The ISC Int'l Journal of Information Security
ISeCure

July 2011, Volume 3, Number 2 (pp. 103–120)
http://www.isecure-journal.org

A Model for Specification, Composition and Verification of Access Control Policies and Its Application to Web Services

Zahra Derakhshandeh [1,*] and Behrouz Tork Ladani [2]
[1] *Department of Computer Engineering, Sheikhbahaee University, Iran*
[2] *Department of Computer Engineering, University of Isfahan, Iran*

ARTICLE INFO.

Article history:
Received: 18 December 2010
Revised: 8 February 2012
Accepted: 9 February 2012
Published Online: 19 May 2012

Keywords:
Access Control Policy (ACP), Verification, Web Service Composition, BPEL, Constrained Policy Graph (CPG)

ABSTRACT

Despite significant advances in the access control domain, requirements of new computational environments like web services still raise new challenges. Lack of appropriate method for specification of access control policies (ACPs), composition, verification and analysis of them have all made the access control in the composition of web services a complicated problem. In this paper, a new independent formal model called Constrained Policy Graph (CPG) for specification of ACPs and their composition as well as verification of conflict or incompatibility among the ACPs is represented. It is shown how CPG can be used in modeling and verification of web service composition ACPs. Also the application of CPG for modeling policies in BPEL processes -as the most common composition method for web services- is illustrated.

1 Introduction

The goal of access control is to limit the operations of an entity only to services and resources that the entity has the right to access them. To specify such a limitations, Access Control Policies (ACPs) are used. Application of ACPs to achieve the required security level in today's systems is still a challenge. In today's applications, a system is often defined by a number of access control policies. In other words, in practice, a single policy cannot provide the required security level of a system. For this reason there is a need for composition of the policies to obtain the overall policy of that system. Although access control in a monolithic system may not seem to be a complex problem, it will be difficult in distributed systems and integrated systems made of several components.

As an illustrating case, organizations require guaranteeing the enforcement of their policies while offering their services. In many cases, web services participating in a composition enjoy acceptable security policies, while the compositional process does not guarantee the security policies of the partners in the composition. There might be cases in which the partner's web services have contradicting ACPs. So it is required to be able to extract the composed access control policy resulting from the organization's processes so that the resulted policy can be analyzed for the presence or absence of conflict according to the partner web services' ACPs.

In spite of remarkable progress in the field of access control, most of the methods (such as [1, 2]) are based on the expression and application of single policies without considering the ability of policy composition necessarily. Moreover, it seems essential to use a method which can provide the possibility of verifying conflicts in composed policies. Although methods such as [3–6] have considered the issue, but these methods do not satisfy the comprehensiveness level expected

* Corresponding author.
Email addresses: z.derakhshande@shbu.ac.ir (Z. Derakhshandeh), ladani@eng.ui.ac.ir (B. Tork Ladani)
ISSN: 2008-2045 © 2012 ISC. All rights reserved.

from a specification and analysis method for most of today's systems (See Section 5).

In this paper, a model for specification of the ACPs called Constrained Policy Graph (CPG) is presented. Note that preliminary versions of some parts in this paper have been already published in [27, 28]. CPG not only provides the possibility of constrained (conditional) and nested expression of the policies, but also includes the required rules for composition of the policies. We define the constraints for subjects, objects, and actions (three basic parts of each access control policy) as well as the possibility of time-bounded policy definition. Additionally, by applying a collection of rules, the possibility of rights and information transfer in the system would be prepared. This way, non-explicit transfers which are performed implicitly as a result of the relations among different entities in the system are also discovered. Using the proposed model, the composed policy can be analyzed, and the presence or absence of conflict property in the policies can be verified.

In the rest of paper, the CPG model will be introduced in Section 2 as a general model and some examples of its application in specification, combination and verification of ACPs as well. In Section 3, the model is used to specify and verify the ACPs in web services as a case of real applications. We also present a method for the extraction of CPG model resulting from the compositional web services to study and analyze the ACPs of the existing compositional web services (to use for the existing web service compositions). In Section 4, a sample of the application of the above method for modeling the policy of BPEL processes (as the most common composition method for web services [7]) is presented. The review and discussion on the related works are dealt with in Section 5. Finally we conclude the paper in Section 6.

2 Constrained Policy Graph (CPG)

2.1 Model specification

A way to model the ACPs should provide a method for explaining of different forms of accesses according to the related system's requirements. ACPs are expressed by subjects, objects and actions. Subjects can be users, groups, roles or applications. Objects can be files, documents, or also any subjects. An ACP determines whether a subject is permitted to perform a set of actions on an object or not. Vimercati, et al in [12] discuss the main features of ACPs and the need for considering different constraints in the specification of ACPs.

We suggest a method for modeling ACPs. In this method, the protection state of a system is represented as a directed graph in which vertices are entities of system and edges are conditionally labeled. The method inherits some of its basic concepts from Take-Grant (TG) protection model [8]. For a short review on TG and make a comparison with the proposed method you can refer to the Section 5.

In the beginning we need to point out the importance of considering constraints in the ACP specification. In our access control model, the applied constraints on the system can influence a subject, an object or an action according to the system requirements. In other words, if a subject has a special role, it is possible to have the capability of performing an action toward an object.

- A subject has a right (or a set of rights) only on specific parts of an object (e.g. a student has only the right of reading digital documents among all library documents or XML documents in a database).
- A subject has a right toward the related object only for performing a specific operation (specific purpose). It is very useful in modeling the web services policies. For example, a user has a specific right on database of a web service (e.g. read and write) only for invoking a special operation of web service while the right of the user toward database may change for other purposes (i.e. different operations).
- On the other hand, an important requirement, common to many applications, is related to the temporal properties of access permissions. In such applications, permissions are granted only for specific periods of time (e.g. periodic authorizations for optimizing resource usage). So, the access control model must have the capability of defining temporal constraints in its ACP specification.

Now we introduce some definitions required for expressing the proposed model.

Definition 1. Basic rights (R): R is the set of negative or positive access rights (include at least read and write) which is considered in the system.

Definition 2. Constrained Policy Graph (CPG): If S and O be the set of subjects and objects respectively, a CPG Ψ is the pair(V, L), where:

- $V \subseteq S \cup O$ is a set of vertices,
- L is set of edges where $L \subseteq V \times V \times \mathit{Label}$ and Label is a set and for each $l \in \mathit{Label}$, we have either $l = C : R$ or $l = C : P_D$ ($\mathit{read}\ R/P_D$ in condition of C) in which:
  - ○ R is a set of basic rights ($P \subseteq R$),
  - ○ $P_D = (V', L')$ is a CPG defined recursively with $D \in V'$, and

- C is the required constraints for possessing P by the source vertex of the corresponding edge toward the destination vertex. It is defined as the quadruple $(C_s, C_o, C_\alpha, C_\Theta)$ in which:
  - C_s and C_o are the constraints over subject and object respectively,
  - C_α determines the operations in which P is valid by subject toward object, and
  - C_Θ indicates a set of temporal constraints.

Note that since we deal with specifying access control policies, subjects and objects do not refer to specific identities but instead show classes of subjects or objects respectively (e.g. applicant as subject and library document as object in a library system). C_s specifies the constraints under which "s" could access to "o". C_o on the other hand specifies the constraints that under which "o" is accessible by "s". We may use labels on subject and object nodes in CPG to enhance the readability of it. However theses labels are only symbolic names for variables s and o respectively; any enforceable constraint should be specified explicitly in C.

Like TG model subjects are represented by ●, objects are represented by ○, and if some vertices play both subject and object roles they are represented by ⊗ symbol. The elements of set C and the way of their definition are explained in more detail as follows:

- C_s: It is a constraint on the subject, i.e. only if the subject follows this constraint, its rights (labeled on the related edge) are valid toward the object. In its basic form, C_s is a simple predicate related to the subject. We use an obligatory parameter referring to its source vertex to show its related subject (necessary for CPG combination purpose). C_s can be a logical formula in general. If a subject possesses a right toward the object without any constraint on that subject, C_s will be true (T) and if the subject cannot access the object in any case, C_s will be false (F). The following grammar defines C_s:

$$C_s = C_s \wedge C_s \mid C_s \vee C_s \mid \neg C_s \mid \beta \mid T \mid F$$

  β is a simple predicate in two-variable-fragment $\mathcal{L}^2_{\approx}$ of first order logic which has been shown that can be decided with first-order resolution methods [31]. $\mathcal{L}^2_{\approx}$ is the set of formulas that do not contain function symbols, that possibly contain equality, and that contain at most two variables. Predicates are expressed depending on the related system requirements. Variables in β refer to the source subject, and are used to make the relationship between the predicates.
- C_o: Subject accesses toward the objects can be restricted depending on the related objects' contents. This restriction would be achieved by conditions (constraints) called "content-dependent conditions". We will show these conditions by using C_o when a subject possesses a right toward the specific part of the object. Like C_s, C_o is a simple predicate (with an obligatory parameter showing its corresponding object) in its basic form and a logical formula in general. We have:

$$C_o = C_o \wedge C_o \mid C_o \vee C_o \mid \neg C_o \mid \beta \mid T \mid F$$

  Like before, β is a simple predicate in $\mathcal{L}^2_{\approx}$ same as in C_s which is expressed depending on the related system requirements.
- C_α: A subject may possess different rights on the object based on performing different operations on that object. Note that the intended operations are apart from access rights which are also operations. In other words, C_α determines higher level operations which cause different rights for the subject toward the object e.g. in a web service, only when its method A is invoked, the corresponding subject have a special access right, but may be banned for its other methods. This constraint is a logical formula as well. Each predicate has two parameters (s and o) which point out the related subject and object of the corresponding edge, respectively. Grammar of C_α is defined as grammar of C_s and C_o, too.

