A Federated Digital Identity Management Approach for Business Processes

Elisa Bertino, Rodolfo Ferrini, Andrea Musci, Federica Paci, and Kevin J. Steuer

CS Department and CERIAS, Purdue University, West Lafayette IN 47907, USA
{bertino,rferrini,amusci,paci,ksteuer}@cs.purdue.edu

Abstract. Business processes have gained a lot of attention because of the pressing need for integrating existing resources and services to better fulfill customer needs. A key feature of business processes is that they are built from composable services, referred to as *component services*, that may belong to different domains. In such a context, flexible multi-domain identity management solutions are crucial for increased security and user-convenience. In particular, it is important that during the execution of a business process the component services be able to verify the identity of the client to check that it has the required permissions for accessing the services. To address the problem of multi-domain identity management, we propose a multi-factor identity attribute verification protocol for business processes that assures clients privacy and handles naming heterogeneity.

Keywords: identity management, business process, naming heterogeneity, interoperability.

1 Introduction

Business processes have gained a lot of attention because of the pressing need for integrating existing resources and services to better fulfill customer needs. A key feature of business processes is that they are built from composable services, referred to as *component services*, that may belong to different domains. In such a context, flexible multi-domain identity management solutions are crucial for increased security and user-convenience. In particular, it is important that during the execution of a business process the component services be able to verify the identity of the client to check that it has the required permissions for accessing the services. Clients identity consists of data, referred to as *identity attributes*, that encode relevant-security properties of the clients. The management of identity attributes in business processes raises however a number of challenges. On one hand, to enable authentication, the propagation of client's identity attributes across the component services should be facilitated. On the other hand, identity attributes need to be protected as they may convey sensitive information about a client and can be target of attacks. Moreover, because business processes orchestrate the functions of services belonging to different domains, interoperability issues may arise in client authentication processes. Such

E. Bertino and J.B.D. Joshi (Eds.): CollaborateCom 2008, LNICST 10, pp. 194–206, 2009.

issues range from the use of different identity tokens and different identity negotiation protocols, such as the client-centric protocols and the identity-providers centric protocols, to the use of different names for identity attributes. The use of different names for identity attributes, that we refer to as *naming heterogeneity*, typically occurs because clients and component services use a different vocabulary to denote identity attribute names. In this case, whenever a component service requests from a client a set of identity attributes to verify its identity, the client may not understand which identity attributes it has to provide.

To address the problem of multi-domain identity management, we propose a multi-factor identity attribute verification protocol for business processes that assures clients privacy and handles naming heterogeneity. The protocol uses an identity attribute names matching technique based on look-up tables, dictionaries and ontology mapping, to match component services and clients vocabularies and aggregate zero knowledge proofs of knowledge (AgZKPK) cryptographic protocol to allow clients to prove with a single interactive proof the knowledge of multiple identity attributes without the need to provide them in clear.

The rest of the paper is organized as follows. Section 2 introduces a running example that is used throughout the paper to illustrate the discussion. Section 3 discusses the main issues related to digital identity management for business processes. Section 4 introduces the notions on which our multi-factor identity attribute verification protocol is based. Section 5 presents the multi-factor identity attribute verification protocol. Section 6 discusses the system architecture. Section 7 reports experimental results. Finally, Section 8 concludes the paper and outlines some future work.

2 Running Example

In this section we introduce an example of business process that implements a loan approval process (see Figure 1). Customers of the service send loan requests. Once

