Identity Attribute-based Role Provisioning for Human WS-BPEL processes

Federica Paci
CS Department
Purdue University
West Lafayette, IN, US
paci@cs.purdue.edu

Rodolfo Ferrini
CS Department
Purdue University
West Lafayette, IN, US
rferrini@purdue.edu

Elisa Bertino
CS Department
Purdue University
West Lafayette, IN, US
bertino@cs.purdue.edu

Abstract

The WS-BPEL specification focuses on business processes the activities of which are assumed to be interactions with Web services. However, WS-BPEL processes go beyond the orchestration of activities exposed as Web services. There are cases in which people must be considered as additional participants to the execution of a process. The inclusion of humans, in turn, requires solutions to support the specification and enforcement of authorizations to users for the execution of human activities while enforcing authorization constraints. In this paper, we extend RBAC-WS-BPEL, a role-based authorization framework for WS-BPEL processes with an identity attribute-based role provisioning approach that preserves the privacy of the users who claim the execution of human activities. Such approach is based on the notion of identity records *and* role provisioning policies*, and uses Pedersen commitments, aggregated zero knowledge proof of knowledge, and Oblivious Commitment-Based Envelope protocols to achieve privacy of user identity information.*

1. Introduction

WS-BPEL has been developed to support the specification of automated business processes that orchestrate activities of multiple Web services. However, there are many applications and situations requiring that people be included as additional participants who can influence the execution of a WS-BPEL process. Human participation within a WS-BPEL process ranges from simple scenarios, such as manual approval, to complex scenarios where data is entered by a user or an interactive decision task is executed. Therefore, it is important to extend WS-BPEL to include the specification of activities that must be fully or partially performed by humans. The inclusion of humans, in turn, requires an access control model to support the specification and enforcement of authorizations to users for the execution of human activities while enforcing constraints, such as separation of duty, on the execution of those activities. One such model is RBAC-WS-BPEL, a role based access control model for inter-organization WS-BPEL processes. RBAC-WS-BPEL supports the specification of authorization information stating which role is allowed to execute which human activities in a process [5]. In order to make an authorization decision on the execution of a human activity, it is necessary to verify the user claiming the execution of the activity is assigned to a role which is granted the execution of the activity. Usually, users are assigned to roles which reflect the users' job function in the organization in which the business process is deployed. When a user has to perform a task in a business process, the user is requested to provide the password that has been given to him at the moment of enrollment, so that the system can authenticate the user and verify if the user is assigned to a role which is authorized to perform the task.

The use of roles, if on one hand, directly supports the specification of authorization in terms of business relevant roles and functions, on the other hand it introduces the problem of *role provisioning*, that is, the management of the assignments of roles to users. To address such shortcoming, in this paper, we propose a major extension to RBAC-WS-BPEL that allows one to dynamically assign users to roles based on properties that characterize the users. We refer to such properties as *identity attributes*. Examples of identity attributes are "social security number", "birth-date" and "employment". In our approach the assignments are driven by high-level identity attribute-based *role provisioning policies*; such policies specify that whenever the identity attributes of a user verify certain conditions, the user is authorized to use a role associated with the activity the user would like to execute. Such entails however addressing two issues. On one hand, to enable identity attribute-based role provisioning, the propagation of users's identity attributes across activities in the same process should be facilitated. On the other hand, identity attributes need to be protected as they may convey sensitive information about users and can be target of attacks leading to privacy breaches. To address the former issue, our approach uses some special purpose certificates. Such a certificate allows a user to prove that his/her identity attributes comply with the provisioning policies of a certain role. As conventional certificates, our certificates have temporal validity intervals and can be revoked. To address the latter issue, our approach uses

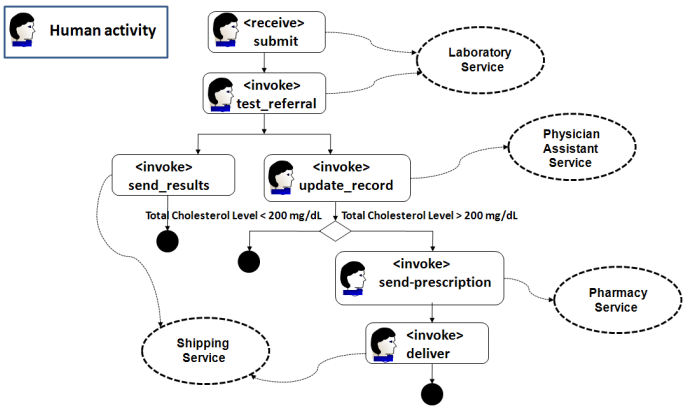

Figure 1. Patient Diagnosis and Treatment process

Pedersen commitments [7], aggregate zero knowledge proof of knowledge techniques [6], and OCBE protocols [4].