  An example which uses C_α is in the modeling of web services policies, i.e. depends on invoking different methods (operations) of a web service by a subject, the subject has different rights toward the related web service.
- C_θ: This constraint, in its basic form, expresses an interval $([t_s, t_d])$ and it means that the mentioned right is valid in the interval between ts and td (an interval is an ordered set of temporal points in which $t_s \leqslant t_d$). C_θ can be extended to a combination of intervals in its general form. The grammar of C_θ is as follows:

$$C_\theta = C_\theta \wedge C_\theta \mid C_\theta \vee C_\theta \mid \neg C_\theta \mid \beta \mid T \mid F$$

  β here is a time interval in the form of $[t_s, t_d]$, T means always and F means never.

Obviously if any constraint in quadruple C be False (F), then C is False (the related label is $F : P$) and means "never". if all constraints of C are True (T), then C equals True which means "unconditionally" or "without any constraint". So, the label of related edge will be $T : P$ or will simply be P.

Consider the Figure 1 which shows applying an example constraint C on accessing a library applicant (i.e. s) to a library document (i.e. o):

This informally says that "Any library applicant who is guest or is working as intern (C_s) for accessing

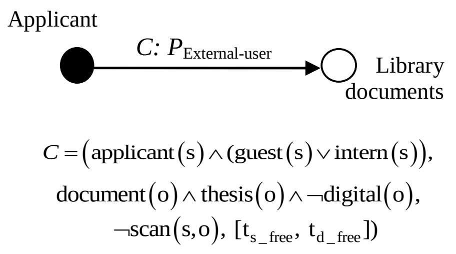

Applicant
C: $P_{\text{External-user}}$
Library documents
$C = (\text{applicant}(s) \wedge (\text{guest}(s) \vee \text{intern}(s)),$
$\text{document}(o) \wedge \text{thesis}(o) \wedge \neg\text{digital}(o),$
$\neg\text{scan}(s,o), [t_{s\_free}, t_{d\_free}])$

Figure 1. An example constraint.

to non digital thesis documents (C_o) in free times of the library (C_θ) should obey the policy of external users ($P_{External-user}$) condition to never scanning the documents (C_α)". Note that $P_{External-user}$ is another CPG and according to the CPG expansion procedure (see Definition 3), the above policy could be expanded.

In Definition 2, P, in its basic form, is the set of basic rights defined in the system. But we have extended it to be a policy again because: Sometimes instead of possessing a right toward an object by a subject (i.e. $P \subseteq R$), the subject may follow a policy for accessing the object. So, P is another CPG (another policy). For example, a college should follow the public policy of library for accessing the library documents. Therefore, we should provide the possibility of nested policy definition for modeling these situations.

Assume that P and Q are two CPGs and the source vertex v_a in policy Q has the policy P_D toward the destination vertex v_b in the condition C. Thus, va corresponds to one of the present vertices in CPG P (we call this vertex D). Therefore, all policies and rights that D has over other entities are also applied on va. Also, if vertex va itself contains other entities, they will follow the new changes recursively (See Example 3). In fact, we encounter with label $C : P_D$ for the connecting edge of two vertices v_a and v_b.

To change a graph with such an edge into a basic graph (i.e. a graph with P from the set R) and studying how to transfer rights and information, we define the policy expansion procedure as follows:

Definition 3. Policy Expansion: If $Q = (V_1, L_1)$ be a CPG in which l_1=(v_a, v_b, C:P_D) $\in L_1$ and $P = (V_2, L_2)$ be another CPG where $l_2 = (D, v_b, \text{C}'{:}P'_X) \in L_2$ and assume that $L'_2 \subseteq L_2$ be the set of all edges outgoing from the vertex D, then $K = (V, L)$ is the expanded CPG of Q regarding P:

$V = V1 \cup V2$,
$L = (L_1 - \{l_1\}) \cup (L_2 - L'_2) \cup \{l_i \mid l_i = (D, v_i, C \wedge C'_i{:}P'_{iX}), (D, v_i, C'_i{:}P'_{iX}) \in L_2, i = 1, 2, \dots\}$,
$V_a = D$

The later expression means that vertices D and v_a are merged together. This procedure will be continued (say in n steps) until all the nested constraint policy

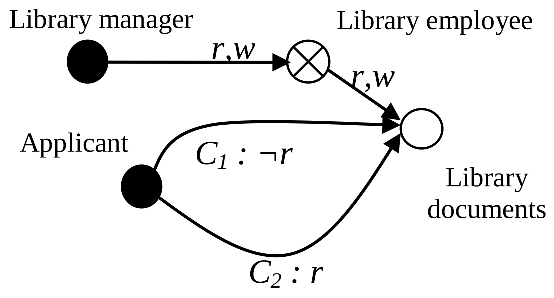


Figure 2. The policy of library.

graphs are expanded to the basic rights (R). In this state the final edge between v_a and v_b will be $l = (v_a, v_b, C \wedge C' \wedge \dots C_{(n)} : R)$.

Definition 4. Combining Constraints: if $C_1 = (C_{1s}, C_{1o}, C_{1\alpha}, C_{1\Theta})$ and $C_2 = (C_{2s}, C_{2o}, C_{2\alpha}, C_{2\Theta})$, then:

$C_1 \wedge C_2 = (C_{1s} \wedge C_{2s}, C_{1o} \wedge C_{2o}, C_{1\alpha} \wedge C_{2\alpha}, C_{1\Theta} \wedge C_{2\Theta})$,
$C_1 \vee C_2 = (C_{1s} \vee C_{2s}, C_{1o} \vee C_{2o}, C_{1\alpha} \vee C_{2\alpha}, C_{1\Theta} \vee C_{2\Theta})$,
$\neg C_1 = (\neg C_{1s}, \neg C_{1o}, \neg C_{1\alpha}, \neg C_{1\Theta})$,

In which:

$C_{1\Theta} \wedge C_{2\Theta} = [t_{1s}, t_{1d}] \wedge [t_{2s}, t_{2d}]$ ($[t_{1s}, t_{1d}] \wedge [t_{2s}, t_{2d}]$ is evaluated to true in t if $t_{1s} < t < t_{1d}$ and $t_{2s} < t < t_{2d}$),
$C_{1\Theta} \vee C_{2\Theta} = [t_{1s}, t_{1d}] \vee [t_{2s}, t_{2d}]$ ($[t_{1s}, t_{1d}] \vee [t_{2s}, t_{2d}]$ is evaluated to true in t if $t_{1s} < t < t_{1d}$ and $t_{2s} < t < t_{2d}$),
$\neg C_{1\Theta} = \neg[t_s, t_d]$ ($\neg[t_s, t_d]$ is evaluated to true in t if t is not in the range t_s and t_d).

Example 1. A college must follow the policy of university library in accessing to library documents, and the policy of the library contains the following rules:

- If the applicant is a guest of library, he/she is not allowed to read the documents in the official times i.e. $t_{s\_\text{official}} \leqslant t \leqslant t_{d\_\text{official}}$. So the related condition is:
  $C_1 = (applicant(s) \wedge guest(s), document(o)$, T, $[t_{s\_\text{official}}, t_{d\_\text{official}}])$.
- If the applicant has a valid recommendation (e.g. from the mayor), he/she can read documents, so $C_2 = (applicant(s) \wedge recommended(s), document(o), T, T)$.

Figure 2 depicts the whole library policy by using CPG. We see that an applicant depending on his/her conditions/roles has different rights over documents.

Now, assume the policy of the university college is partially as follows:

- The college obeys the library policy in autumn and winter, i.e. $C'_1 = (T, document(o)\ , T, [t_{s\_aw},$

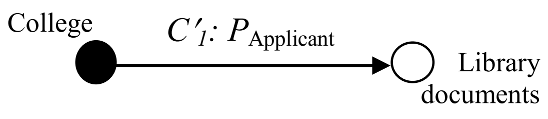


Figure 3. The policy of College.

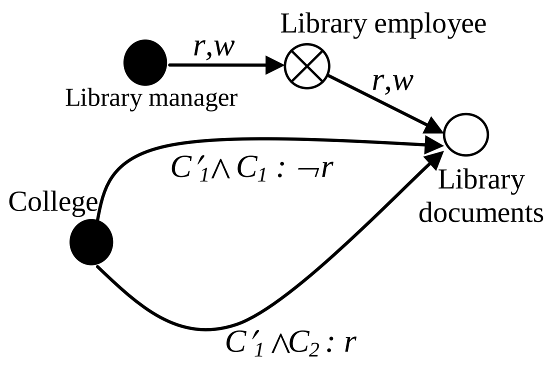


Figure 4. The college policy after expansion.

t_{d_aw}]). Observe Figure 3.

Because the college has the role as an applicant, it must be mapped to applicant vertex of library policy (Figure 3). Now, we use Policy Expansion procedure for expanding the college policy over library documents to reach the basic CPG (Figure 4). For instance, the result of $C_1' \wedge C_1$ is as follows:
$C_1' \wedge C_1 = (applicant(s) \wedge guest(s), document(o), T, ([t_{s\_\text{official}}, t_{d\_\text{official}}] \wedge [t_{s\_aw}, t_{d\_aw}]))$

At the same time, it is possible that the college itself contain other entities which are connected to each other according to their related policies. Assume that the college has manager, students, and teachers so its policy is a combination of these partial policies.