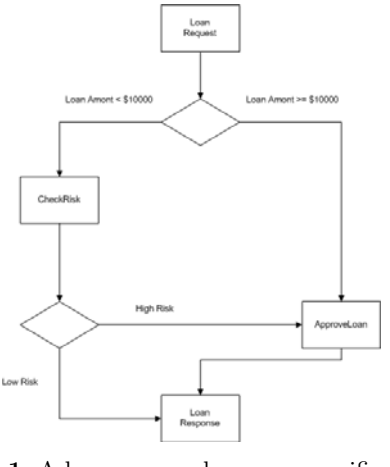


Fig. 1. A loan approval process specification

a request is received, the loan service executes a simple process resulting in either a "loan approved" message or a "loan rejected" message. The decision is based on the amount requested and the risk associated with the customer. For amounts lower than 10,000$ a streamlined process is used. In the streamlined process low-risk customers are automatically approved. For higher amounts, or medium and high-risk customers, the credit request requires further processing. For each request, the loan service uses the functions provided by two other services. In the streamlined process, used for low amount loans, a *risk assessment* service is used to obtain a quick evaluation of the risk associated with the customer. A full *loan approval* service (possibly requiring direct involvement of a loan expert) is used to obtain an assessment about the customer when the streamlined approval process is not applicable. Four main activities are involved in the process:

- `Loan Request` allows a client to submit a loan request to the bank
- `Check Risk` (provided by *risk assessment* service) computes the risk associated with the loan request
- `Approve Loan` (provided by *loan approval* service) determines if the loan request can be approved or rejected
- `Loan Response` sends to the client the result of the loan request evaluation process

risk assessment and *loan approval* services require a set of identity attributes from the client who has submitted the loan request. The risk assessment service asks $DrivingLicense$, $CarRegistration$ and $EmployeeID$, whereas the loan approval service requires $EmployeeID$ and $CreditCard$.

3 Identity Management for Business Processes

Managing and verifying clients identity in a business processes raise a number of challenging issues. A first issue is related to how the client's identity attribute have to be managed within the business process. The client of a business process is not aware that the business process that implements the required service invokes some component services. The client thus trusts the composite service but not the component services. Therefore, every time the component services have to verify the client's identity, the composite service has to act as an intermediary between the component services and the client. Moreover, since the client's identity attributes may contain sensitive information and clients usually do not trust the component services, the client's identity attributes should be protected from potential misuse by component services.

Another issue is related to how the identity verification process is performed. Because component services belong to different domains, each with its own identity verification policies, the sets of identity attributes required to verify client's identity may partially or totally overlap. Therefore, the client has to prove several times the knowledge of the same subset of identity attributes. It is thus important to take advantage of previous client identity verification processes that other component services have performed.

Finally, another issue is the lack of interoperability because of naming heterogeneity. Naming heterogeneity occurs when component services define their identity verification policies according to a vocabulary different from the one adopted by clients. Therefore, component services and clients are not able to have "meaningful" interactions because they do not understand each other. Thus, it is also necessary that client identity verification process supports an approach to match identity attribute names of component services and clients vocabularies. In such respect, a first question to be addressed is which matching technique to use, which in turn depends from the types of variation in identity attribute names. A second question is related to the matching protocol to use, that is, by which party the matching has to be performed and whether the fact that a client has already performed a matching with a component service may help in a subsequent matching.

To address such issues we propose a multi-factor identity attribute verification protocol for business processes that supports a privacy usage of clients identity attributes and that guarantees interoperable interactions between clients and component services. In what follows, we provide more details about our approach.

4 Preliminary Concepts

To enable multi-factor identity attribute verification, clients have to register their identity attributes to a *registrar* [1]. The registrar is an additional component in digital identity management systems that stores and manage information related to identity attributes. For each client's identity attribute m, the registrar records an identity tuple $(\sigma_i, M_i, tag, validity - assurance, ownership - assurance, \{W_{ij}\})$. Each identity tuple consists of tag, an attribute descriptor, the Pedersen commitment [5] of m, denoted as M_i, the signature of the registrar on M, denoted as σ_i, two types of assurance, namely *validity assurance* and *ownership assurance* and a set of weak identifiers $\{W_{ij}\}$. M_i is computed as $g^m h^r$, where g and h are generators in a group G of prime order q. G and q are public parameters of the registrar and r is chosen randomly from $\mathbb{Z}_q$. 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 together to perform multi-factor authentication. The identity tuples of each registered client can be retrieved from the registrar by the component services or the registrar can release to the client a certificate containing its identity record.

We assume that each of the component services define their identity verification policies by specifying a set of identity attribute names that have to be required from the client.

