An Interoperable Approach to Multifactor Identity Verification

Federica Paci, Rodolfo Ferrini, Andrea Musci, Kevin Steuer Jr., and Elisa Bertino
CERIAS and CS Department Purdue University

Digital identity management (DIM) technologies are crucial to ensure the effective use of personally identifying information in Internet transactions. A digital identity contains data that uniquely describes an individual as well as information about the individual’s relationships to other individuals and entities. Privacy of these identity attributes is thus a crucial requirement.

Interoperability is another key requirement for successful DIM systems. However, it is a challenging, continuously evolving requirement, as industry and other organizations continue to introduce new capabilities and standards, such as SAML (http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf), Liberty Alliance (www.projectliberty.org/liberty/resource_center/specifications/liberty_alliance_id_wsf_1_1_specifications), OpenID (http://openid.net), CardSpace (http://msdn.microsoft.com/en-us/library/aa480189.aspx), Credentica (www.credentica.com), WS-Federation (http://specs.xmlsoap.org/ws/2006/12/federation), and Higgins (www.eclipse.org/higgins). The “Digital Identity Management Proposals” sidebar describes related work in this area.

Interoperability issues include the use of different identity tokens, negotiation protocols, and names for identity attributes. Initiatives such as the Concordia Project (http://projectconcordia.org/index.php/Main_Page) aim to harmonize existing identity specifications and protocols; however, such initiatives do not address the problem of naming heterogeneity. Naming heterogeneity occurs in DIM systems when the various parties involved in managing digital identities—clients, relying parties (RPs), and identity providers (IdPs)—use different vocabularies to denote identity attribute names.

We investigate the problem of naming heterogeneity in the context of the clients’ identity verification process. In this process, an RP specifies a set of identity attributes required from the user on behalf of whom the client is running and verifies that this user is the owner of these identity attributes. A user is the owner of identity attributes if entitled by a proper authority, such as an IdP, to
Digital Identity Management Proposals

The most relevant DIM proposals include the following:

- OpenID (http://openid.net),
- Credentica (www.credentica.com),
- Liberty Alliance (www.projectliberty.org/liberty/resource_center/specifications/liberty_alliance_id_wsf_1_1_specifications),
- Shibboleth (http://shibboleth.internet2.edu),
- WS-Federation (http://specs.xmlsoap.org/ws/2006/12/federation),
- SAML (http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf),
- Higgins (www.eclipse.org/higgins), and

We can classify these proposals into user-centric and federated DIM frameworks.

User-centric frameworks

CardSpace, Credentica, and OpenID are user-centric. The main difference among such proposals is the protocol they use to verify user identity. In CardSpace, the user selects from a set of information cards representing the digital identities that satisfy a relying party’s (RP’s) policy. The identity provider (IdP) that issued the card releases to the user a security token encoding claims corresponding to the selected information card. The user then passes the card and the token to the RP.

Credentica and CardSpace support similar identity verification protocols: The RP verifies the user’s identity based on an IdP-issued ID token, which encodes claims about the identity the user presents to the RP.

In OpenID, when users access an RP’s website, they provide an OpenID that is the URL of a webpage listing their IdPs. The RP selects an IdP, and the browser is redirected to the IdP’s webpage. If the IdP successfully verifies the user’s identity, the browser is redirected to the designated return page on the RP website, along with an assertion that the user is authenticated.

Our approach for digital identity verification aligns with user-centric DIM frameworks because the user can view the attributes returned by the RP as a result of the matching process through the client interface.

Federated frameworks

Other relevant proposals, such as Liberty Alliance, Shibboleth, and WS-Federation, are based on the notion of federated identity. Federations facilitate the use of user attributes across trust boundaries.

In Liberty Alliance, a federation consists of a circle of trust including service providers (SPs) and IdPs with mutual trust relationships. The circle of trust enables single sign-on (SSO) across different SPs’ websites. When an SP requests user authentication, the IdP authenticates the user and then issues an authentication assertion. The SP validates the assertion and determines whether to accept it. Once the SP authenticates the user, the user can sign on to other service sites without having to be reauthenticated at each site.