Our approach has many advantages. It is based on high-level access control policies; the use of high-level policies is crucial in order to comply with privacy laws and organizational policies. Auditors and high-level managers can directly see which are the access control policies enforced by the organization. The use of role provisioning based on identity attributes makes role management more flexible and supports multi-domain environments. For a user in a domain to be able to use a role, and thus execute an activity in a process from another domain, all the user has to do is to provide his/her identity attributes. The use of our privacy-preserving protocols secures the use of those identity attributes, thus preventing privacy breaches.

The remainder of the paper is organized as follows. Section 2 introduces a running example. Section 3 presents the main components of RBAC-WS-BPEL authorization model. Section 4 introduces aggregate zero knowledge proof of knowledge and OCBE protocols. Section 5 describes our privacy-preserving identity attribute-based role provisioning approach. Sections 6 and 7 discuss the system architecture and report experimental results, respectively. Section 8 outlines related work. Section 9 concludes the paper and outlines future research directions.

2. Running Example

We now introduce an example of a fictitious WS-BPEL process deployed in an hospital and consisting of several organizational units: the laboratory, the pharmacy, the drug distribution unit, and several diagnostic units.

The process orchestrates the following operations:

- the `submit` operation, by the `Laboratory` service, that allows a laboratory assistant to insert in the system the type of blood test for a patient and to print labels for the blood samples;
- the `test_referral` operation, by the `Laboratory` service, that allows a doctor to comment the test results;
- the `send_results` operation, by the `Shipping` service that allows a Laboratory employee to schedule the shipment of the blood test results to the patient;
- the `update_record` operation, by the `Physician Assistant` service, that allows a doctor or a nurse to read and update the patient medical record;
- the `send_prescription` operation, by the `Pharmacy` service, that allows a physician to send a drug prescription to the hospital's pharmacy.
- the `deliver` operation, by the `Shipping` service, that allows a pharmacy's employee to schedule the deliver of prescribed drugs to a patient.

The process is organized as follows (see Figure 1). First, a laboratory assistant invokes operation `submit` (`<receive> submit` activity). Then, a doctor of the hospital views and comments the blood test results of the patient by executing the operation `test_referral` (`<invoke> test_referral` activity). Once the test results are ready, the activities `<invoke> update_record` and `<invoke> send_results` are performed in parallel. After, the activity `<invoke> update_record` has been performed, a physician of the Blood Diseases unit prescribes a treatment to the patient if the level of total cholesterol in the patient's blood is higher than 200 mg/dL; otherwise the patient does not need any treatment. In the former case, the prescription is sent to the hospital's pharmacy by invoking the operation `send_prescription` (`<invoke> send_prescription` activity). Then, a pharmacy's employee schedules the delivery of the prescribed drugs to the patient by invoking operation `deliver` (`<invoke> deliver` activity).

3. RBAC-WS-BPEL Framework

RBAC-WS-BPEL applies to WS-BPEL business processes deployed in a single organization composed of different organizational units. RBAC-WS-BPEL inherits all the components of traditional RBAC models: users, roles, permissions, role hierarchies, user-role assignment and role-permission assignment relations. Moreover, RBAC-WS-BPEL supports the specification of authorization constraints such as separation of duty and binding of duty that restrict the set of users that can perform a given activity. In RBAC-WS-BPEL, permissions represent the ability to execute an activity of a WS-BPEL business process and are specified as tuples of the form $(A_i, Action)$ where A_i identifies an activity and $Action$ identifies the execution of the activity. Permissions are assigned to roles that are structured in a *role hierarchy* that defines a permission inheritance relation among the roles. Authorization constraint can be expressed

as a binary relation on the set of users or roles. An authorization constraint is represented by a tuple $< D, (A_1, A_2), \rho >$, where D is the user or role who has executed activity A_1, called the *antecedent activity*, A_2 is the *consequent activity* to which the constraints is applied and ρ is a relation on U, the set of users, or on R, the set of roles. A constraint $< D, (A_1, A_2), \rho >$ is *satisfied* if, whenever $x \in D$ performs A_1 and y performs A_2, then $(x, y) \in \rho$.

Authorizations and authorization constraints are evaluated to verify that the execution of the activity by the user does not violate any authorization constraints and does not prevent some other subsequent activities from completing [5].

Example 3.1: The following are examples of RBAC-WS-BPEL components that can be defined for our running example. We associate with the patient's diagnosis and treatment process the following role hierarchy: on top of the hierarchy there is the `Hospital Medical Director` role that dominates the `Department Director`, `Laboratory Director`, `Pharmacist`, and `Delivery Manager` roles. The `Department Director` dominates the roles `Primary Physician` and `Nurse` while the `Laboratory Director` role dominates the `Laboratory Assistant` role. The `Delivery Manager` dominates on his turn the `Delivery Boy` role. (`Nurse`, `<invoke> update_record`, execute) is an example of authorization that can be defined for activity `<invoke> update_record`: it states that `<invoke> update_record` can be performed by the `Nurse` role. We can also define the following authorization constraint: $\langle U, (\texttt{<invoke>test\_referral}, \texttt{<invoke>send\_prescription}, \neq \rangle$: it represents a separation of duty constraint for the activities `<invoke>test_referral`, and `<invoke>send_prescription`.