Because of naming heterogeneity, clients may not understand component services identity verification policies. The type of variations that can occur in clients and component services identity attribute names can be classified in: *syntactic*, *terminological* and *semantic* variations.

- *Syntactic variations* arise because of the use of different character combinations to denote the same term. An example is the use of "CreditCard" and "Credit_Card" to denote a client's credit card.
- *Terminological variations* refer to the use of different terms to denote the same concept. An example of terminological variation is the use of the synonyms "Credit Card" and "Charge Card" to refer a client's credit card.
- *Semantic variations* are related to the use of two different concepts in different knowledge domains to denote the same term.

Syntactic variations can be identified by using look up tables. A look up table enumerates the possible ways in which the same term can be written by using different character combinations. In detecting terminological variations, dictionaries or thesaurus such as WordNet[6] can be exploited. Finally, semantic variations can be determined by using ontology matching techniques. An ontology is a formal representation of a domain in terms of concepts and properties with which those concepts are related. Ontologies can be exploited to define a domain of interest and for reasoning about its features. Ontology mapping is the process whereby two ontologies are semantically related at conceptual level; source ontology concepts are mapped onto the target ontology concepts according to those semantic relations [4]. Typically an ontology matching algorithm takes in input two ontologies O_i and O_j, and returns a set of triples of the form $\langle c_i, c_j, s\rangle$, where c_i is a concept belonging to ontology O_i, c_j is a concept belonging to ontology O_j that matches concept c_i, and s is a *confidence score*, that is, a value between 0 and 1, indicating the similarity between the matched concepts.

To enable the matching of identity attributes by using the above techniques, we make the following assumptions. Component services' identity verification policies are defined according to their domain vocabulary ontology. Moreover, they track existing mappings with other component services' ontologies. Such mappings are formally represented by tuples of the following form:

$$\langle O_{CS}, CS', O_{CS'}, \{\langle c_1, c_2, s_{1,2}\rangle, \ \ldots \ , \langle c_l, c_m, s_{l,m}\rangle\}\rangle$$

where O_{CS} is the ontology of a component service CS, CS' is a component service whose ontology $O_{CS'}$ matches ontology O_{CS} and $\{\langle c_1, c_2, s_{1,2}\rangle, \ldots \ , \langle c_l, c_m, s_{l,m}\rangle\}$ is the set of concepts mappings $\langle c_i, c_j, s_{i,j}\rangle$ where $c_i \in O_{CS}$ and $c_j \in O_{CS'}$. Moreover, each component service keeps a look up table containing alternative character combinations and store a set of synonyms, denoted as *Synset*, for each of the identity attribute names used for expressing its identity verification policies. Finally, since we want to avoid that the client proves several times the possession of a same set of identity attributes, we assume that component services have a PKI infrastructure that allows them to issue certificates to clients. These certificates (see Definition 1 below) assert that an identity attribute by a client matches an identity attribute by a component service and that the component service has verified that the client owns the attribute. Clients can use these certificates to prove that they own a set of identity attributes without going through the authentication process during the execution of the same business process instance in which the certificates have been released. Instead, clients can use the

certificates in business processes different from the one in which the certificate have been issued to prove there is a mapping between a set of client's attributes and a service's ontology. This distinction is motivated by the fact that there is a trust relationship between the component services in the same business process instance, that may not exist with services external to the process.

Definition 1 (Proof-of-Identity Certificate). *Let S be a component service participating to a business process BP and C be a client. Let O_S be the ontology describing the domain of S and $AttrSet$ be the set of C's identity attribute names. The proof of identity certificate released by S to C upon a successful verification is a tuple $\langle Issuer, Owner, OID, Mappings, IssuanceDate\rangle$ where: $Issuer$ is the identifier of S, $Owner$ is the identifier of C, OID is O_S ontology identifier, $Mappings$ is a set of tuples of the form $\langle Attr, Concept\rangle$ where $Attr \in AttrSet$ and $Concept \in O_S$, and $IssuanceDate$ is the release date of the certificate.*

Besides being stored by the clients, proof-of-identity certificates released during the execution of a business process instance are stored in a local repository, denoted as $CertRep$, by the composite service for the whole process execution.