The Shibboleth initiative supports cross-domain SSO. WS-Federation does not propose another identity verification protocol but specifies how to use WS-Trust, WS-Security, and WS-Security-Policy to provide mechanisms for identity brokering, attribute discovery and retrieval, authentication, and authorization claims between federation partners protecting the privacy of these claims across organizations.

Our approach assumes a “relaxed” notion of federation in that a trust relationship does not need to exist between all the SPs and IdPs.

Interoperability

Due to the number of proposed DIM systems, interoperability is a major issue. A first step toward achieving interoperability is the adoption of SAML. SAML does not provide an identity verification protocol but supports a standard syntax for the representation of assertions about identity attributes and IdP authentications. SAML is important in our approach as it facilitates the exchange of identity tuples and mapping certificates across parties in a federation.

The Concordia and Higgins projects address interoperability across existing identity verification protocols. Concordia’s goal is to help drive the development of use case scenarios where multiple identity specifications, standards, or other initiatives might coexist.

Higgins is an open source framework that enables the integration of identity, profile, and relationship information across multiple heterogeneous systems. Higgins’ key component is the Identity Attribute Service, a pluggable framework for the integration and abstraction of identity and relationship data across multiple data sources. IdAS uses the Higgins Context Data Model, which provides an abstraction to represent identity-related data.

The main difference between our work and these initiatives is its focus on naming heterogeneity rather than on heterogeneity among identity verification protocols. As such, our approach is complementary to these initiatives.

Use these attributes for obtaining services and resources. An identity verification policy specifies such a set of identity attributes. If the clients and the relying parties use names for the identity attributes from different vocabularies, when a client requests a resource or a service from an RP, the client may not understand which identity attributes it must provide.

We propose a multifactor identity verification protocol that supports identity attribute name matching. In multifactor verification, when a client presents an identity attribute requested by an RP, the RP requires additional identity attributes as proof of client ownership of the identity attribute. The protocol uses an identity attribute names matching technique based on lookup tables, dictionaries, and ontology mapping to match RPs’ and clients’ vocabularies. The technique also uses the aggregated zero knowledge proofs of knowledge (AgZKPK) cryptographic protocol to allow clients to prove, with a single interactive proof, their ownership of multiple identity attributes. Since clients do not need to provide the attributes in clear text, this protocol ensures their privacy.
NAMING HETEROGENEITY

To address naming heterogeneity, the first issue to investigate is which matching technique to use, which in turn depends on the variations in identity attribute names. The variations can be classified as syntactic and terminological:

- Syntactic variations refer to the use of different character combinations to denote the same term. An example is the use of “CreditCard” and “Credit_Card” to denote a credit card. We identify syntactic variations using lookup tables that enumerate the possible ways the same term can be written by using different character combinations.
- Terminological variations refer to the use of different terms to denote the same concept. An example of a terminological variation is the use of the synonyms “credit card” and “charge card.” We identify terminological variations by means of dictionaries or a thesaurus such as WordNet (http://wordnet.princeton.edu), which retrieves all the synonyms of a given term.

A comprehensive identity attribute matching approach should also consider the attribute’s domain and ontology. A domain is a set of application-related concepts that are relevant with respect to a certain area of interest and is usually formalized by an ontology. An ontology represents a domain in terms of concepts and properties with which those concepts are related. If two attributes are associated with concepts belonging to the same ontology, we can easily compare their meanings and determine the relationship between them.

However, we cannot always assume that clients and RPs associate their attributes with the same ontology. Therefore, we identified a third category—semantic variations. Semantic variations denote the same term using two concepts from different domains characterized by different ontologies.

Another important issue in naming heterogeneity is determining which party should perform the matching. In our context, either party can execute the matching process. However, performing the matching at the client has an obvious drawback: To get access to services or resources for which it does not have proper identity credentials, the client might lie and assert that an identity attribute referred to in the RP policy matches one of its attributes. Therefore, the RP should perform the match. Notice that if a privacy-preserving protocol is used to perform identity verification, as in our approach, the RP will not learn the values of the client’s identity attributes and therefore does not have an incentive to lie.