4. Basic Notions

In this section, we introduce the basic cryptographic notions on which our privacy-preserving role provisioning approach is based. We, first, introduce aggregated zero knowledge proof of knowledge (AgZKPK) protocol, and then OCBE protocols. AgZKPK is a cryptographic protocol to allow users to prove the ownership of multiple identity attributes without revealing anything about such identity attributes. OCBE protocols are cryptographic protocols to allow users to prove that their identity attributes satisfy a predicate without disclosing anything about such identity attributes. Note that by adopting AgZKPK and OCBE protocols no information about identity attributes is disclosed, and thus users' privacy is preserved.

4.1. Zero-knowledge proof of knowledge protocol

Zero-knowledge proof of knowledge (ZKPK) protocol allows a party U referred to as the prover, to convince the verifier, V, that U can open a commitment $c = g^x h^r$, without showing the values x and r in clear. Aggregate zero-knowledge proof of knowledge (AgZKPK) allows U to convince V that U knows how to open multiple commitments $c_i = g^{x_i} h^{r_i}$.

Aggregate Zero-knowledge proof of knowledge
A trusted party T generates public parameters G, p, g, h. A prover U who holds private knowledge of values $x_1, \cdots, x_n$ and $r_1, \cdots, r_n$ can convince a verifier V that U can open the Pedersen commitments $c_i = g^{x_i} h^{r_i}$ as follows.

1) U computes $c = g^x h^r$ where $x = x_1 + \cdots + x_n$ and $r = r_1 + \cdots + r_n$
2) U randomly chooses $y, s \in \mathbb{F}_p^*$, and sends V the element $d = g^y h^s \in G$.
3) V picks a random value $e \in \mathbb{F}_p^*$, and sends e as a challenge to U.
4) U sends $u = y + ex, v = s + er$, both in $\mathbb{F}_p$, to V.
5) V accepts the proof if and only if $g^u h^v = d \cdot c^e$ in G.

4.2. OCBE protocols

The Oblivious Commitment-Based Envelope (OCBE) protocols, proposed by Li and Li [4], ensure that a receiver Re can decrypt a message sent by a sender Se if and only if its committed value satisfies a condition given by a predicate in Se's access control policy, while Se learns nothing about the committed value. The possible predicates are comparison predicates $=, \neq, >, \geq, <$ and $\leq$.

The OCBE protocols are built with several cryptographic components:

1) The Pedersen commitment scheme.
2) A semantically secure symmetric-key encryption algorithm $\mathcal{E}$, for example, AES, with key length k-bits. Let $\mathcal{E}_{\mathsf{Key}}[M]$ denote the encrypted message M under the encryption algorithm $\mathcal{E}$ with symmetric encryption key Key.
3) A cryptographic hash function $H(\cdot) : \{0,1\}^* \rightarrow \{0,1\}^k$. When we write $H(\alpha)$ for an input α in a certain set, we adopt the convention that there is a canonical encoding which encodes α as a bit string, i.e., an element in $\{0,1\}^*$, without explicitly specifying the encoding.

Given the above notation, to give an idea how OCBE protocols work we describe the OCBE protocol for $\geq$ predicate, denoted as GE-OCBE. The OCBE protocols for other predicates can be derived and described in a similar fashion.

GE-OCBE Protocol

Parameter generation
T runs a Pedersen commitment setup protocol to generate system parameters $\mathsf{Param} = \langle G, g, h \rangle$, and outputs the order of G, p. In addition, T chooses another parameter ℓ, which specifies an upper bound for the length of attribute values, such that $2^\ell < p/2$. T also outputs $\mathcal{V} = \{0, 1, \ldots, 2^\ell - 1\} \subset \mathbb{F}_p$, and $\mathcal{P} = \{\mathsf{GE}_{x_0} : x_0 \in \mathcal{V}\}$, where

$$GE_{x_0} : \mathcal{V} \rightarrow \{\mathsf{true}, \mathsf{false}\}$$

is a predicate such that $\mathsf{GE}_{x_0}(x)$ is true if and only if $x \geq x_0$.

Commitment
T chooses an integer $x \in \mathcal{V}$ for Re to commit. T then randomly chooses $r \in \mathbb{F}_p$, and computes the Pedersen commitment $c = g^x h^r$. T sends x, r, c to Re, and sends c to Se.