5 Interoperable Multi-factor Authentication

In this section, we present a multi-factor authentication protocol for business processes. The protocol takes place between a client, the composite service and a component service. Since the client is not aware of the component services, the composite service has to mediate the interactions between them. The protocol consists of two phases that make use of the notion of proof-of-identity certificate introduced in the previous section (see Figure 2). In the first phase, the component service matches the identity attributes of clients vocabulary with its own attributes to help the client understand its identity verification policy. In the second phase, the client carries out an aggregate ZKPK protocol to prove to the component service the knowledge of the matched identity attributes. Algorithm 1 summarizes the different phases of the protocol.

5.1 Identity Attribute Matching Protocol

The technique that we have developed for matching identity attribute names from different vocabularies is based on the combined use of look-up tables, dictionaries, and ontology mapping.

As we have already mentioned, an important issue is which party has to execute the matching. In our context, the matching can be executed by the client, the composite service or the component services. Performing the matching at the client has the obvious drawback that the client may lie and asserts that an identity attribute referred to in the component services policy matches one of its attribute, whereas this is not the case. The matching process cannot be performed by the composite service because it should have access to information which are local to the component services. Therefore, in

our approach, the matching is performed by the component services. Notice that because of the privacy-preserving protocol that we use (see next section), the composite service and the component services will not learn the values of the identity attributes of the client and therefore do not have incentives to lie.

Algorithm 1: `Multi-factor verification protocol`

Input: $CertRep$: proof-of-identity certificates repository
$AttrProof$: set of identity attributes requested from the client
Output: c_i: proof-of-identity certificate

(1) **foreach** $a_i \in AttrProof$
(2) **if** $\exists c_j \in CertRep$ such that c_j prove the knowledge of a_i
(3) a_i is verified
(4) **else**
(5) `Match` a_i with client's proof-of-identity certificates
(6) `Verify` AgZKPK
(7) `Release` new proof-of-identity certificate c_i
(8) `Store` c_i in $CertRep$

A second issue is how to take advantage of previous interactions that the client has performed with other component services. It is also important to exploit mappings that can exist between ontologies by different component services. To address such issue, the matching protocol relies on the use of the proof-of-identity certificates and matching techniques. We assume that $AttrProof$ is the set of identity attributes that a component service asks to a client to verify its identity. The identity attribute name matching process is carried out between the client, the component service and the composite service when some attributes in $AttrProof$ do not match any of the attributes in $AttrSet$, the set of clients' identity attributes. We refer to the set of component service's identity attributes that do not match a client attribute name to as $NoMatchingAttr$. The matching process consists of two main phases. The goal of the first phase is to match the identity attributes that have syntactical and terminological variations. During this phase, the component service sends to the composite service, for each identity attribute a_i in the $NoMatchingAttr$ set, the set $Synset_i$ that contains a set of alternative character combinations and a set of synonyms. Thus, the composite service sends the sets $Synset_i$ to the client. The client verifies that for each identity attribute a_i, there is an intersection between $Synset_i$ and $AttrSet$. If this is the case attribute a_i is removed from $NoMatchingAttr$. Otherwise, if $NoMatchingAttr$ is not empty, the second phase is performed. During the second phase the client sends $CertSet$, the set of its proof-of-identity certificates to the composite service that forwards them to the component service. Thus, in the second phase of the matching process the component service tries to match the concepts corresponding to the identity attributes the

