
  - IdP does more than authentication – notably it provides attribute management;
  - extra authentication method can be transparent to user;
  - Gives a type of two-factor user authentication.

Implementing the idea

- We propose two ways of adding this extra authentication method.
- Designed to minimise impact on user.
- First method (so called PoA method) has been prototyped.

Enhancing user authentication I

- Proof-of-Authenticity (PoA) method:
  - requires user platform to store a secret PoA value (known only to the client and the RP), that is sent to the RP during the authentication process. This proves to the RP that the genuine user is involved.
  - PoA value is randomly generated by the RP, and new value is sent to the user platform after every successful authentication (e.g. a HTTP cookie).
  - provision of PoA value should be transparent to the user, and hence will not affect the usability of the system.

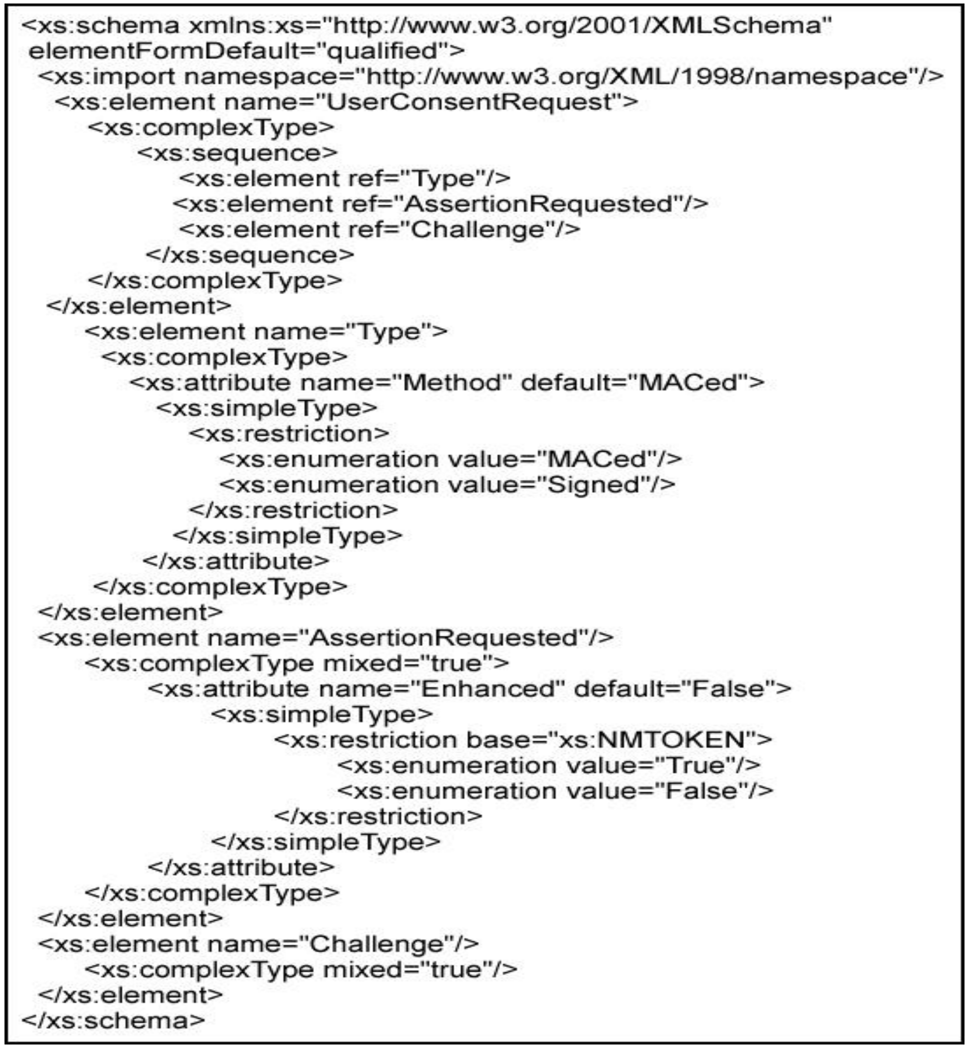
Enhancing user authentication II

- Challenge-response method:
  - requires modifying the XML-based security policy declaration message sent from the RP to the user platform.
  - requires user platform to either share a secret key with the RP or possess a signature key pair for which the RP has a trusted copy of the public key. The key is used as the basis of a challenge-response authentication of the user to the RP.