Interaction

- Re makes a data service request to Se.
- Based on the request, Se sends to Re a predicate $\mathsf{GE}_{x_0} \in \mathcal{P}$.
- Upon receiving this predicate, Re sends to Se a Pedersen commitment $c = g^x h^r$.
- Let $d = (x - x_0) \pmod p$. Re picks $r_1, \ldots, r_{\ell-1} \in \mathbb{F}_p$, and sets $r_0 = r - \sum_{i=1}^{\ell-1} 2^i r_i$. If $\mathsf{GE}_{x_0}(x)$ is true, let $d_{\ell-1} \ldots d_1 d_0$ be d's binary representation, with d_0 the lowest bit. Otherwise if GE_{x_0} is false, Re randomly chooses $d_{\ell-1}, \ldots, d_1 \in \{0, 1\}$, and sets $d_0 = d - \sum_{i=1}^{\ell-1} 2^i d_i \pmod p$. Re computes ℓ commitments $c_i = g^{d_i} h^{r_i}$ for $0 \leq i \leq \ell - 1$, and sends all of them to Se.
- Se checks that $cg^{-x_0} = \prod_{i=0}^{\ell-1} (c_i)^{2^i}$. Se randomly chooses ℓ bit strings $k_0, \ldots, k_{\ell-1}$, and sets $k = H(k_0 \parallel \ldots \parallel k_{\ell-1})$. Se picks $y \in \mathbb{F}_p^*$, and computes $\eta = h^y, C = \mathcal{E}_k[M]$, where M is the message containing requested data. For each $0 \leq i \leq \ell - 1$ and $j = 0, 1$, Se computes $\sigma_i^j = (c_i g^{-j})^y, C_i^j = H(\sigma_i^j) \oplus k_i$. Se sends to Re the tuple

$$\langle \eta, C_0^0, C_0^1, \ldots, C_{\ell-1}^0, C_{\ell-1}^1, C \rangle.$$

Open
After Re receives the tuple $\langle \eta, C_0^0, C_0^1, \ldots, C_{\ell-1}^0, C_{\ell-1}^1, C \rangle$ from Se as above, Re computes $\sigma_i' = \eta^{r_i}$, and $k_i' = H(\sigma_i') \oplus C_i^{d_i}$, for $0 \leq i \leq \ell - 1$. Re then computes $k' = H(k_0' \parallel \ldots \parallel k_{\ell-1}')$, and decrypts C using key k'.

LE-OCBE, the OCBE protocol for the $\leq$ predicates, can be constructed in a similar way as GE-OCBE. Other OCBE protocols (for $\neq, <, >$ predicates) can be built on EQ-OCBE, GE-OCBE and LE-OCBE.

5. RBAC-WS-BPEL Role provisioning

In this section we describe how our identity attribute-based role provisioning approach is interleaved with RBAC-WS-BPEL process life cycle.

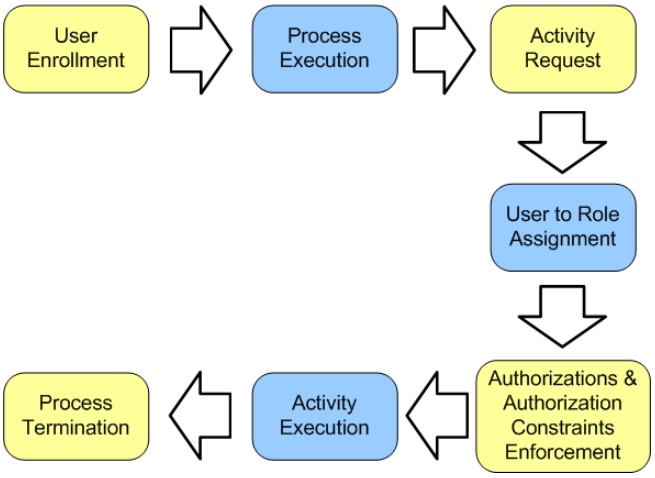

Figure 2. RBAC-WS-BPEL Business process life cycle

5.1. User Enrollment

In order to support our privacy-preserving role provisioning approach we assume that users enroll their identity attributes when they join the organization. Identity Providers (IdPs) issue certified identity attributes to users and control the sharing of such information. Users enroll then their certified identity attributes at an entity called *Identity Manager* (IM). The IM stores and manages information related to identity attributes that are used for role provisioning. Note that, unlike IdPS, the information stored at the IM does not include the values of the identity attributes in clear. In particular, the IM stores for each user an Identity Record that contains an identity tuple for each user's identity attribute m. An identity tuple consists of tag, an attribute descriptor, the Pedersen commitment of m, denoted as M_i, the signature of the IM on M, denoted as σ_i, and two types of assurance, namely *validity assurance* and *ownership assurance*. M_i is the Pedersen commitment of m and it is computed as $g^m h^r$, where g and h are generators in a group G of prime order p and r is a random secret from $\mathbb{Z}_p$ that is known only to the user. Instead, G, p, g and h are public parameters of the IM. 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.