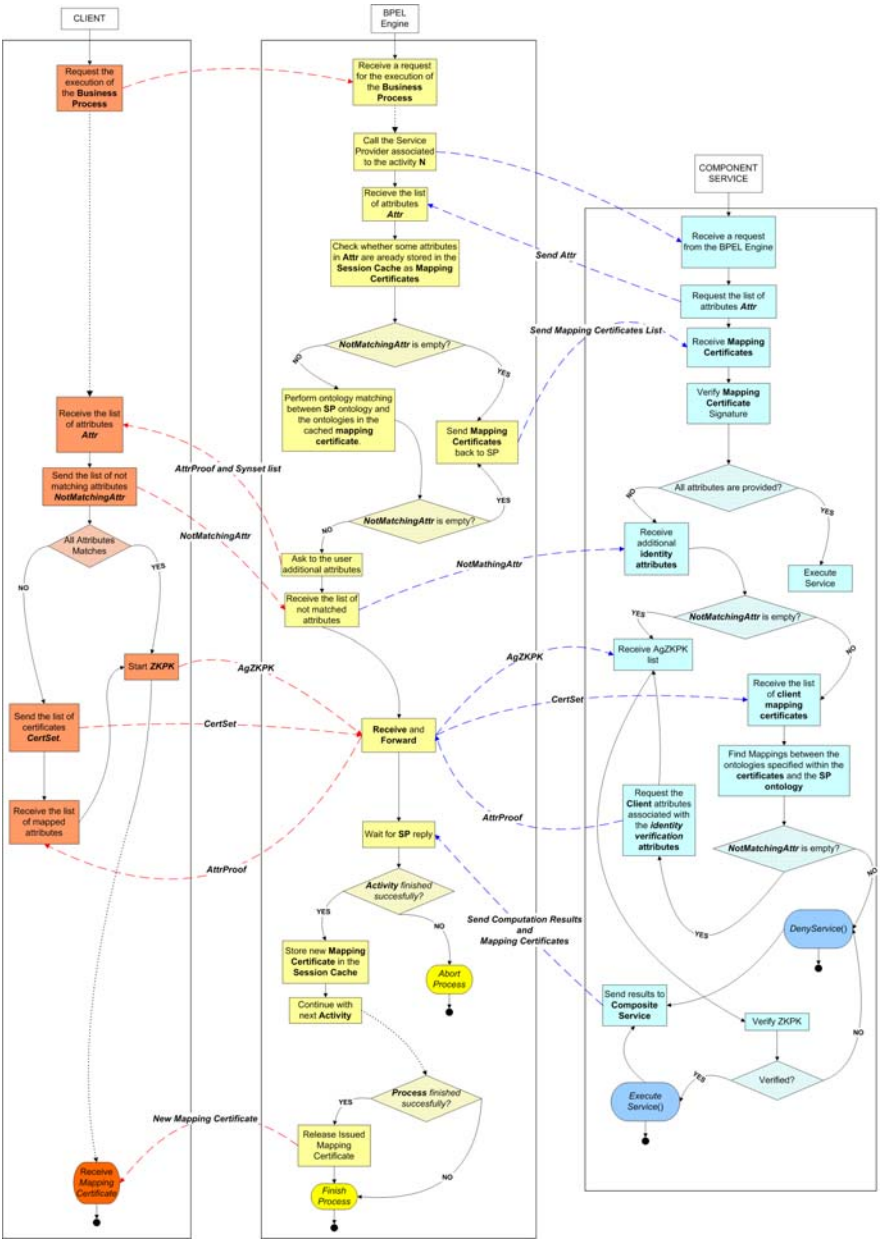


Fig. 2. Approach schema

client is not able to provide with concepts from the ontologies of the services which have issued the proof-of-identity certificates. Only matches that have a confidence score s greater than a predefined threshold are selected. The acceptance threshold is set up by the component service to assess the matches'

validity. The greater the threshold, the greater is the similarity between the two concepts and thus higher is the probability that the match is correct. If the component service is able to find mappings for its concepts, it then verifies by using the information in the proof-of-identity certificates that each matching concept matches a client's attribute *Attr*. If this check fails, the component service notifies the composite service that terminates the interaction with the client.