PRELIMINARY CONCEPTS

Our approach assumes a DIM system consisting of the following entities: RPs, IdPs, registrars (Rs), and clients. RPs provide services to clients as in conventional e-commerce and other federated environments. IdPs issue certified identity attributes to users and control the sharing of such information. Rs are additional components, introduced by our approach, to store and manage information related to strong identity attributes that are used in our multifactor identity attribute verification approach. Strong identity attributes uniquely identify an individual in a population, whereas weak identifiers correspond to many individuals.

Unlike IdPs, Rs do not store the values of the identity attributes in clear text; Rs only contain the cryptographic, semantically secure commitments of the strong identity attributes, which the clients then use to construct zero knowledge proofs of knowledge of those attributes. A ZKPK is an interactive method for proving to another party that a (usually mathematical) statement is true, without revealing anything other than the veracity of the statement. A commitment scheme or a bit commitment scheme allows a user to commit to a value while keeping it hidden and preserving the user’s ability to reveal the committed value later.

For each user, the R stores an identity record (IdR) containing an identity tuple for each identity attribute m. Each identity tuple, which essentially is an identity token, consists of the following:

- a tag;
- an attribute descriptor;
- the Pedersen commitment of m, denoted as \( M_i \);
- the signature of the R on \( M_i \), denoted by \( \sigma_i \);
- two types of assurance indicators, namely validity assurance and ownership assurance; and
- a set of weak identifiers, \( \{ W_{mj} \} \), which can be aggregated to perform multifactor verification.

\( M_i \) is computed as \( g^m h^r \pmod{p} \), where \( m \) is the value of the identity attribute, \( r \) is a random number in \( Z_p \), and
is known only by the client, and \(g\) and \(h\) are generators of a group \(G\) of prime order \(p\). \(G, g, h,\) and \(p\) are public parameters of the R. **Validity assurance** corresponds to the confidence about the validity of the identity attribute based on the verification performed at the identity attribute's original issuer. **Ownership assurance** corresponds to the confidence about the claim that the principal presenting an identity attribute is its true owner. Weak identifiers are used to denote identity attributes that can be aggregated to perform multifactor verification. The RPs can retrieve from R the identity tuples of each registered client (online mode), or the R can release to the client a certificate containing its identity record (offline mode).

RPs can specify two different types of policies to verify clients' identity. The first type requires a client to provide a set of identity attributes in clear text; the RP typically applies this policy when the identity attribute encodes information required for the transaction to execute between the client and the RP. The second type requires a client to prove the possession of a set of identity attributes without providing the actual values. The RP typically applies this policy when the identity attribute's content is not relevant for the transaction execution itself. Therefore, to satisfy the latter type of policy, the client only has to prove it knows how to open the commitments of the identity attributes specified in the policies.

To enable the matching of identity attributes by using lookup tables, dictionaries, and ontology mapping, we assume that RPs define identity verification policies according to their domain ontology. Therefore, the identity attribute names listed in the identity verification policies correspond to a concept in the ontology. Moreover, each RP maintains a lookup table containing alternative character combinations and a table recording sets of synonyms for some of the identity attribute names in its identity verification policies.

We also assume that the RPs and R have a common public-key infrastructure, which allows each party to issue certificates to users and verify identity tokens and certificates that other parties issue. Thus, R can issue users a certificate containing their identity tuples, and RPs can issue **mapping certificates** following a successful identity verification process. A mapping certificate asserts that a client's identity attribute matches a concept in the RP's ontology and that this RP has verified that the client owns the attribute and received it from an authorized IdP.