The identity tuples of each registered user can be retrieved from the IM by the identity verifier (*offline* mode) or the IM can release to the user a certificate containing his/her identity record (*offline* mode).

5.2. Privacy-Preserving Role provisioning

In RBAC-WS-BPEL the execution of an activity is granted to a user if the user is assigned to a role which has the permission to execute the activity and the execution does not violate any authorization constraint. The assignment of users to roles is governed by role provisioning policies defined according to the following definitions.

Definition 5.1 (Attribute Condition): An attribute condition $\mathcal{C}ond$ is an expression that can take one of the following two forms: (1) "$name_A$ $\mathtt{op}$ l", where $name_A$ is the name of an identity attribute A, $\mathtt{op}$ is a comparison operator such as $=, <, >, \leq, \geq, \neq$, and l is a value that can be assumed by attribute A; (2) "$name_A$" where $name_A$ is the name of an identity attribute A.

Definition 5.2 (Role Provisioning Policies): A *Role provisioning Policy* $\mathcal{P}ol$ is an expression of the form "$\mathcal{R} \leftarrow \mathcal{C}ond_1, \mathcal{C}ond_2, \ldots \mathcal{C}ond_n$", $n \geq 1$, where $\mathcal{R}$ identifies a role and $\mathcal{C}ond_1, \mathcal{C}ond_2, \ldots \mathcal{C}ond_n$ are attribute conditions.

Example 5.1: An example of role provisioning policy for the role *Hospital Medical Director* is:
Hospital Medical Director $\leftarrow$ *Bachelor* = *Medical*, *Age* $>$ 55.
Such policy states that for a user to be assigned to the *Hospital Medical Director* role he/she must prove the possession of a medical bachelor degree and to be older than 55.

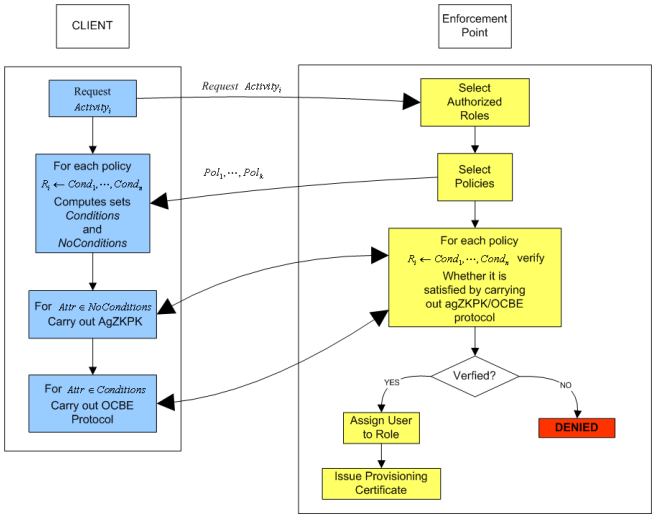

Figure 3. Role provisioning Protocol Steps

The evaluation of role provisioning policies against user's identity attributes is performed when the user claims the execution of a human activity of a WS-BPEL process. In what follows, we propose a protocol to verify whether a user satisfies a role provisioning policy while preserving user's privacy. The protocol is performed by the client running on behalf of the user[1], and the enforcement point which decides whether the execution of a human activity can be granted to the client. The main steps of the protocol are summarized in Figure 3. When a client claims the execution of a human activity *Activity*$_i$, the enforcement point selects the roles that have the permission to perform *Activity*$_i$. Then, the enforcement point selects the role provisioning policy defined for the roles authorized to execute *Activity*$_i$ and forward them to the client. For each policy received, the client groups the identity attributes it has to provide according to the type of attribute conditions in which they appear. The identity attributes A in the attribute conditions of the form "$name_A$" are listed in the set *NoCondition*, while the identity attributes in the attribute conditions of the form "$name_A$ $\mathtt{op}$ l" are part of the set *Condition*. To satisfy the condition of the form "$name_A$", the client has to prove the possession of the identity attributes in the set *NoCondition* by carrying out an AgZKPK protocol with the enforcement point. First, the client retrieves the Pedersen commitments of the identity attributes m_i in *NoCondition* set and the corresponding signatures σ_i. Then the client computes the aggregated commitment M and the aggregated signature σ and sends them to the enforcement point. Then, the client and the enforcement point perform the steps described in Section 4.1.

By contrast, to prove the satisfaction of policy conditions of the form "$name_A$ $\mathtt{op}$ l, the client has to perform a OCBE protocol for each of the identity attributes in *Condition* set. The OCBE protocol to be executed corresponds to the comparison operator specified in the policy conditions. The OCBE protocol is executed between the enforcement point which acts as the sender Se, and the client that plays the role of the receiver Re. We have added an additional step to the protocol to let the enforcement point know if the client satisfies the policy condition. The enforcement point chooses the message M to be a random bit string, which will be used as a secret of Se. At the end of the protocol, after opening the envelope, Re shows Se the decrypted message M'. The client satisfies the conditions in the policy if $M = M'$, or fails if otherwise. Since the random bit string M contains no useful information, a qualified client must choose to show the correctly decrypted secret message M, in order to continue the interaction with the enforcement point.