Algorithm 2: `Verification()`

Input:
Output:

(1) **C**: `Receive`($Match$)
(2) $AttrMathces$.`Add`($Match$)
(3) **foreach** $\langle Attr, Id_i \rangle \in AttrMatches$
(4) $\{M_i, \sigma_i\}$:= `Select`($RegCert$, $Attr$)
(5) $M = \prod_{i=1}^{n} M_i$
(6) randomly picks y, $s \in [1..q]$
(7) $d = g^y h^s \pmod{p}$
(8) `Send`($\{M_1, \ldots, M_n\}$, $\{\sigma_1, \ldots, \sigma_n\}$, M, σ, d);
(9) **CS**: `Receive`($\{M_1, \ldots, M_n\}$, $\{\sigma_1, \ldots, \sigma_n\}$, M, σ, d)
(10) randomly picks $e \in [1..q]$
(11) `Send`(e)
(12) **C**: `Receive`(e)
(13) $u := y + em \pmod{q}$ where $m = m_1 + m_2 + + m_n$
(14) $w := s + er \pmod{q}$ where $r = r_1 + r_2 + \ \ldots \ + r_n$
(15) `Send`(u, w)
(16) **CS**: `Receive`(u, w)
(17) **if** $g^u h^w == dM^k \pmod{p} \wedge \sigma == \prod_{i=1}^{t} \sigma_i$)
(18) `Execute`(**S**);
(19) `IssueCertificate`();
(20) **else**
(21) `Send`(**Service Denied**)

5.2 Multi-factor Authentication

Once the client receives $Match$, the set of matched identity attributes from the composite service, it retrieves from the registrar or from its $RegCert$ the commitments M_i satisfying the matches and the corresponding signatures σ_i. The client aggregates the commitments by computing $M = \prod_{i=1}^{n} M_i = g^{m_1+m_2+\ldots+m_i} h^{r_1+r_2+\ldots+r_i}$ and the signatures into $\sigma = \prod_{i=1}^{n} \sigma_i$, where σ_i is the registrar 's signature on the committed value $M_i = g^{m_i} h^{r_i}$. According to the ZPK protocol, the client randomly picks y, s in $[1, ..q]$, computes $d = g^y h^s \pmod{p}$, and sends d, σ, M, M_i , $1 \leq i \leq t$, to the composite service that on in turn sends these values to the component service. The component service sends back a random challenge $e \in [1, .., q]$ to the client. Then the client computes $u = y + em$

(mod q) and $v = s + er \pmod{q}$ where $m = m_1 + \ \ldots\ m_t$ and $r = r_1 + \ \ldots\ r_t$ and sends u and v to the composite service. The composite service forwards u and v to the component service. The component service accepts the aggregated zero knowledge proof if $g^u h^v = dc^e$. If this is the case, the component service checks that $\sigma = \prod_{i=1}^{n} \sigma_i$. If also the aggregate signature verification succeeds, the component service releases a proof of identity certificate to the client. The certificate states that client's identity attributes in the $Match$ set are mapped onto concepts of the component service ontology and that the client has successfully proved the knowledge of those attributes. The composite service sends the proof-of-identity certificate to the client and stores a copy of the certificate in its local repository $CertRep$. The proof-of-identity certificate can be can be provided by the composite service to another component service to allow the client to prove the knowledge of an attribute without performing the aggregate ZKP protocol. The component service that receives the certificate has just to verify the validity of the certificate.

Example 1. Assume that a user Bob submits a loan request to the loan service introduced in Example 1. The risk assessment service wants to verify Bob identity and it asks him to provide $DrivingLicense$, $CarRegistration$ and $EmployeeID$ identity attributes. Bob provides to the loan service the aggregate proof of $DrivingLicense$, $CarRegistration$ and $EmployeeID$ to the loan service, that forwards them to the risk assessment service. The risk assessment service verifies by carrying out an aggregate ZPK protocol with Bob that he owns $DrivingLicense$, $CarRegistration$ and $EmployeeID$ and release to Bob a proof-of-identity certificate that asserts Bob has $DrivingLicense$, $CarRegistration$ and $EmployeeID$ identity attributes. Therefore, when the loan approval service requires Bob to prove the possession of $EmployeeID$ and $CreditCard$ identity attributes, the loan service requests to Bob only to provide $CreditCard$ identity attribute and sends the proof-of-identity certificate released by the risk assessment service to the loan approval service.