2.2 CPG transfer rules

Since system entities have various rights toward each other and because of their relations; it is possible to appear new rights in the system (there may occur some information flows in that system). We introduce some transfer rules in our model. As in TG, applying these rules on system's ACP graph may result in new edges from two existing edges. Since our model considers constraints, we complete TG transferring rules according to our goals. For example, we show this method for some rules as follow. Note that the transfer occurs when two constraint sets of related existing edges contributed to the emergence of the new edge, occur simultaneously. Also, each parameter of constraints sets is mapped to related vertex during the new transfer (i.e. each C_s parameter is mapped to related subject and each C_o parameter is mapped to related object and each C_α parameter is mapped to related subject and object).

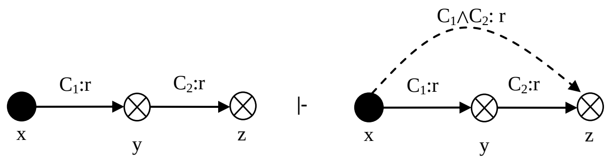


Figure 5. The spy rule.

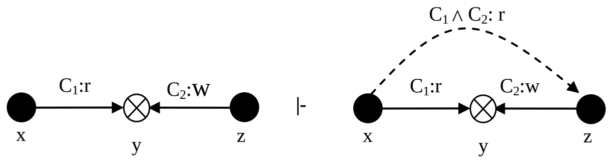


Figure 6. The Post rule.

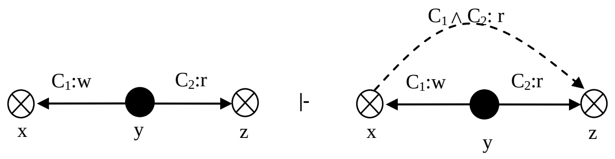


Figure 7. The Pass rule.

- *Spy*: Let x, y and z be three distinct vertices in a CPG_0, and let x and y be subjects. Let there be an edge from x to y labeled $C_1 : r$ and an edge from y to z labeled $C_2 : r$. Then the *Spy* rule defines a new CPG_1 graph with an implicit edge from x to z labeled $C_1 \wedge C_2 : r$. Observe Figure 5.
- Post: Let x, y, and z be three distinct vertices in CPG_0 and let x and z be the subjects and let there be an edge from x to y labeled $C_1 : \alpha$ with $r \in \alpha$ and an edge from z to y labeled $C_2 : \beta$ where $w \in \beta$; in this case the post rule changes CPG_0 graph to CPG_1 graph by the addition of a new implicit edge from x to z labeled $C_1 \wedge C_2 : r$. Observe Figure 6.
- Pass: Let x, y, and z be three distinct vertices in CPG_0 and let y be subject and let there be an edge from y to x labeled C_1:α with $w \in \alpha$ and an edge from y to z labeled C_2: β where $r \in \beta$; in this case the pass rule changes CPG_0 graph to CPG_1 by the addition of a new implicit edge from x to z labeled $C_1 \wedge C_2 : r$. Observe Figure 7.
- Find: Let x, y, and z be three distinct vertices in CPG_0 and let z be subject and let there be an edge from z to y labeled C_1: α with $w \in \alpha$ and an edge from y to x labeled C_2: β where $w \in \beta$; in this case the *find* rule changes CPG_0 graph to CPG_1 graph by the addition of a new implicit edge from x to z labeled $C_1 \wedge C_2 : r$. Figure 8 shows the *find* rule.

Later in the paper samples of these transfers will be studied.

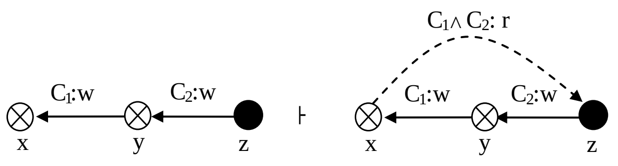


Figure 8. The Find rule.

2.3 Composition of ACPs

Although different organizations protect their resources, they need to be communicated together to reach their common and related goals. For example, nowadays composition of web services is one of the key requirements to meet the demands of users whose needs are not met by existing web services [13]. Moreover, verifying that whether the composed web service supports the ACPs of participant web services or not, is a challenge. Few studies such as ([14], [15]) have ever tried to represent a solution to this problem. The following grammar expresses kinds of our proposed policy combinations:

$$K = P \cup Q \mid P \cap Q \mid P - Q \mid \neg P$$

where K, P and Q are CPGs.

Now, we explain the way of obtaining these combinations:

2.3.1 Union ($P \cup Q$):

By means of this, two policies are merged into a policy which contains a union of both policies. This operator admits the access which one or both of its components permit.

Definition 5. Union ($P \cup Q$): If $P = (V_1, L_1)$ and Q=(V_2, L_2) are two CPGs, then $K = (V, L)$ is the union of these two graphs, in which $V = V_1 \cup V_2$ and $L = L_1 \cup L_2$.

Note that if a vertex is a subject in one graph and is an object in another one, then it plays the both roles (i.e. $\otimes$) in the graph of union.

Sometimes after union of two CPGs, we can simplify the combined graph. The following definitions shows way that how to simplify the graph.

Definition 6. Constraints Equivalence: Two constraints $C_1 = (C_{1s}, C_{1o}, C_{1\alpha}, C_{1\Theta})$ and $C_2 = (C_{2s}, C_{2o}, C_{2\alpha}, C_{2\Theta})$ are equivalent ($C_1 \equiv C_2$) if $C_{1s} \longleftrightarrow C_{2s} \wedge C_{1o} \longleftrightarrow C_{2o} \wedge C_{1\alpha} \longleftrightarrow C_{2\alpha} \wedge C_{1\Theta} \longleftrightarrow C_{2\Theta}$.

Definition 7. CPG simplification: If $P = (V, L)$ be a CPG in which $l_1 = (v_a, v_b, C_a : P_1) \in L, l_2 = (v_a, v_b, C_b : P_2) \in L$, and $C_a \equiv C_b$, then we have $l = (v_a, v_b, C : P_1 \cup P_2) \in L$ where:

- If $P_1, P_2 \in R$, then $P_1 \cup P_2$ is the union of these two sets.
- If P_1, P_2 are two CPGs, their union is produced according to Definition 5 and then the policy expansion procedure could be applied on the produced graph to reach the basic CPG. Note that we can, at first, expand the nested graphs by applying the policy expansion procedure and then get the union of both produced graphs as Definition 5.
- If either P_1 orP_2 is a member of R and another one is a CPG, we begin by expanding the nested graph and then will try to get the union of two graphs like we mentioned in Definition 5.

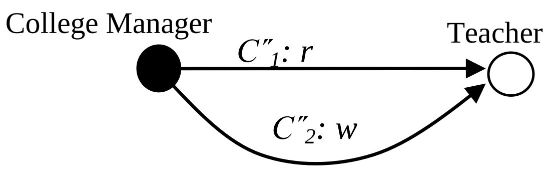


Figure 9. The policy of the College Manager.

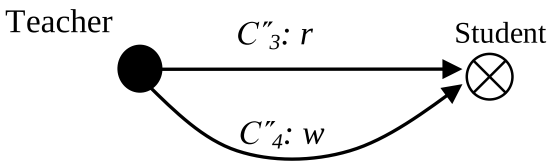


Figure 10. The teacher policy.

Example 2. Consider the college in the Example 1 with manager of the college, as well as two teacher agent and student agent acting as intelligent agents of teachers and students of the college in the college management system. Suppose that each one has its own policy. For example consider the policy of the college manager as follows:

- College manager is allowed to read the teacher agent to verify its code: C_1= ($manager(s)$, $teacher(o)$, $verify(s, o)$, T).
- He/she can change the code of teacher agent at the time of selecting the (real) teachers for instructing lessons: C_2= ($manager(s)$, $teacher(o)$, T, $[t_{s\_selection}, t_{d\_selection}]$).

Figure 9 shows the policy of the college manager.

On the other hand, the policy of the teacher agent is as follows:

- Teacher agent is allowed to access the student agent at the exams season: C''_3= ($teacher(s)$, $student(o)$, T, $[t_{s\_exam}, t_{d\_exam}]$).
- Teacher agent can modify the student agent configuration at the time of marks announcement: C''_4 = ($teacher(s)$, $student(o)$, T, $[t_{s\_announce}, t_{d\_announce}]$).

The teacher policy is shown in Figure 10.

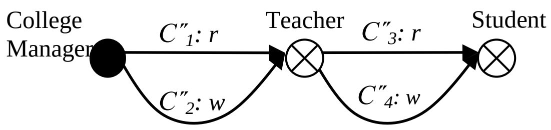


Figure 11. The college policy.

Now, we obtain the college policy based on its characters policies. It is gained from the union of their policies (Figure 11). Note that the teacher vertex changes from object state into subject-object state.

Example 3. As we saw, Figure 4 was obtained from applying the policy expansion procedure on the college policy. On the other hand, the college contains the mentioned characters. So, they must obey the college's policy toward the library documents in order to access them. Figure 12 shows the college's policy toward library documents considering its characters. We only show the teacher's policy toward documents for the sake of figure clarity.

Similarly, the university as a subject must follow the policy of its upper organization e.g. Department of Research in its educational rules. Furthermore, the university consists of many entities, with their own policies, that may be interrelated. By using the mentioned method, the university policy can be obtained on the basis of its components policies and relations. Our model can express the groups of users who follow a common policy. It is possible to apply a policy for all group members or exclude some members of this group with some constraints (i.e. if these members obey the constraints, they have the same policy as other members of that group).

2.3.2 Intersection ($P \cap Q$):

It allows only those accesses that are permitted by both components P and Q. For example, suppose that the library employee allows the college students to access to the library documents. On the other hand, the manager of college permits specific persons to access to these documents. In this case, ACP of the college' s students to the mentioned documents is obtained from the intersection of both policies (i.e. only the accesses are valid which both manager and library employee permit). The intersection CPG is obtained as follows:

- If nested policy exists in P or Q, we must apply the policy expansion procedure to reach the basic CPG for each one (P or Q).
- Then, we apply CPG transferring rules to add the implicit transfers to each CPG, if exist. Note that without considering these indirect transfers, the combination results and analysis outcomes may be incomplete or even incorrect.
- Finally, we intersect two resulted CPGs as the following definition.