If a client proves that it satisfies all the conditions $\mathcal{C}ond_1$, $\mathcal{C}ond_2, \ldots \mathcal{C}ond_n$ in a role provisioning policy $\mathcal{P}ol : \mathcal{R} \leftarrow \mathcal{C}ond_1, \mathcal{C}ond_2, \ldots \mathcal{C}ond_n$", $n \geq 1$, it is assigned to the role $\mathcal{R}$. The enforcement point issues a certificate to the client asserting the roles the client has been assigned and the identity attributes the possession of which has been verified by the enforcement point. We denote such certificate as *role provisioning certificate*.

1. In the rest of the paper, for presentation simplicity, we use the term client to refer both the actual end-user and to the system running on behalf of the user on the user machine or other personal device or proxy

Definition 5.3 (Role Provisioning Certificate): Let P be the enforcement point and C be a client of a business process BP. A role provisioning certificate released by P to C upon a successful role provisioning policy verification is a tuple $\langle\ Issuer, Owner, IdentityAttr, Roles, Validity, Signature\rangle$ where $Issuer$ is the identifier of P, $Owner$ is the identifier of C, $IdentityAttr$ is the set of identity attributes that P has verified are owned by C, $Roles$ is the set of roles to which C has been assigned,$Validity$ is a tuple $(NotBefore, NotAfter)$ where $NotBefore$ is the issuance date of the certificate and $NotAfter$ is the date after which the certificate is no longer valid and $Signature$ is the signature of P on the whole certificate.

Role provisioning certificates make unnecessary for the client to prove it satisfies a role provisioning policy every time the clients requests the execution of a human activity. When a client claims the execution of an activity *Activity*$_i$, the client just present the role provisioning certificates, if any, to the enforcement point. The enforcement point, first, evaluates if the certificate is not expired and the validity of the signature's retrieving issuer's public key. If the certificate is valid, the enforcement point checks that the client is assigned to one of the roles that are authorized to execute *Activity*$_i$. If this is not the case, the enforcement point checks if there is at least one role to which the client is assigned that dominates the roles that have the permission to perform *Activity*$_i$. If such role can be found, the enforcement point starts the authorization process based on the evaluation of the permission the client has and on authorization constraints. Otherwise, the enforcement point tries to assign the client to one of the roles that have the permission of performing *Activity*$_i$ based on role provisioning policies defined for these roles.

Example 5.2: Assume a client John Smith claims the execution of activity `<invoke> test-referral` in the patient diagnosis and treatment process introduced in Example 1. The role authorized to perform such activity is `Laboratory Assistant` the role provisioning policy of which is *Laboratory Assistant* $\leftarrow$ *Certified_LaboratoryAssistant*, *Bachelor = Medical Technology*. This policy requires the client to have a laboratory assistant certification and to have a bachelor in Medical Technology. The client, running on behalf of John Smith, to prove he satisfies the policy for role `Laboratory Assistant`, carries out an AgZKPK protocol for the condition *Certified_LaboratoryAssistant* and a OCBE protocol for the policy condition *Bachelor = Medical Technology* with the enforcement point. If the execution of AgZKPK and OCBE protocols is successful, the enforcement point issues to the John the following role provisioning certificate that John can present when he will request the execution of another human activity: $\langle EP,\ JohnSmith,\ \{Certified\_LaboratoryAssistant, Bachelor\},\ \{LaboratoryAssistant\},(01-15-2009, 02-15-2009),\ 3AFJSfFIO43 = 0D33SF\rangle$.

6. System Architecture

The RBAC-WS-BPEL architecture includes several components shown in Figure 4. 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 RBAC-WS-BPEL Enforcement Service offers two WSDL interfaces, one for the process and one for the clients. The first interface provides the operations `intiateActivity` and `onActivityResult` for starting and completing the execution of a WS-BPEL human activity respectively. The second interface provides the operation `listActivity`, that allows clients to visualize the activities they can claim and the operation `claimActivity` to claim and execute them [6]. The Client Interface allows a user to invoke the operations `listActivity` and `claimActivity` of RBAC-WS-BPEL Enforcement Service.

We have extended the RBAC-WS-BPEL architecture with an additional component, that is, the Identity Manager Service. Such components provides the functionalities for user enrollment. It manages users' identity records that are stored in a dedicated repository. We have also extended the Client Interface with the functionalities to prove the satisfaction of a role provisioning policy by carrying out AgZKPK and OCBE protocols and to manage the role provisioning certificates and the certificates containing the identity records issued by the Identity Manager Service.