6 System Architecture and Implementation

In this section we discuss the system architecture that supports our multi-factor identity attributes authentication for business processes. We assume that our processes are implemented as WS-BPEL business processes, that is, as business processes in which each component service is implemented by a Web service. The main components of the architecture are: the BPEL engine, the `Identity Attribute Requester` module, the `Client`, the `Registrar`, the `Identity Verification Handler` module, and the component Web services. The WS-BPEL engine is responsible for scheduling and synchronizing the various activities within the business process according to the specified activity dependencies, and for invoking Web services operations associated with activities. The `Identity Attribute Requester` module extends the WS-BPEL engine's functions by carrying on the communication with the client asking for new identity attributes whenever necessary. The `Identity Attribute Requester` keeps in a

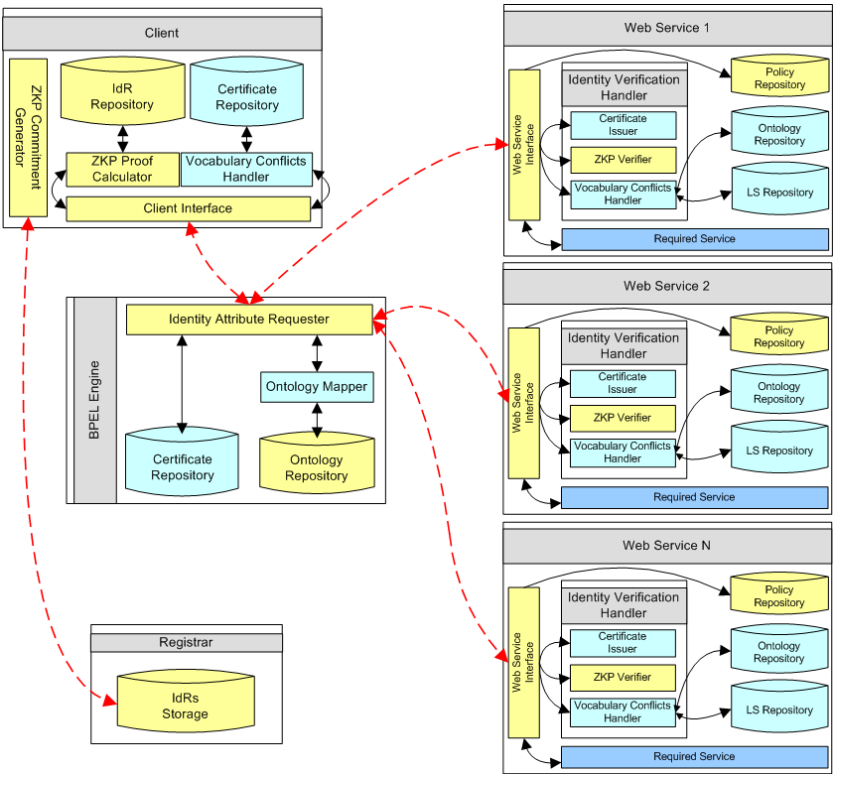


Fig. 3. System architecture

local repository the mapping certificate associated with previous clients identity verifications. The `Client` supports the functions to trigger the execution of the WS-BPEL business process, to select the identity attributes matching the ones requested by the component services, and to generate the aggregate ZKP of the matched attributes. The `Registrar` component provides functions for storing the clients' identity records and retrieving the public parameters required in the AgZKPK protocol. The `Identity Verification Handler` intercepts the components services invocation messages and provides functions for matching client identity attribute names and performing the aggregate ZKP verification. Finally, the component Web services support the operations that are orchestrated by the business process.

The `Identity Attribute Requester`, the `Identity Verification Handler` modules, and the component Web services have been implemented in JAVA. The `Identity Verification Handler` implements the identity attribute name matching protocol using the Falcon-AO v0.7 [2,3] ontology mapping API and WordNet 2.1 English Lexical database [6]. The `Client` application has been implemented in JSP while the `Registrar` has been implemented as a JAVA servlet. As BPEL engine we have chosen ODE. Finally, we have used Oracle 10g DBMS to

store clients' identity records, ontology mappings, set of synonyms, session data and mapping certificates.