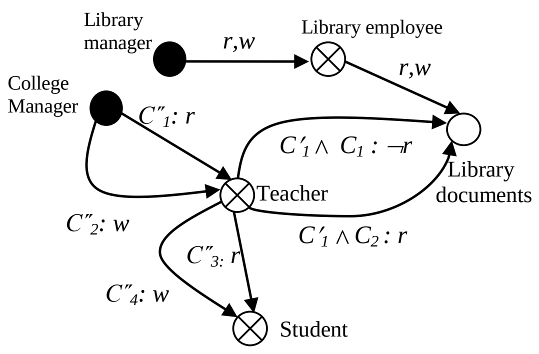


Figure 12. College's policy toward library documents (considering its principals).

Definition 8. Intersection ($P \cap Q$): If $P = (V_1, L_1)$ and $Q = (V_2, L_2)$ are two CPGs, then $K = (V, L)$ is the intersection of these two graphs in which $V = V_1 \cap V_2$ and for each two edges $l_1 = (v_a, v_b, C_1 : P_1) \in L_1$ and $l_2 = (v_a, v_b, C_2 : P_2) \in L_2$, the edge $l \in L$ is the corresponding edge of v_a and v_b in K that $l = (v_a, v_b, C_1 \wedge C_2 : P_1 \cap P_2)$, where:

- P_1, $P_2 \in$ R then $P_1 \cap P_2$ is the intersection of P_1 and P_2. If the resulted set of $P_1 \cap P_2$ is empty, the related edge is omitted.

2.3.3 Subtraction ($P - Q$) :

It restricts policy P by eliminating all the accesses in the second policy (Q). The subtraction graph is obtained as follows:

- If the nested policy exists in P or Q, we apply the policy expansion procedure to reach the basic CPG for each one (P or Q).
- Then, we apply transferring rules on each CPG to obtain implicit transfers, if exists.
- we subtract two resulted CPGs as the following definition.

Definition 9. Subtraction ($P - Q$) : If P=(V_1, L_1) and Q=(V_2, L_2) are two CPGs and if $L_1^{'} \subseteq L_1$ is a set of edges in P, which have the same corresponding beginning (subject) and end (object) vertices in both P and Q (i.e. every l_1=(v_a, v_b, C_1:P_1) $\in L_1$ in P which has an equivalent edge l_2=(v_a, v_b, C_2:P_2) $\in L_2$ in Q, is placed in $L_1^{'}$ set), then $K = (V, L)$ is the subtraction of these two graphs that:

$V = V_1$, and

$L = (L_1 - L_1') \cup \{l_i \mid$

$$l_i = \begin{cases} (v_a, v_b, C_{1i} \wedge C_{2i} : P_1 i - P_{2i}) & \text{if } C_{1i} \equiv C_{2i} \\ (v_a, v_b, C_{1i} : P_{1i}) & \text{otherwise} \end{cases}$$

$\}$

Where: $P_1, P_2 \in R$, so $P_1 - P_2$ is the subtraction of

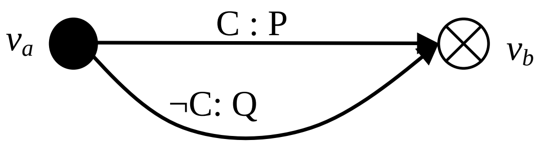


Figure 13. Conditional relations CPG model.

them.

2.3.4 Negation ($\neg Q$):

A CPG expresses the rights of its subjects toward related objects based on constraints. Negation of the CPG expresses possessing negative of mentioned rights for related subject toward the object in the same constraints, meaning that the subject would not have the specified rights in the policy in mentioned situation towards that object. Thus:

Definition 10. Negation ($\neg Q$): If $Q = (V, L)$ be a CPG and $L = \{l_i \mid l_i = (v_{ai}, v_{bi}, C_i : P_i)\}$, then $K = (V', L')$ is the negation of Q in which: $V' = V$, $L' = \{l'_i \mid l'_i = (v_{ai}, v_{bi}, C_i : \neg P_i)\}$ where:

- If $P \in R$, $\neg P$ is the negation of this set.
- If P is a CPG, then $\neg P$ will be obtained recursively by using Definition 10.

2.3.5 Conditional relations:

In addition to the mentioned operators, we can model conditional relations by using CPGs as follows:

The subject v_a, in condition C, possesses the policy P toward other entity (v_b) and for the other conditions (except C), it has the policy Q toward v_b. In fact, we encounter conditional expression: "if C then P else Q" for v_a toward v_b. It can be modeled by using a CPG, as Figure 13.

2.4 Conflict detection

During combination of CPGs or verification of them, it is possible to encounter a case in which a subject simultaneously has two inconsistent policies or rights toward the same object.

Definition 11. Conflict property: If $Q = (V, L)$ be a CPG in which $v_a, v_b \in V$ and $l_1 = (v_a, v_b, C_a : P) \in L$ and $l_2 = (v_a, v_b, C_b : \neg P) \in L$, and $C_a \equiv C_b$, then there is a conflict in CPG Q.

The conflict detection procedure of CPG Q is as follows:

- If nested policy exists in Q, apply the policy expansion procedure to reach the basic CPG (according to Definition 3).
- Then, apply CPG transfer rules to add the implicit transfers to the CPG, if exists.

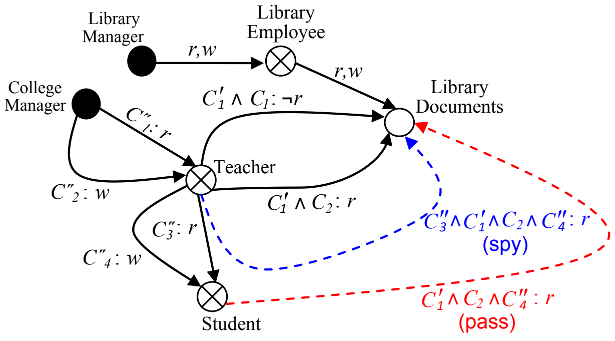


Figure 14. College policy toward library documents (after applying Pass and Spy rule).

- Finally, check the conflict property in the resulted CPG (according to Definition 11).

Example 4. Consider the previous example in which one part of college policy toward the library documents was expanded. The expanded CPG is a basic one, i.e. there are no unexpanded policies. There is just one explicit conflict candidate i.e. there are two read access rights having opposite sign from the Teacher to the Library documents. According to the definition of conflict property, it can be a conflict only if their corresponding constraints are equal i.e. we should check if $C'1 \wedge C1 \equiv C'1 \wedge C2$. It is equal to proving the following theorem:

$(guest(s) \longleftrightarrow recommended(s)) \wedge ([t_{s\_aw}, t_{d\_aw}] \wedge [t_{s\_crowd}, t_{d\_crowd}] \longleftrightarrow [t_{s\_aw}, t_{d\_aw}])$

Intuitively we should know if we could deal with recommended persons same as guests? Also are the whole autumn and winter as assumed to be crowded periods of library? If the answers of both questions are yes, then we have a conflict in the college access control policy.

Also we should apply transfer rules to achieve more candidates for conflict. Here we only consider a single possible instance. Figure 14 depicts the results of applying Pass and Spy transfer rules on CPG of Example 3 that results in two implicit constrained read rights which one of them is candidate for conflict with another existent explicit negative read right.

To detect a conflict we should check if $C''_3 \wedge C'_1 \wedge C_3 \wedge C''_4 \equiv C'_1 \wedge C_3 \wedge C''_4$ or not. It is equal to proving the following theorem:

$(recommended(s) \longleftrightarrow guest(s)) \wedge ([t_{s\_exam}, t_{d\_exam}] \wedge [t_{s\_aw}, t_{d\_aw}] \wedge [t_{s\_announce}, t_{d\_aannounce}] \longleftrightarrow [t_{s\_aw}, t_{d\_aw}] \wedge [t_{s\_crowd}, t_{d\_crowd}])$

Intuitively beside the previous questions, here we should know if the periods of exams and remark announcements are considered as crowded periods or not?

Note that if we think about automating the conflict detection process, besides having a knowledge base, we need to have a theorem prover for $\mathcal{L}^2_{\approx}$ language (two-variable-fragment of first order logic) which fortunately there is decidable resolution based proof procedures for it [31]. Furthermore the algorithm used to find the candidate conflicts is a graph traverse algorithm of order N^2 where N is the number of vertices in the fully expanded CPG (i.e. the basic CPG resulted from expanding all nested policies and applying transfer rules).

2.5 More analysis of ACPs

In this section, we define a method for more accurate analysis of the ACPs according to CPG model features. By using this model not only we are able to specify the ACPs, but also we can examine rights and information transferring in related systems.

In many cases, the system works according to integration of system parts, i.e. it works based on a combined process obtained from combination of system parts processes. While the resulted process of the combination seems faultless, after obtaining the policy of combined process and analyzing it, we may encounter some inconsistencies. For example, the resulted policy of combination has contradictions with the policy of each participated part. Nevertheless the policy of one part is in contradiction with some others, e.g. this part is used in the combination process by mistake (e.g. in web service composition as mentioned before).

In general, analysis of a model is arising from responses given to questions which are posed in mind. What is primarily important in an access control model is how to transfer the rights and information in a related system. More accurately, we are interested in answering the following questions:

- Based on satisfying which condition (conditions), does the subject x obtain the right α toward the object y?
- Can subject x obtain the right α based on occurring condition C toward the object y?