1. *MACed-Response Mechanism*.
   - Challenge: n
   - Response: $MAC_k(id_{RP} \| n)$
2. *Signed-Response Mechanism.*
   - Challenge: n
   - Response: $S_{\text{user}}(id_{RP} \| n)$

Enhancing user authentication III

- XML Schema
  for the new tags



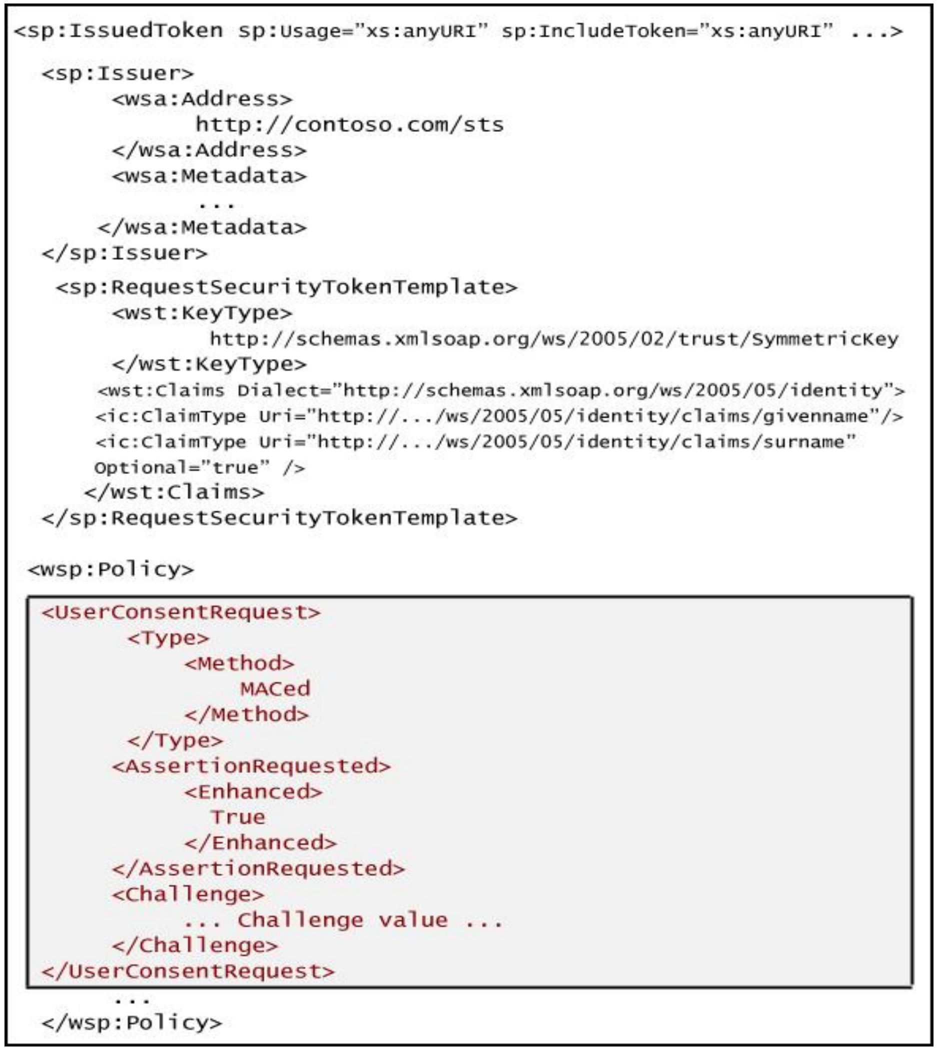
```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="qualified">
  <xs:import namespace="http://www.w3.org/XML/1998/namespace"/>
   <xs:element name="UserConsentRequest">
       <xs:complexType>
            <xs:sequence>
                 <xs:element ref="Type"/>
                <xs:element ref="AssertionRequested"/>
                 <xs:element ref="Challenge"/>
             </xs:sequence>
       </xs:complexType>
  </xs:element>
      <xs:element name="Type">
        <xs:complexType>
           <xs:attribute name="Method" default="MACed">
              <xs:simpleType>
                  <xs:restriction>
                     <xs:enumeration value="MACed"/>
                     <xs:enumeration value="Signed"/>
                  </xs:restriction>
                </xs:simpleType>
             </xs:attribute>
        </xs:complexType>
 </xs:element>
 <xs:element name="AssertionRequested"/>
      <xs:complexType mixed="true">
             <xs:attribute name="Enhanced" default="False">
                    <xs:simpleType>
                           <xs:restriction base="xs:NMTOKEN">
                                  <xs:enumeration value="True"/>
                                  <xs:enumeration value="False"/>
                           </xs:restriction>
                    </xs:simpleType>
             </xs:attribute>
      </xs:complexType>
 </xs:element>
 <xs:element name="Challenge"/>
      <xs:complexType mixed="true"/>
 </xs:element>
</xs:schema>
```

Royal Holloway
University of London

Information Security Group

Enhancing user authentication IV

- Modified RP Security Policy:



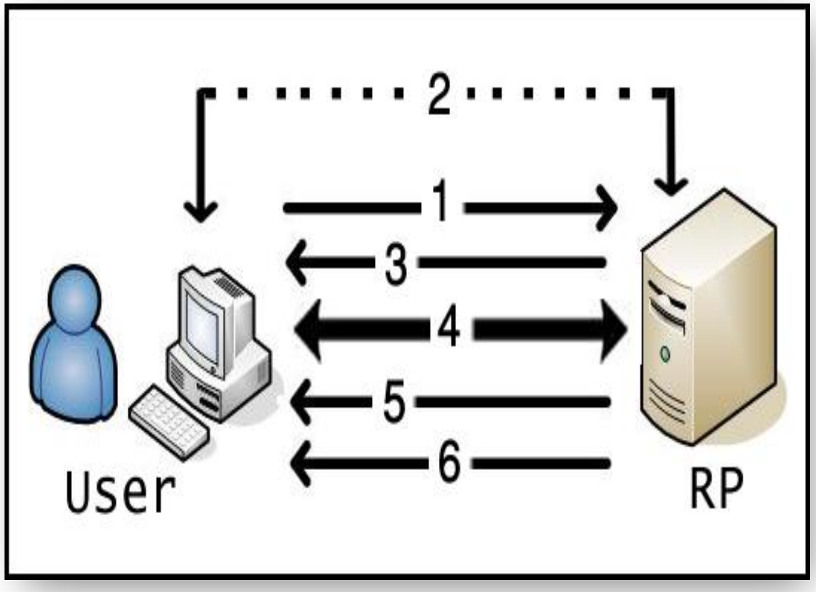
```
<sp:IssuedToken sp:Usage="xs:anyURI" sp:IncludeToken="xs:anyURI" ...>
  <sp:Issuer>
       <wsa:Address>
             http://contoso.com/sts
       </wsa:Address>
       <wsa:Metadata>
             ...
      </wsa:Metadata>
  </sp:Issuer>
   <sp:RequestSecurityTokenTemplate>
       <wst:KeyType>
              http://schemas.xmlsoap.org/ws/2005/02/trust/SymmetricKey
       </wst:KeyType>
      <wst:Claims Dialect="http://schemas.xmlsoap.org/ws/2005/05/identity">
      <ic:ClaimType Uri="http://.../ws/2005/05/identity/claims/givenname"/>
      <ic:ClaimType Uri="http://.../ws/2005/05/identity/claims/surname"
      Optional="true" />
     </wst:Claims>
  </sp:RequestSecurityTokenTemplate>

 <wsp:Policy>
  <UserConsentRequest>
        <Type>
            <Method>
                MACed
            </Method>
        </Type>
       <AssertionRequested>
            <Enhanced>
              True
            </Enhanced>
       </AssertionRequested>
       <Challenge>
            ... Challenge value ...
       </Challenge>
  </UserConsentRequest>
       ...
  </wsp:Policy>
```

Implementing a PoA Method I

- A proof of concept implementation of the PoA method has been successfully tested.
- The prototype was built on the Pamela Project's implementation of the RP CardSpace component.
- The implementation involved creating two software modules held on the RP server.
- The PoA is stored on the user machine in the form of an HTTP cookie.
- The two software modules that implement the proof-of-authenticity method are called PoASet and PoACheck. They are integrated with the CardSpace-enabling software on the RP server.

Implementing a PoA Method II

- The message flow.



1. **User** → **RP** : Login request using CardSpace.
2. **User** ↔ **RP** : RP checks whether or not the user has got the correct PoA.
3. **User** ← **RP** : Sorry you cannot use CardSpace this time!
4. **User** ↔ **RP** : Authentication of the user using another mechanism (e.g. username/password).
5. **User** ← **RP** : You have been authenticated, Welcome!
6. **User** ← **RP** : PoA to be presented next time.

Discussion I

- Proposed methods can increase the privacy level of CBIM systems, and help to make the RP's judgement regarding the validity of the security token less critical.
- Dishonest identity providers cannot impersonate users. This enhances system reliability from SP perspective, and benefits users by reducing the risk to their personal information held by IdPs.
- Proposed methods also reduce the significance of 'token-stealing' attacks.
- Challenge-response method builds on widely used WS-SecurityPolicy, so integrating method into current CBIM systems should be straightforward.

Discussion II

- Possible disadvantage of proposed methods is impact on user mobility.
- Limitation of PoA method – user must be authenticated once using another authentication system before using CBIM system. However, this may not be a major issue, especially if user is a frequent visitor to RP web site.
- Limitations of challenge-response method:
  - requires modifications to CBIM-enabling components on user machine and RP server;
  - key management overhead; however, if the shared key is compromised or stolen by an attacker, then it would not by itself give immediate access to the RP, since it only provides a second authentication method.

Thank You!

"On Facebook, 273 people know I'm a dog. The rest can only see my limited profile."