For example, consider a client “John White,” who is characterized by a set of identity attributes \{DriverLicense, SocialSecurityNumber\}. Suppose John would like to buy a book from the BookStoreOnline RP, which defines its identity verification policies according to an ontology \(O_{BookStoreOnline}\). To prove his identity, John must provide BookStoreOnline the identity attributes SocialSecurityNumber and National Identification Number, and between DriverLicense and Driver Permit. Because of naming heterogeneity, the DIM system running on behalf of John does not understand which attributes to provide. Thus, BookStoreOnline and John's system carry out the matching protocol, as Figure 1 shows. As a result of the matching, BookStoreOnline determines the associations between the attributes SocialSecurityNumber and National Identification Number, and between DriverLicense and Driver Permit. Upon verification of John's identity attributes, BookStoreOnline issues John the mapping certificate.
Mapping certificates both determine the matching between clients’ and RPs’ identity attributes and populate the list of semantic neighbors of RPs—other RPs that have successfully mapped their ontology with the client’s ontology. The list updates whenever a client presents a mapping certificate during the identity attribute matching phase of our protocol, and the RP that issued the certificate is not on the list. Thus, the RP tries to match its ontology with one of the RPs that issued the certificate. If successful, the RP that issued the certificate joins the list of semantic neighbors.

INTEROPERABLE MULTIFACTOR IDENTITY VERIFICATION

Our privacy-preserving multifactor identity verification protocol, which supports interoperable interactions between clients and RPs, consists of two processes. In the first process, the RP matches the identity attributes in the client’s vocabulary with its own attributes to help the client understand its identity verification policy. In the second phase, the client carries out an AgZKPK protocol to prove to the RP the knowledge of the matched identity attributes.

Identity attribute names matching

The matching process first identifies syntactical and terminological variations in identity attribute names. During this phase, the RP, referred to as source RP, sends a set of alternative character combinations and a set of synonyms for the identity attributes listed in its policy.

The RP thus presents the client with predefined naming alternatives. If by using the RP’s information, the client cannot match its identity attributes with the source RP’s, the client sends the RP the mapping certificates released by other RPs. In the second phase of the matching process, the source RP tries to match the concepts corresponding to the identity attributes the client cannot provide with concepts from the ontologies of the RPs that have issued the mapping certificates.

The RP can use either direct or indirect ontology mappings to determine matches:

- Direct ontology mappings are found by mapping the ontology of the source RP with the ontologies of the RPs that have issued a mapping certificate.
- Indirect ontology mappings use the client’s mapping certificate as well as mappings of the source RP’s semantic neighbors to build a mapping path—a sequence through which a concept $c$, which the source RP must match with a client identity attribute name, is indirectly mapped on a concept $c_t$ of the ontology of an RP that has issued a mapping certificate.

The source RP only selects matches that have a confidence score $s$ greater than a predefined threshold $\tau$. The source RP sets the acceptance threshold $\tau$ to assess the matches’ validity. The greater the threshold, the greater the similarity between the two concepts and the higher the probability that the match is correct. If the source RP finds direct or indirect mappings for its concepts, it then uses the information in the mapping certificates to verify that each matching concept matches a client’s attribute. If this check fails, the RP notifies the client, and the interaction terminates.

For example, consider an RP “Source” that has two semantic neighbors, RP2 and RP4, and a client “Target” that has provided a mapping certificate issued by the RP. Source has to find a match between the concept SocialSecurity# in its ontology and a client attribute. To determine such a match, Source tries to map SocialSecurity# with a
concept in Target’s ontology by exploiting the mapping with RP2 and RP4.

In Figure 2, a pair of concepts, along with a confidence number indicating their similarity, represents a mapping. In this example, Source finds a mapping path—(SocialSecurity#, Social_Security_Number, 0.910), (Social_Security_Number, National_Security_Code, 0.753), (National_Security_Code, Fiscal_Code, 0.838)—through which SocialSecurity# maps to Fiscal_Code in Target’s ontology. A match occurs if Fiscal_Code corresponds to a client identity attribute.

**Multifactor verification**

The multifactor identity verification process consists of several steps, as Figure 3 shows. Once the client receives the set of matched identity attributes \(m_1, \ldots, m_t\) from the RP, it computes the aggregated proof

\[
M = \prod_{i=1}^{t} M_i, \quad \text{and the aggregated signature } \sigma = \prod_{i=1}^{t} \sigma_i,
\]

where \(M_i = g^{m_i} h^r\) represents the commitments of the matched identity attributes and \(\sigma_i (1 \leq i \leq t)\) represents their corresponding signatures (C1). According to the AgZKPK protocol, the client randomly picks \(y, s \in [1, \ldots, p]\) (C2), computes \(d = g^h \mod p\) (C3), and sends \(M, \sigma, d, \{M_1, \ldots, M_t\}, \{\sigma_1, \ldots, \sigma_t\}\) to the RP. The RP accepts the zero knowledge proof if \(g^h = dM^e\) (RP2). If the RP accepts the zero knowledge proof of the commitments, it checks the validity of the aggregated signature \(\sigma\) (RP3); if the signature verification succeeds, the RP grants the service and releases a mapping certificate to the client (RP4).