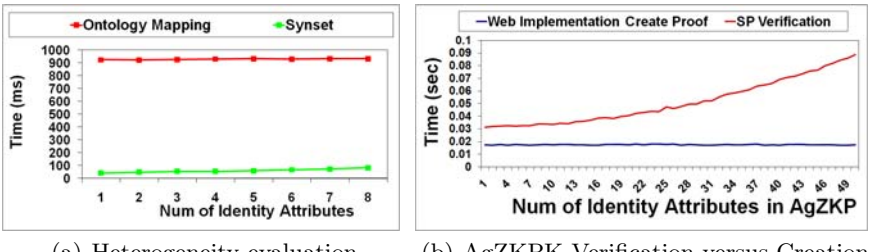
7 Experimental Evaluation

We have performed several experiments to evaluate the AgZKPK process that characterize the proposed approach to multi-factor identity verification and the identity attribute names matching process. To execute the tests we have developed a BPEL process composed by four component Web services and we have created a set of ontologies with an average cardinality of 60 concepts. We have carried out the following experimental evaluations:

- we have measured the time taken by a component Web service to perform the two different phases of the identity attribute names matching process by varying the number of identity attributes that have to be matched from 1 to 8. (Figure 4(a));
- we have measured the time taken by a component Web service to generate the aggregate ZKP by varying the number of identity attributes being aggregated from 1 to 50. (Figure 4(b));
- we have measured the time taken by a component Web service for aggregate ZKP verification execution time varying the number of identity attributes being aggregated from 1 to 50. (Figure 4(b));

The execution time has been measured in CPU time (milliseconds). Moreover, for each test case we have executed twenty trials, and the average over all the trial execution times has been computed.

Figure 4(a) shows the execution times of the two phases of the matching protocol for varying values in the number of identity attributes verified by a component service. The execution time of the first phase (green line) slightly increases and is around 60 ms. Instead, the time of the second phase is constant because even if the number of identity attributes to be match increases, this phase performs always the same operation, that is, matching two ontologies. Figure 4(b) reports the times to create an AgZKP and to verify it for varying values in the number of identity attributes being aggregated. The execution time to generate the AgZKP



(a) Heterogeneity evaluation (b) AgZKPK Verification versus Creation

Fig. 4. Experimental results

(represented by the blue line in the graph) is almost constant for increasing values in the number of identity attributes. The reason is that the creation of AgZKP only requires a constant number of exponentiations. By contrast, the time that the component Web service takes to perform identity attributes verification linearly increases with the number of identity attributes to be verified. The reason is that during the verification the component Web service is required to multiply all the commitments to verify the resulting aggregate signature.

8 Concluding Remarks

In this paper we have proposed a digital identity management approach for business processes. Our approach uses a combination of techniques from the area of semantic web and security protocols. We plan to extend this work in several directions. One direction is related to deal with heterogeneous identity negotiation protocols. The second direction is related to the definition of a language for identity verification policies that would allow service providers to specify conditions on identity attributes. We also plan to extend the AgZKPK protocol to verify that identity attribute's commitments satisfies such conditions.

References

1. Bhargav-Spantzel, A., Squicciarini, A.C., Bertino, E.: Establishing and Protecting Digital Identity in Federation Systems. Journal of Computer Security 14(3), 269–300 (2006)
2. Choi, N., Song, I.Y., Han, H.: A survey on ontology mapping. SIGMOD Record 35 (3), 34–41
3. Falcon, `http://iws.seu.edu.cn/projects/matching/`
4. Kalfoglou, Y., Schorlemmer, M.: Ontology mapping: the state of the art. The Knowledge Engineering Review 18(1), 1–31 (2003)
5. Pedersen, T.P.: Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992)
6. WordNet, `http://wordnet.princeton.edu/`