Since CPG considers the constraints in modeling the ACPs, it is possible to consider and analyze constrained transfers accurately. The answer to the first question is very useful in many cases (e.g. the constraint resulted from this question can be analyzed and adjusted more accurately). The second question distinguishes whether a transfer occurs based on distinct conditions or not.

The TG model has defined predicates for analyzing the graphs ([9], [10], [11]). We mention the "can.know predicate as an example:

- Can.know(x, y, G_0): The predicate can.know (x,y,G_0) is true if and only if there exists a sequence of protection graphs $G_0,\ldots, G_n$ such that G_n is derived from G_0 by rule applications, and in G_n there is an edge from x to y labeled r or an edge from y to x labeled w, and if the edge is explicit, its source is a subject.

Since TG predicates are defined unconditionally, they can be used for analyzing the CPGs in always-true conditions. But for our interests, we define the following predicate based on the proposed model.

Definition 12. Can.Obtain property: The predicate can.obtain(C, α, x, y, CPG_0) is true for the condition C, the access right α, and vertices x (as a subject) and y (as an object or a subject), if and only if there exist the graphs $CPG_0, CPG_1, \ldots, CPG_n$, so that $CPG_0 \vdash^* CPG_n$ by applying transferring rules, and there is an edge with label $C : \alpha$ from x to y in CPG_n.

If we assume the set of all desired rules as A, transferring from CPG_0 to CPG_n, is resulted from applying the whole applicable rules on $CPG_i (0 \leqslant i \leqslant n-1)$, beginning from CPG_0, with this condition that after obtaining CPG_n, no rules of A are applicable on CPG_n. This process can be ended when desired result is found (when the desired edge with label of α appears from x to y) otherwise it will continue until it reaches CPG_n. Also the process is decidable. If we consider the size of CPG_n (which is indeed the most expanded graph in the sequence) as N (i.e. the number of its nodes), then the order of operations is of $n.|A|.N$ in which $|A|$ is the cardinality of set A.

Example 5. Remember the graph of Figure 11 from Example 2. Suppose we want to know whether the manager is able to obtain the right r toward the student or not (If yes, in which constraint). In other words, is the result of predicate can.obtain $(C, r, manager, student, CPG_0)$ true? In which constraint (C)?

After applying the Spy rule, Figure 15 is obtained. Two edges $l_1 = (manager, teacher, C''_1 : r)$ and $l_2 = (teacher, student, C''_3 : r)$ contribute to create the implicit edge. As mentioned before, the parameters of constraints C''_1 and C''_3 are mapped to related vertices and their conjunction results in the constraint of new edge. So, for the new edge $l = (manager, student, C''_5 : r)$, the constraint C''_5 is obtained as follows:

$$C''_5 = C''_1 \wedge C''_3 = (T, T, verify(manager, teacher), T) \wedge (T, T, T, [t_{s\_exam}, t_{d\_exam}]) = (T, T, verify(manager, teacher), [t_{s\_exam}, t_{d\_exam}])$$

Also, after applying the *find* rule, the edge $l' = (student, manager, C''_6 : r)$ is created and C''_6 is: $C''_6 = C''_2 \wedge C''_4 = (T, T, T, ([t_{s\_selection}, t_{d\_selection}] \wedge [t_{s\_announce}, t_{d\_announce}]))$ As we saw, information

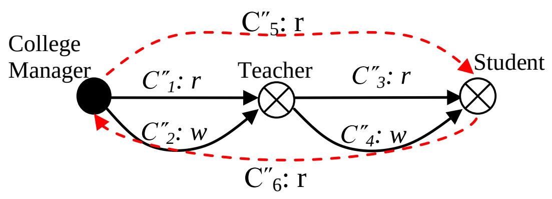


Figure 15. Applying spy and find rules on college graph.

transfer has two different directions depending on system constraints (the constraints which are held in the system). If condition C_5'' holds, the manager of college has the right of reading the student's information in the combined graph, and with holding condition C_6'', the student reads the manager information. These transfers occur after combining the manager policy and the teacher policy for obtaining the college policy.

3 CPG for Modeling ACP's in Composition of Web Services

Service composition is a fundamental technique for developing web service-based applications. In general, most of the time a single service cannot achieve the user's goals, while several services coming from different providers can be composed to satisfy this objective.

The features of a combined web service are divisible into two functional and non-functional features. The functional features express the work process and the manner of the related system interactions, while the non-functional features are quality features which are subject to changes as time goes by and the interests of the related organizations are changed. Of the non-functional features, one can point to access control security feature.

Analyzing the non-functional properties of composition ensuring from correctness of composed web service is still a challenge [13]. Despite lots of web service composition methods, such as BPML [16], BPEL [7], WSCI [17] and etc, these methods only describe the functional properties of composition [18]. An important problem in web service composition is that partners which contribute in a combination may have inconsistent policies. So, composed web service ACP must be modeled and analyzed ensuring from the correctness of composition.

On the other hand, because of the relation of the entities participating in the composition with different rights toward each other, the study of the transfer of the implicit rights and information in compositional systems are one of the necessities required for the confidence in an apt definition and the compatibility of

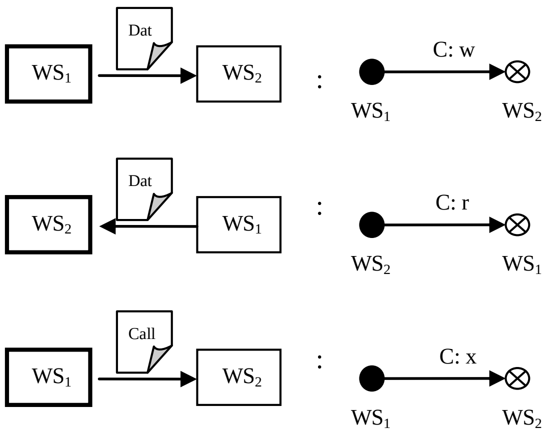


Figure 16. CPG model for different kind of data transfer in the web service composition.

the policies of the composition's partners. In other words, we need a method for modeling the web service ACPs and obtaining the policy resulting from their composition (with regard to the presence of the problems of the composition methods in the absence of the expression of non-functional features), which not only models the rights of the entities in relation to each other, but also takes the information transfers into consideration. We take advantage of CPG model for this purpose and will express the access control policies in the web service composition in next section.

3.1 Web service composition ACPs

As we know, web services are related together from sending or receiving the information or generally from invoking (executing) each other. According to each case, the information flow is different. We define the new right **x**, in addition to **r** and **w** rights, for modeling the web service composition ACPs:

Definition 13. X (execute): It is the right of executing an object for a subject. This right is changed to one of **r** or **w** rights, as follow:

For example, when a web service WS_1 invokes another one (WS_2), it has the right of executing the WS_2. During this operation WS_1 may send some information to WS_2 or vice versa. When the information is sent, WS_1 has the **w** right toward WS_2 and from receiving data, WS1 has the **r** right toward that (WS_1 has both **r** and **w** rights over WS_2 for both send and receive states). Figure 16 shows each type of these interactions and the related CPG model.

3.1.1 Constraints in Extraction of Composition Model

The constraint C depicted in Figure 16 signifies the conditions in which a web service has a specific r/w access right to another web service. In unconditional state, this constraint has a value of T (True) which we don't mention as before to keep simplicity. These constraints are different in the area of web services according to their different applications. The current constraints in web service composition methods concerning all choreography or orchestration by using the constituents of a foursome C compound are taken into consideration and explained as follows:

- C_s and C_o: generally, if there is a need for the source web service to enjoy special features to have a right to the target vertex, then these conditions are expressed by C_s constraint. In the same way, the constraint governing the target web service is also expressed by C_o.

  Depending on the composition goal, the composition of web services comprises a set of web services communicating with each other. Each web service can have different roles. It can appear in different roles in one composition or even it can belong to several compositions at the same time. The roles are some privileges which are required for specific activities in the process of the composition of web services.

  By using a CPG model, the role of each service (whether compositional or partner web service) is expressed by the use of two constraints C_s for source web service and C_o for target web service.
- C_Θ: is used to express temporal constraints on accesses if necessary.
- C_α: expresses a service (a method) which is to be invoked and used from the service sets presented by the target web service. This is principally named "operation" in the domain of web services.

Any web service may present its services by different policies to others. As an example, for the invocation of "A" service, one requester has the right to read the web service information, but for the invocation of "B" service of the same web service, it does not have such a right to that web service. Therefore, each request to a specific method of a web service is modeled in the form of an operation and represents a constraint on the destination vertex for which a requester has a specific right to the mentioned web service. So, in CPG model, the name of the requested operations is modeled by C_α.

For the purpose of obtaining the final CPG of composition, we must follow some steps in the following order:

(1) Model the CPG of each participant web service policy in composition over its related entities (e.g. resources or other interrelated entities) as mentioned in Section 2.
(2) Model the policy of composed web service according to its relation with each partner as mentioned in this section.
(3) Obtain the union of resulted CPGs from two steps 1 and 2, as mentioned in subsubsection 2.3.1. Note that, Web service composition is a kind of cooperation. If we don't use the union operation, then we could not have any cooperation. To clarify the issue, let us have an example: Assume that in web service composition of a travel agency, a web service is used for flight reservation which has access to flights DB. Also there is another web service for hotel reservation which has access to hotel information DB. The web service composition can be established only when access to both DBs be provided. Whereas the intersection of these access rights might be null.

The final graph of composition is obtained from these three steps. Now, it can be verified and analyzed for contradictions or conflicts as mentioned in Section 2.4 and Section 2.5.