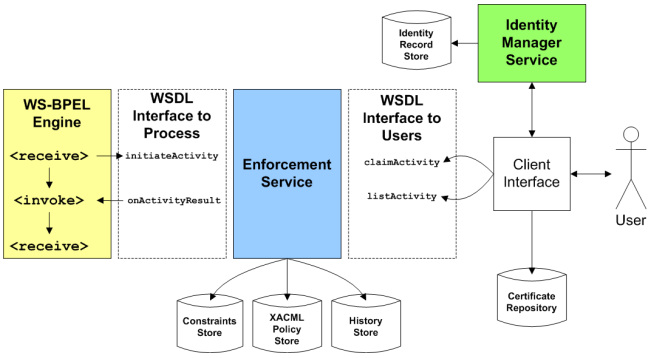

Figure 4. RBAC-WS-BPEL architecture

7. Implementation and Experimental Results

We have implemented the role provisioning protocol and integrated it in the RBAC-WS-BPEL Client Interface and Enforcement Service. We have also implemented the Identity Manager Service as a Java servlet. Moreover, we have chosen ODE as BPEL engine and Oracle 10g DBMS to store

clients identity records and role provisioning certificates. We have performed several tests to evaluate the performance of the privacy-preserving role provisioning approach. We have collected the following data:

1) the time taken by the Client Interface to generate the AgZKP by varying the number of policy conditions to be verified from 1 to 50;
2) the time taken by the RBAC-WS-BPEL Enforcement Service to verify an AgZKP by varying the number of policy conditions to be verified from 1 to 50;
3) the time taken by the Enforcement Service to verify a role provisioning policy condition by using OCBE protocols by varying the value of parameter ℓ from 5 to 20; the time includes the time to create the envelope and the time to verify that the value sent by the user matches the encrypted value sent by the Enforcement Service;
4) the time taken by the Client Interface to generate the commitments $c_i = g^{d_i} h^{r_i}$, $0 \leq i \leq \ell - 1$ and the time to open the envelope sent by the Enforcement Service by varying the value of parameter ℓ from 5 to 20; the commitments $c_i = g^{d_i} h^{r_i}$, $0 \leq i \leq \ell - 1$ are computed to prove to the Enforcement Service that the user satisfies the condition in the role provisioning policy.

To run the experiment we have generated a WS-BPEL process composed by 21 activities, a set of 50 potential users, a role hierarchy of 7 roles, and a set of role provisioning policies of increasing complexity. We have measured the execution time in CPU time (milliseconds). Moreover, for each test case we have executed twenty trials, and computed the average execution time over all the trial executions.

Figure 5 reports the times to generate an AgZKP and to verify it for varying values in the number of identity attributes specified in the role provisioning policy. The execution time to generate the AgZKP (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 RBAC-WS-BPEL Enforcement 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 Enforcement Service has to multiply all the commitments to verify the resulting aggregate signature.

Figure 6 shows that the value of parameter ℓ has a high impact on the execution time of the policy conditions' verification. Such time linearly increases with the value of ℓ. The reason is that when the value of ℓ increases, the Enforcement Service has to compute a higher number of $\sigma_i^j = (c_i g^{-j})^y, C_i^j = H(\sigma_i^j) \oplus k_i$ to be sent to the Client Interface and the Client Interface, in order to decrypt the envelope, has to compute a higher number of $\sigma_i' = \eta^{r_i}$,