**IMPLEMENTATION AND EVALUATION**

We performed several experiments to evaluate the identity attribute names matching process and the AgZKPK process. We implemented the R and RP components as Java servlets and adopted JSP for client component development. To execute the tests, we deployed 32 different RP components and one client component. We used eight machines on which we deployed four RP components, each with the following system configuration: Windows XP Professional SP3, Intel Dual Core Processor 2.33 GHz, and a 2-Gbyte 667-MHz DDR2 RAM. We used the Purdue campus wireless network (802.11g, 54 Mbps) for communication among the computers.

We also generated a set of ontologies with an average cardinality of 60 concepts. Then, to create the set of semantic neighbors for each RP component, we first computed the mappings between a subset of these ontologies.
creation (blue line) increases. We expected an increase in mapping path computation because the same basic operations repeat for a number of times equal to the number of attributes. Thus, the computation of a mapping path is more efficient than ontology mapping with a smaller number of attributes.

Figure 4b shows the time required for computing the mapping paths when varying the number of RPs. The computing time increases because, to build the mapping path, each RP must find a mapping between its concept and a concept of its semantic neighbors. This step repeats a number of times equal to the number of RPs.

Figure 4c shows the times to generate and verify an AgZKPK, with varying values of aggregated attribute numbers. This execution time (blue line) is almost constant for increasing values in the number of identity attributes because the creation of AgZKPK requires a constant number of exponentiations. By contrast, the RP component’s verification time increases linearly with the number of identity attributes to verify because the RP component must multiply all the commitments to verify the resulting aggregate signature.

Our approach to naming heterogeneity in digital identity management uses a combination of techniques from the Semantic Web and security protocols. We plan to extend this work in several directions. We will limit RPs’ knowledge about the users’ previous interactions with other RPs based on the information contained in mapping certificates. We will also investigate the use of nyms instead of weak identifiers in identity tuples for the purpose of attribute matching. To allow service providers to specify conditions on identity attributes, we will attempt to define a language for identity verification policies. We also plan to extend the AgZKPK protocol to verify that identity attributes’ commitments satisfy such conditions.

Acknowledgments

This article is based in part on work supported by the US Department of Homeland Security under grant award number 2006-CS-001-000001, under the auspices of the Institute for Information Infrastructure Protection (I3P) research program.

References


Federica Paci is a postdoctoral candidate in the Department of Computer Science at Purdue University. Her research interests include access control for service-oriented architectures, digital identity management, and privacy in social
networks. She received a PhD in computer science from the University of Milan. Contact her at paci@cs.purdue.edu.

Rodolfo Ferrini is a research assistant in the Department of Computer Science at Purdue University. His research interests include access control models, privacy preservation, and Semantic Web technologies. He received a PhD in computer science from the University of Bologna. Contact him at rferrini@cs.purdue.edu.

Andrea Musci is a research assistant in the Department of Computer Science at Purdue University. His research interests include digital identity management and access control for business processes. He received a bachelor’s in computer science from the University of Rome 3. Contact him at amusci@cs.purdue.edu.

Kevin Steuer Jr. is a graduate student in the Department of Computer Science at Purdue University. His research interests include digital identity management and information assurance. He plans to graduate with a master’s of computer science from Purdue University in 2010. Contact him at ksteuer@cs.purdue.edu.

Elisa Bertino is a full professor in the Department of Computer Science at Purdue University and research director of the Center for Education and Research in Information Assistance and Security. Her research interests include privacy techniques for databases, digital identity management, policy systems, and security for Web services. She received a PhD in computer science from the University of Pisa. Contact her at bertino@cs.purdue.edu.