As a sample of the application of the presented method in the application of CPG model for modeling the ACPs in the composition of the web services, we proceed to modeling of the ACPs in BPEL processes.

4 Modeling ACP's in BPEL processes

4.1 Extraction of CPG model resulting from BPEL process

BPEL models a business process which includes a composition of activities implemented on the basis of the defined control flow.
The most important sections comprising a BPEL process can be divided into three groups:

(1) Business process partners
(2) The activities explain the business process logic using interactions between process and its partners.
(3) A collection of variables maintaining the state of the process.

The conceptual core of BPEL is the messages exchanged between the composed process and the web services entering into its partnership (partner web services). The general scenario in BPEL is usually initiated by receiving one message by the composition process and then by calling a set of external web services so that the compositional operations are performed and the final response is ultimately announced to the requester.

In access control, the activities which involve BPEL process with outside environment are taken into consideration; since the transfer of rights and information

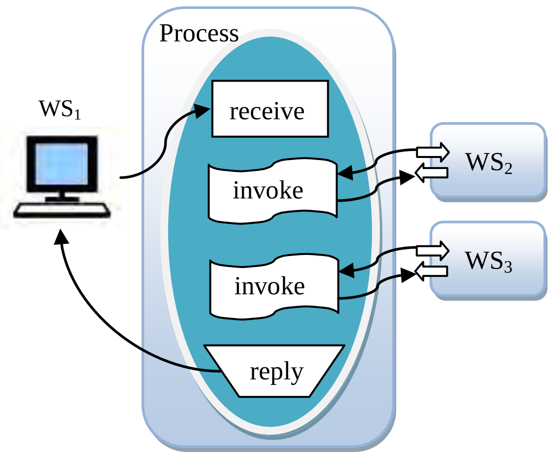


Figure 17. BPEL process.

is possible in this way. These are the activities of invocation of web services (invoke), the receipt of the requests (receive), and their reply are communication bridges for the compositional process with outside entities. Figure 17 depicts a simple design of BPEL process consisting of the three mentioned activities.

Each activity in BPEL process is represented by a label which has the features among which some are compulsory and some are voluntary used according to the needs. These features have their own values for different partners, depending on the process need.

The BPEL process specifies the desired related partner in every activity by features such as “partnerLink”, “partner”. Also this process is connected to the desired service (specified by “operation” and “portType” parameters) by matching with the related existing information in WSDL document of each partner. By the “partner” parameter, the partner web service is specified and by “partnerLink”, one link with specific name between the two partners is generated [7]. Now, the method of obtaining CPG graph resulting from the mentioned activities in BPEL is expressed.

- Invoke: by this activity, the composed web service invokes an operation from a partner web service in a composed process. In this manner, the composed web service has the right to execute the service related to the partner for this operation in CPG model.

  This operation is of two kinds: one-way and two-way (request-response) and is in conformity with the related operations in WSDL document of the partner web service. The general format of this instruction is as Figure 18.

  The “partner” section specifies which web service is the intended partner in this activity. It is known that the operational pattern of WSDL documents is divided into two forms: one-way and two-way.

  In the same way, the operation invoked by “invoke” activity is modeled according to the direction of data exchanges. If the desired information is sent to the partner web service, it is equal to compositional process writing right (w) toward the partner web service, and if the information is received from the partner, it is equal to compositional process reading right (r) toward the partner web service. In case of both sending and receiving, the compositional process has both r and w rights toward the partner (i.e. x right).

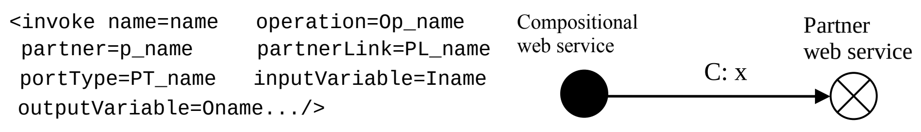
  Depending on the need, two variables are used for sending and receiving information in this activity: the “inputVariable” and “outputVariable” for sending and receiving information, respectively.
- Reply: the compositional web service sends the result of the request (request for executing the operation which is already accepted by receive activity) to the requester. This is equal to the reading right (r) of the requester toward compositional web service. Note that the requester is also one of the partner web services in the composition. This activity format is as Figure 19.
- Receive: This activity specifies from which partner it is waiting to receive the request. By the use of “partnerLink” and “partner” parameters, the partner intended process (the requester) is specified and by the use of “portType” and “operation”, it specifies which operation of the compositional process can be invoked by this requester. In this way, doing the requested operation by the partner and sending the result to it is guaranteed by reply instruction. The general format of this instruction is as follows:

```
<receive name=name     operation=OP_name
   partner=p_name      partnerLink=PL_name
   portType=PT_name    variable=userreq ... />
```

In this activity, if “*OP_name*” operation needs to receive information from the requester, the “userreq” variable is used for this purpose.

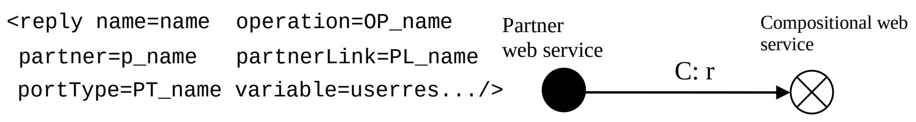
The composition of receive activity and one reply activity, results in one request-response operation in “portType” related to WSDL document of composed process (Figure 19 clearly states this matter).

As you can see, because of the presence of the reply, this set encompasses the sending of data to the requester. But the receipt of the data by the process depends on the type of the request. It is possible that the specified operation in receive activity requires the sending of the data from the requester to the process. It is equal to partner writing right (w) toward the compositional process and these pieces of information are transferred by “userreq” variable. If only the in-



```
<invoke name=name   operation=Op_name
 partner=p_name     partnerLink=PL_name
 portType=PT_name   inputVariable=Iname
 outputVariable=Oname.../>
```

Figure 18. CPG model of the invoke activity.



```
<reply name=name   operation=OP_name
 partner=p_name    partnerLink=PL_name
 portType=PT_name variable=userres.../>
```

Figure 19. CPG model of the reply activity.

vocation of operations is done without data sending, no right (r or w) is obtained by the activity of receive, and there is no need to use "userreq" variable in the instruction format.

In this way, the basic right set in CPG model of BPEL is $R = \{r, w, x\}$, where r, w, and x are read, write, and execution rights, respectively.

Depending on the needs that the related subject or object should be bound to in each activity, the C restraint in the above three activities are determined. As an example, the determined rights in each of the three above activities for doing specific operation of the object web service are done. That is, for the mentioned operation ($operation = Op\_name$), the expressed right on the web services are true toward each other. The name of this operation is specified by the "operation" parameter in each of invoke, receive, and reply operations, and is applied by C_α constraint on the policy related to that activity.

As an example, in invoke activity, for operation named OP_name, the process has the right of execution (x) of the partner's web service. So, the condition for the relation edge between composed process and the mentioned partner is $C = (T, T, OP\_name(s, o), T)$ and on the whole, the label of this edge will be $(T, T, OP\_name(s, o), T) : x$. where, s and o (e.g. in invoke activity) are compositional web service and partner web service specified by partner parameter of this activity, respectively.

Consider another state: each partner can take part in one or several roles in a process. In BPEL process, these roles are expressed by "partnerRole" and "myRole" parameters and consequently, in WSDL file, for different operations it is specified by "role" parameter. The "myRole" parameter shows the role of BPEL process in the link between that process and the partner and the "partnerRole" parameter specifies the role of partner in the mentioned link.

As an example, a purchaser sends its purchase request to the compositional process. In this link the compositional process plays the role of the seller. But, when the compositional process sends the request to the seller's web services, the process plays the role of the purchaser and the partner's web service plays the role of the seller. Observe the following code part:

```
<receive name="receive"   partnerLink="Buyer"
   operation="request"    variable="request"
   initiate="yes"
</receive>
<invoke name="quote_supplier1"
   partnerLink="Supplier1"
   operation="request_quote"
   inputVariable ="part_request"
   outputVariable="part_quote"
</invoke>
<invoke name="quote_supplier2"
   partnerLink="Supplier2"
   operation="request_quote"
   inputVariable ="part_request"
   outputVariable="part_quote"
</invoke>
...
<!--construct a proposal from the part quotes received -->
<reply name="reply"
   partnerLink="Buyer"
   operation="send_proposal"
   variable="proposal"
</reply>
```

Assume that we want to specify the constraint in a state that the compositional web service invokes one of the seller's web services. The above code shows that in this state, the role of compositional process is myRole="Requestor" and the role of web service of the partner is Role= "purchaser". Therefore, C in the related edge of these two entities is as follows:

$C_s = Requestor(s)$: s is the source of the edge(here, the compositional process)

$C_o = Purchaser(o)$: o is the communication edge destination parameter(here, the partner)

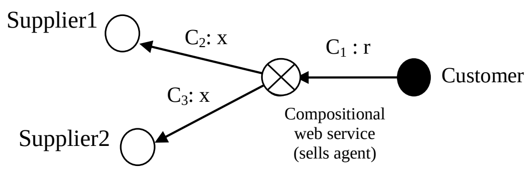


Figure 20. CPG model for sale BPEL process.

4.2 Example

Consider the following scenario: a purchaser needs to buy a computer system (PC) and there are a number of suppliers who produce and present a part of the components of a computer system.