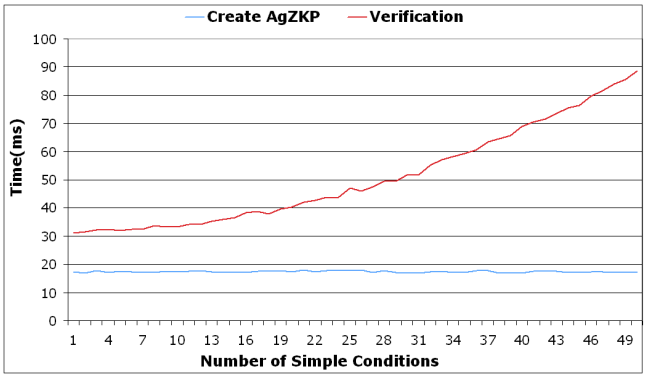

Figure 5. First and Second Test Cases Results

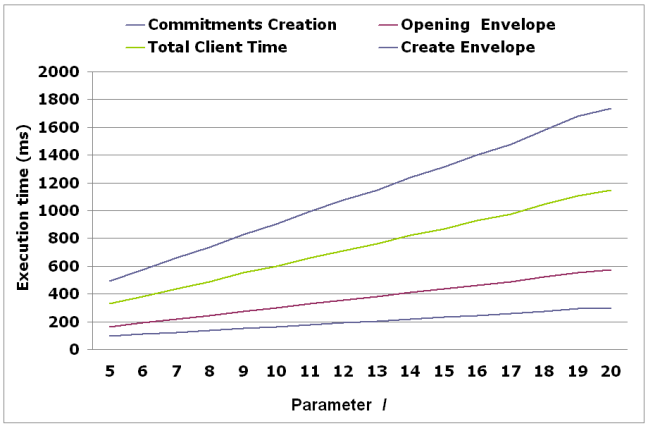

Figure 6. Third and Fourth Test Cases Results

and $k_i' = H(\sigma_i') \oplus C_i^{d_i}$, for $0 \leq i \leq \ell - 1$. Therefore, in the implementation of our approach, the parameter ℓ must be kept as small as possible in order to reduce the computational cost.

8. Related Work

The closest proposals to ours are BPEL4People [1], the approach by Koshutanski et al. [3], and the one by Xiangpeng et al. [8]. BPEL4People is a recent proposal to handle person-to-person WS-BPEL business process. With respect to RBAC-WS-BPEL, in BPEL4People users that have to perform the activities of a WS-BPEL business process are directly specified in the process by user identifier(s) or by groups of people's names. No assumption is made on how the assignment is done or on how it is possible to enforce constraints like separation of duties.

Koshutanski et al. propose an authorization model for business processes based on Web services. The model by Koshutanski et al. and RBAC-WS-BPEL both assume an RBAC model and support authorizations constraints on the set of users and roles. They also consider the problem of taking authorization decision on the execution of business

process's activities. The main difference with RBAC-WS-BPEL is in the approach to take authorization decision. In the model by Koshutanski et al., an authorization decision is taken by orchestrating the authorization processes of each Web service, the activities of which are orchestrated in the business process, while in RBAC-WS-BPEL an authorization decision is taken independently for each activity in the process.

Xiangpeng et al. propose an RBAC access control model for WS-BPEL business process. Roles correspond to `<partnerRole>` elements in the WS-BPEL specification and are organized in a hierarchy. Permissions correspond to the execution of the basic activities in the process specification. In addition, separation of duty constraints can be specified. Compared to such model, RBAC-WS-BPEL's BCPL provides a constraints language supporting the specification of a broader range of authorizations constraints.

All such previous approaches do not make any assumption on how the assignment of users to roles is performed. Therefore our approach is the the first to propose a concrete solution to support human activity execution in WS-BPEL processes and to assign users to roles in order to make authorization decision on the execution of such activities.

9. Conclusion

In this paper we have proposed a protocol supporting a dynamic assignment of users to roles that have the permission to execute a human activity claimed by the users. The assignment is based on the notion of role provisioning policy that specifies conditions on the users'identity attributes. Since identity attributes encode sensitive information about users, we have developed a privacy-preserving role provisioning protocol that allows users to prove compliance with role provisioning policies while disclosing no information about their identity attributes. We have implemented our role provisioning protocol in the context of RBAC-WS-BPEL, a RBAC access control framework for WS-BPEL processes previously developed by us. We have implemented our protocol and performed several test to evaluate the protocol's performance.

Acknowledgment

This work was supported in part by the National Science Foundation under the ITR Grant No. 0428554 "The Design and Use of Digital Identities", by AFOSR grant A9550-08-1-0260, and by the U.S. 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. The I3P is managed by Dartmouth College.

References

[1] A. Agrawal et al. *WS-BPEL Extension for People (BPEL4People), Version 1.0*, 2007. http://www.adobe.com/devnet/livecycle/pdfs/bpel4people_spec.pdf.

[2] A. Alves. et al. Web Services Business Process Execution Language, Version 2.0, OASIS Standard, April 2007. http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.pdf.

[3] H. Kostutanski and F. Massacci. An access control framework for business processes for web services. In *Proceedings of the ACM Workshop on XML Security*, pages 15–24, October 2003.

[4] J. Li and N. Li. Oacerts: Oblivious attribute certificates. *IEEE Transactions on Dependable and Secure Computing*, 3(4):340–352, 2006.

[5] F. Paci, E. Bertino, and J. Crampton. An access control framework for ws-bpel. *International Journal of Web service Research*, 5(3):20–43, 2008.

[6] F. Paci, E. Bertino, S. Kerr, A. Lint, A. Squicciarini, and J. Woo. Veryidx - a digital identity management system for pervasive systems (invited paper). In *Proceedings of 6th IFIP Workshop on Software Technologies for Future and Embedded Ubiquitous Systems (SEUS)*, October 2008.

[7] T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO '91: Proceedings of the 11th Annual International Cryptology Conference on Advances in Cryptology*, pages 129–140, 1991.

[8] Z. Xiangpeng, A. Cerone, and P. Krishnan. Verifying bpel workflows under authorisation constraints. In *In Proceedings of Fourth International Conference on Business Process Management (BPM 2006)*, 2006.