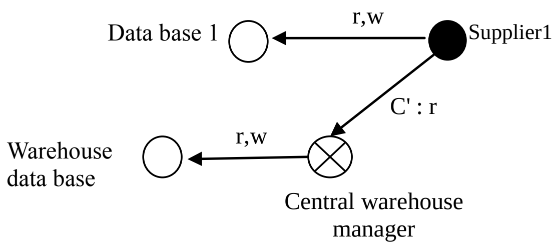
The purchaser sends his request to a sells agent (compositional web service) instead of referring to each all suppliers. The sells agent inquires about the price and other information of the components and prepares them for the purchaser by getting in touch with the suppliers. The following code part shows a simple sample of this scenario based on BPEL process.

```
<receive name="receive"   partnerLink="Buyer"
   operation="request"   variable="request"
   initiate="yes"
</receive>
<invoke name="quote_supplier1"
   partnerLink="Supplier1"
   operation="request_quote"
   inputVariable ="part_request"
   outputVariable="part_quote"
</invoke>
<invoke name="quote_supplier2"
   partnerLink="Supplier2"
   operation="request_quote"
   inputVariable ="part_request"
   outputVariable="part_quote"
</invoke>
...
<!--construct a proposal from the part quotes received -->
< reply name="reply"   partnerLink="Buyer"
   operation="send_proposal"   variable="proposal"
</reply>
```

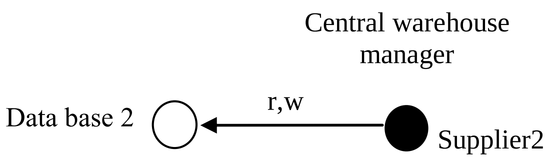
We want to consider the model resulting from the communication between the sells agent with two suppliers of 1 and 2, and sent the result to the purchaser by CPG graph. Figure 20 depicts this model in which C_1, C_2, and C_3 constraints are as follows:

$C_1 = (T, T, send\_proposal(customer, sells\ agent), T)$
$C_2 = (T, T, request\_quote(sells\ agent, Supplier1), T)$
$C_3 = (T, T, request\_quote(sells\ agent, Supplier2), T)$

In this example, in the two operations of "*request_quote*" from two partners, the information of their computer parts is sent from the web services of the partner to the compositional web service. Therefore, the execution right (x) in invoke activity is changed into the reading right (r) of these pieces of information.



(a) policy of supplier 1.



(b) policy of supplier 2.

Figure 21. policy of suppliers.

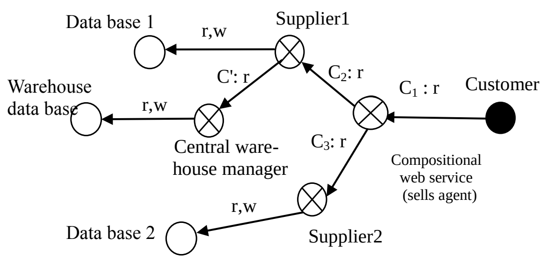


Figure 22. Final graph of sale process considering partners policies.

At the same time, as we know every partner web service has its own access control policy. For example, the policy of suppliers 1 and 2 is assumed to be as 21a and 21b.

Now, the union of the policy of partner web services in the composition and compositional policy are obtained (Figure 22). We have ignored the inclusion of customer's policy as another partner web service in this example to keep the simplicity of the figure.

By obtaining the final graph of the composition, we can analyze the policies. The methods of analysis and conflict verification in CPG graph are presented in Section 2. We suffice to a single analysis here as follows:

Assume that we want to examine the manner of information transfer between the customer and Data base 1. Therefore, we examine to know what the result of the predicate of can.obtain is and in case that the value of this predicate is true, we examine the conditions for its being true. The graph in Figure 22 is assumed to be in CPG_0, then after two using of "spy" rule we reach the new relation edge between the "customer" and the "data base 1" vertices. Observe Figure 23.

In other words, the value of following predicate will

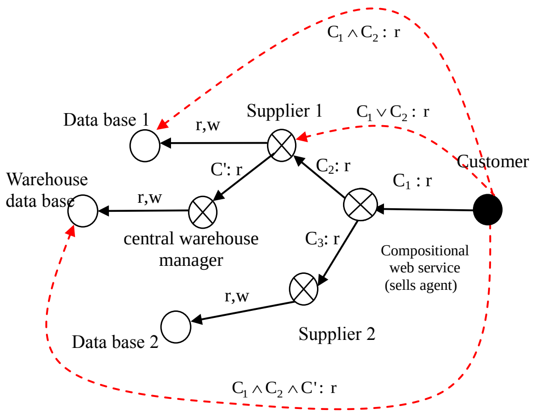


Figure 23. Final graph and applying some transfer rules.

be true:
$can.obtain(C, r, customer, database1, CPG_0) = true$ Where the C condition is: $C = C_1 \wedge C_2 = (T, T, send\_proposal(customer, salesagent) \wedge request\_quote(salesagent, supplier1), T)$
In this way several samples of the applying transfer rules on the graph in Figure 22 is also shown in Figure 23.

5 Related Works

We have proposed the Constrained Policy Graph (CPG) model, which inherits Take-Grant protection model (TG) [8] basic concepts. TG is an access control protection model which represents transformation of rights and information between entities inside a system either implicitly or explicitly. It was first presented by Jones et al. to solve the "Safety Problem". To model the transfer of rights and information, TG uses a set of rules called de-jure rules and de-facto rules [9–11]. TG doesn't provide a method to restrict the ACPs with constraints. Also, it doesn't support policy combination and nested policy specification. Unlike TG, CPG provides the means for nested definition and constrained specification of ACPs. Furthermore, it contains the necessary rules for the policy combination and expansion. We define the constraints by restricting subjects, objects and actions. In addition, CPG model has the ability of time-dependent policy definition. Different kinds of constraints can be combined using our method clearly. Also, we can discover the implicit transfers of rights and information by applying a set of rules on each CPG graph. Ultimately, using CPG model, we can analyze the presence or absence of the conflict or contradiction among ACPs.

Many researches have been devoted to modeling ACPs. Most of these methods have modeled the policy of a system on the basis of a single policy. For example, Bertino et al in [1, 2] proposed a temporal model for the access control. However, their method does not support the possibility of the application of several policies. Of course, this need for the composition has been taken into consideration by some methods. As an example:

Bonatti et al in [3], proposed an algebraic method for the specification and composition of ACPs. Contrary to previous methods, this method makes the composition of the policies feasible, but CPG has also considered temporal constraint specification. On the other hand, the transfer information which might take place because of the relation of the entities with different policies can be discovered by CPG and can be added to the system policy graph (as it was observed in Section 2.2).

Siewe et al in [4] proposed a method for the specification and composition of the ACPs. This method has considered the policies on the basis of temporal dependencies among the access permissions using Interval Temporal Logic. As it was brought up before, in addition to the temporal constraints, other constraints (e.g. constraints on subjects, objects, etc.) are required for the determination of the validity condition of an ACP in our goals, which isn't studied in this method.

Jajodia et al in [5] proposed a method for the application of several ACPs in a single system. Different policies in this method are described by a unique language and are analyzed on the basis of the decision-making rules defined. These policies are saved in the system library, so depend on the requests from users, new rules are added to it. The purpose of the application of several policies in one system is to translate the policies into a single language, save and analyze them in this system, while in CPG, different types of the composition of different policies (on demand) are presented. In addition, the work in [5] has only taken the constraints into consideration for the classification of the objects and grouping of the subjects, while the others are ignored.

A very similar work to our proposal in objectives is the work of Koch et al. in [29] which presents some basic properties of a formal model for AC policies based on graphs and graph transformations and addresses the problem of detecting and resolving conflicts. In the presented framework in [29] a policy is formalized by four components: a type graph, positive and negative constraints and a set of rules.

Positive and negative constraints in [29] can be considered as a formal documentation of the initial requirements and of the development process of rules which is conceptually similar to the definition of constraints in CPG. While constraints in [29] are modeled using graph transfer rules, in CPG we use FOL.

Using logic in CPG not only provides more expressive power than graph transfer rules, but also helps in more granular and more sophisticated specification of constraints.

An important aspect of the work in [29] is supporting type information of the AC policy via type graph that specifies the node and edge types which may occur in the instance graphs modeling system states. CPG does not support specifying type information; however in CPG we consider nodes to be subject, object or both. When an actual AC policy is specified using CPG, each node (subject/object or both) is represented by its actual name (e.g. manager, employee, library document, etc. see examples in the paper). Moreover CPG benefits from considering various constraints on subjects, objects. Considering this information in addition to node names, CPG takes advantages of an implicit type system which makes a rigorous framework for identifying various entities in the system without having an explicit type system.

The integration of two AC policies in [29] is done on the syntactical level and is limited only to the union of two policies where their common sub policy is not duplicated. This is while CPG supports more integration scenarios including intersection and subtraction of AC policies.

Considering the above discussion, CPG provides more granular and sophisticated model for detection of conflicts and analysis of inconsistencies. However Koch et al. propose strategies to automatically resolve the conflicts after detection which is not considered in CPG.

As we know, an access control model is an abstract of the system security policies which bounds the relation among the entities of a system. Since logic enjoys a formal foundation, it then provides independence of the model from the manner of implementation, deductibility, possibility of correctness verification, flexibility, and advantages of a descriptive nature while being simple and expressive. A simple, reliable, and general bed for access control systems is then obtained by logic. In spite of these advantages, the most important problem with logical methods is their lack of understanding by those users unfamiliar with it. Another advantage of CPG against the above methods is that while it benefits from a formal infrastructure, at the same time the specification and analysis of the policies and their perception can be simpler by taking advantage of the graphical scheme of the graphs.

One of the important problems most systems still suffer is that different systems have not been able to properly express their policies yet. In most cases, they express their own policy, but when a part of the policy is not specified, it is therefore interpreted differently. Some systems consider and allow unspecified policies and some do not allow them. CPG can discover the implicit transfers in the system and add them to the access control model of that system by the application of transfer rules. In this way, the possible transfers as well as the conditions for their possibility are discovered and if they are against the related system policies they can be updates by an appropriate definition of what the system needs. Also, flexibility in the definition of the constraints in CPG provides the possibility of policy precise adjustment according to the system demands. Moreover, different levels of abstraction are supported by our method (employing the policy expansion rule in Section 2).

After the introduction of CPG in this paper, we studied the problems existing in the web service composition and the manner of their solution by CPG. Web service composition methods only specify the functional specifications of the compositional web services and also, verify and analyze these functional properties of the composition (such as [19–26]).

Rouached et al in [14] modeled the ACPs of web services using Event Calculus (EC). It specifies the access request to web service and also the result of this access (confirms or rejects). The target of this research is to prevent the assignment of two contradicted rights to one role which might lead to a conflict among a set of requests. In fact the work in [14] expresses the request for the operation of an action on a web service and its possibility or impossibility by the use of EC, but it does not address on what policy and access control model such a decision is taken. This method does not have a way for definition of an access control model of a web service and its manner of modeling and composition of the policies. Moreover, many states are required to be studied according to our targets which are not intended in this method: Consider the following conditions of three web services of A, B, and C. Assume that A permits B to have an access permit to its sources, but rejects the requests received from C. If B authorizes the requests from C, such a set might be indirectly incompatible with the policy of A. It is because web service C might be able to have access to the resources of A through B. This problem requires a method with a capability to study the transfer of rights and information in the compositional system and also to arrange the constraints in an appropriate way to prevent such a transfer.

Li et al. also in [30] present a graph transformation based approach to check whether an internal business process of an organization implemented using BPEL conforms to the organization's privacy policy or not. Also they propose a formal consistency verification using graph transformations. Authors in [30] use an approach similar to that of Koch et al [29] for modeling and verification of policies which was studied and compared to our work previously in this section. Furthermore their work suffers from never considering policy integration and policy expansion which have been considered in CPG.

6 Conclusion

In this paper, a method for specification and combination of access control policies is represented. This method is capable of expressing nested authorization policies, conditional (constrained) policies, combination of policies, and verification of them to find conflict amongst the combined policies. We considered the implicit transfers of the rights and information which happens because of the entities, as well as relations with different policies in the system. Furthermore, we showed one of the main applications of our proposed model in the web service composition. We presented the way of modeling web services' ACPs and extracting the ACP of composed web services, using CPG model. This outcome can be analyzed the existence of the conflicts or contradictions between composition partners' policies.

Acknowledgements

We would like to thank the work done by handling editor and the anonymous referees for their scholarly criticism. We are also grateful to Dr. Morteza Amini for his comments on earlier versions of this paper which were a great help in improving the paper.

References

[1] E. Bertino, C. Bettini, E. Ferrari, P. Samarati. A Temporal Access Control Mechanism for Database systems. *IEEE Transactions on knowledge and data engineering*, 8(1): 67–80, February 1996.

[2] E. Bertino, C. Bettini, E. Ferrari, P. Samarati. An Access Control Model Supporting Periodicity Constraints and Temporal Reasoning. *ACM Transaction on Database Systems*, 23(3): 213–285, September 1998.

[3] P. Bonatti, S. D. C. di Vimercati, P. Samarati. A modular approach to composing access control policies. *In Proc. 7th ACM Conf. on Communications and Security*, pages 164–173, August 2000.

[4] F. Siewe, A. Cau, H. Zedan. A Compositional Framework for Access Control Policies Enforcement. *FMSE'03, Washington, DC, USA*, pages 32–42, October 30, 2003.

[5] S. Jajodia, P. Samarati, V. S. Subrahmanian, E. Bertino. A unified framework for enforcing multiple access control policies. *ACM transaction on Database Systems*, June 2001.

[6] E. Bertino, S. Jajodia, and P. Samarati. A flexible authorization mechanism for relational data management systems. *ACM Transactions on Information Systems*, 17(2): 101–140, April 1999.

[7] Alexandre Alves, et al. Web Services Business Process Execution Language. *OASIS*, 31 January 2007.

[8] A. Jones, R. Lipton, and L. Snyder. A Linear Time Algorithm for Deciding Security. *Proc 17 th Annual Symp. On the Foundations of Computer Science*, 33–41 October, 1976.

[9] M. Bishop. Conspiracy and information flow in the Take-Grant Protection Model. *Journal of Computer Security*, vol 4(4), pp. 331–360, 1996.

[10] M. Bishop. Theft of Information in the Take-Grant Protection Model. *Journal of Computer Security* (1994/1995).

[11] M. Bishop. The transfer of information and authority in a protection system. *ACM*, 1979.

[12] S. D. C. di Vimercati, P. Samarati, S. Jajodia. Policies, models, and languages for access control. In S. Bhalla, editor, DNIS, volume 3433 of Lecture Notes in Computer Science, pages 225–237, Springer, 2005.

[13] A.Charfi, M.Mezini. Middleware Services for Web Service Compositions, *Chiba, Japan, ACM .WWW* 2005.

[14] M. Rouached, Claude Godart. Securing Web Service Compositions: Formalizing Authorization Policies Using Event Calculus. *ICSOC 2006:* 440–446.

[15] Z. Derakhshandeh, B. Tork Ladani, N. Nematbakhsh. Verification of Access Control Policies in Composition of Web Services Using Take-Grant Protection Model. *in Proceedings of the fourth Iranian Society of Cryptology Conference (ISCC07), Iran University of Science and Technology*, October 16–18, 2007.

[16] Cheng, Y., Lee, E. W. and Dilip, K. L. Web Services Composition- An Overview of Standards. *Information Technology Standards Committee*, Section Four, Oct. 2004.

[17] A. Assaf et al. Web Service Choreography Interface 1.0. *W3C*, August 2002.

[18] Ter Beek, M.H., Bucchiarone, A. and Gnesi, S. A Survey on Service Composition Approaches: From Industrial Standards to Formal Methods. *Technical Report 2006-TR-15*, ISTI, Consiglio Nazionale delle Ricerche.

[19] Salaun, G., Bordeaux, L. and Schaerf, M. Describing and Reasoning on Web Services using

Process Algebra. *In Proceedings. IEEE International Conference on Web Services*, pages 43-50, 2004.

[20] Ferrara, A. Web Services: a Process Algebra Approach. *In Proceedings of the 2nd International Conference on Service-Oriented Computing (ICSOC'04), New York, NY*, pages 242–251. ACM Press, 2004.

[21] G. Diaz, Juan-Jose Pardo, Mara-Emilia Cambronero. Verification of Web Services with Timed Automata. *Electronic Notes in Theoretical Computer Science* 157 (2006) 19–34.

[22] Mariya Koshkina, Franck van Breugel. Verification of Business Processes for Web Services. *York University, Department of Computer Science*, October 2003.

[23] L.G. Meredith, S. Bjorg. Contracts and Types. *Communications of the ACM*, 46, No. 10, pp. 41–47, October 2003.

[24] H. Schlingloff, A. Martens, K. Schmidt. Modeling and Model Checking Web Services. *Electronic Notes in Theoretical Computer Science* 126 (2005) 3–26.

[25] Ganna Monakova, Oliver Kopp, and Frank Leymann. Improving Control Flow Verification in a Business Process using an Extended Petri Net. *1st Central-European Workshop on Services and their Composition, ZEUS 2009, Stuttgart, Germany*, March 2–3, (2009).

[26] Song W., Ma X., Ye C., Dou W. and L J. Timed Modeling and Verification of BPEL Processes Using Time Petri Nets. *9th International Conference on Quality Software*, pp. 92–97.

[27] Zahra Derakhshandeh, Behrouz Tork Ladani, Naser Nematbakhsh. Using Constrained Policy Graph for Modeling and Analysis of The Web Service Composition Policies. *in proceedings of the IADIS International Conference WWW/INTERNET 2008, Freiburg, Germany*, 13–15 Oct. 2008.

[28] Zahra Derakhshandeh, Behrouz Tork Ladani, Naser Nematbakhsh. Application of Constraint Policy Graph for Specification and Analysis of Access Control Policies in BPEL processes. *in Proceedings of the 6th International ISC Conference on Information Security and Cryptology (ISCISC'09), University of Isfahan*, October 2009.

[29] M. Koch, L. V.Mancini, and F. Parisi-Presicce. Graph Based Specification of Access Control Policies. *J. Comput. Syst. Sci.*, 71(1):133, 2005.

[30] Y. Li, H. Paik, and B. Benatallah. Formal consistency verification between BPEL process and privacy policy. *In Proceedings of the 2006 International Conference on Privacy, Security and Trust: Bridge the Gap Between PST Technologies and Business Services (PST '06), ACM, New York, USA*, 2006.

[31] H. De Nivelle and I. Pratt-Hartmann. A resolution-based decision procedure for the two-variable fragment with equality. *In Proc. IJCAR'01*, vol. 2083 of LNAI, pp. 211–225. Springer, 2001.

Zahra Derakhshandeh received her B.Sc. degree in 2005 and M.Sc. degree in 2008 both from University of Isfahan. After that, she is the faculty member of computer science and IT department of Sheikhbhaee University. Her main interests are Network and Security.

Behrouz Tork Ladani received the BS degree in Software Engineering from University of Isfahan (Isfahan, Iran) in 1996, MS in Software Engineering from Amir-Kabir University of Technology (Tehran, Iran) in 1998, and PhD in Computer Engineering from Tarbiat-Modarres University (Tehran, Iran) in 2005. He is currently an Assistant Professor and head of Department of Information Technology in University of Isfahan. His research interests include Cryptographic Protocols, Access Control, Trust, and Formal Verification. He is currently member of executive council of Iranian Society of Cryptology (ISC).