Integrating Access Control Obligations in the Session Initiation Protocol for Pervasive Computing Environments

Hassan Sharghi(✉) and Ramiro Liscano

Department of Electrical, Computer and Software Engineering,
University of Ontario Institute of Technology, Oshawa, ON L1H 7K4, Canada
{Mohammadhassan.Sharghigoorabi,Ramiro.Liscano}@uoit.ca

Abstract. The widely use of advanced technologies in the sensor network and computing has facilitated the development of convenient pervasive applications in order to access information at anytime and anywhere. The traditional access control mechanisms cannot appropriately protect the access and usage of digital resources in the highly distributed and heterogeneous computing environment. In such an environment, enforcing continuously the access control policies during the access period is a challenge because traditional authorization decisions are generally made at the time of access requests but do not consider ongoing controls. Obligations are the vital part of many access control policies and they specify mandatory behavior that should be conducted by a user of the access control system in sensitive domains. Therefore, utilizing a mechanism to approve the fulfillment of the obligation is required for continuing or revoking the access decision. We leveraged the capability of Session Initiation Protocol (SIP) to manage the communication between entities in order to provide a mechanism to handle the continuous enforcement of the obligation. Meanwhile, we present several scenarios which indicate our proposed model can manage the obligatory behavior that affects the continuity of access to resources in pervasive computing environment.

Keywords: Access control · Obligation · Behavior · Session initiation protocol · Pervasive computing

1 Introduction

Pervasive computing provides an environment where computing systems can be utilized everywhere in order to allow users to access any time to data and services. The continuous development of mobile devices, sophisticated sensor networks, and the mature communication technologies enable the pervasive computing paradigm to access a variety of resources in order to process the collected data from interconnected objects to provide different services. Satisfying the various needs of users to obtain different services at anytime and anywhere in such a heterogeneous environment generates challenges for security and privacy protection.

© Springer International Publishing AG 2017
E. Aïmeur et al. (Eds.): MCETECH 2017, LNBIP 289, pp. 24–40, 2017.
DOI: 10.1007/978-3-319-59041-7_2

Access control mechanism is an acceptable approach in security domain to allow legitimate users to access different resources in the secure manner. Typically, access control deals only with authorization decision on users request to access the target resources. The objective of access control is to prevent from unauthorized disclosure (confidentiality), avoid improper malicious modifications (integrity), and ensure the access for authorized entities (availability).

Predefined policies are often used to manage the access control system. A policy includes a set of rules to specify the permission of a subject for performing some actions on resources if particular conditions are satisfied. Separating the policy from the implementation of a system helps to modify adaptively the policy. The modification of policies enables the administrator to handle the changes in the strategy for managing the system and controlling the behavior of users without changing the implementation of the underlying components.

Access control is concerned with the problem of granting access to resources and services, but controlling the activities of authenticated and authorized users is not in charge of the traditional access control systems after granting permission. Furthermore, traditional access control models such as Mandatory Access Control (MAC), Discretionary Access Control (DAC), and Role-Based Access Control (RBAC) do not work properly in pervasive computing environment because such an environment is dynamic and sensitive to contextual information as well as pervasive computing applications usually do not have well-defined security perimeter [22].

Context information should be involved in access control system for pervasive computing applications because the effective attributes of entities may change based on the context over time. For example, a subject in the role of data provider, equipped with a mobile device, may send geolocation data to a network operator who has the role of data consumer. However, the role of data consumer for network operator may change to data provider when a location-based service provider requests for this specific data. Context data can come from various sources such as location sensors, calendars, clocks, and many other sensors. The level of granularity is an important characteristic of context data. For instance, a user's location can be represented in exact coordinates, or only the name of country. Consequently, the granularity of provided context data affects the precision of access control system.

Different kinds of policies are required to protect sensitive resources in pervasive computing application domain. Authorization policies representing by a set of rules protect resources through making appropriate access decisions. However, such authorization rules cannot control the activities of the user after granting the access. Integrating obligations into access control policies can dictate the mandatory operation that must be performed by the user before, after, and during the access [7].

We define the obligation as an obligatory behavior that should be conducted by an entity. Obligation is classified in two perspectives: functional and structural [6]. Functional obligation represents what a subject is obliged to perform with respect to change or maintain the state of affairs. In this view, obligation

is a commitment that represents a promise to do something or to behave in a particular manner. Structural obligation represents what a subject is obliged to perform in order to fulfill a responsibility such as directing, supervising and monitoring. In this view, obligation is a duty that a subject must perform and it is not specifically part of the access control purpose.

Using obligations to handle the usage of resources raises the question of how the enforcement will be managed. Since traditional access control is performed only once before the resource is accessed by the subject, the access control system is not able to control the fulfillment of obligation while the access is in progress. Therefore, the lack of capability to check the obligations for the continuity of access purpose may lead to the security breach in sensitive environments. To resolve this issue, we proposed to utilize Session Initiation Protocol (SIP) in order to handle the enforcement of obligations. SIP is an application-layer control protocol for creating, modifying, and terminating sessions with one or more participants [8].

In this paper, we propose an approach to integrate obligations into access control system in order to control the behavior of the user. Overall, this paper presents the following contributions: (i) modeling the obligation based on subject's behavior and using a behavior based language to express the obligation, and (ii) applying session initiation protocol to approve the fulfillment of the obligation for the purpose of continuing or revoking the access decision.

The remaining of this paper is organized as follows: Sect. 2 provides an overview of related works. Our proposed approach for modeling and enforcing the obligation is presented in Sect. 3, followed by two application scenarios in Sect. 4 in order to illustrate some of our approach's functionalities. Finally, Sect. 5 provides some concluding remarks and outlines the direction for future research.

2 Related Work

Access control has been intensively explored in recent years and several access control models have been proposed. The dynamic nature of context information makes trouble to develop a fine-grained access control in the pervasive computing application domain. Therefore, there are several attempts such as [18,21] to improve the existing access control models using contextual information for pervasive computing applications. In [12], authors developed a context-aware role based access control model for pervasive computing applications through creating a new model for role. They defined roles as part of an application's design so that roles come into existence only during the application's lifetime instead of giving a long lifetime to a role.

Access control model based on a particular context such as location [2,24], time and spatial information [17] were also suggested. Toahchoodee et al. [23] proposed trust based access control for ubiquitous computing application where users are not known in advance. The access privilege of a user relies on his trust level which in turn depends on the contextual information.

Our research is different because we do not rely on an access control model based on a particular attribute or context. Our approach can be applied for any models that possess extensible capability to express the obligations. However, attribute based access control provides sufficient opportunity to apply our approach for a specific domain.

Damiani et al. [4] proposed a method to solve the problem of how to introduce continuous enforcement in location-based access control. They considered user movement as a parameter for the validity of access request and used the usage control process for the continuous enforcement of a request.

However, existing usage control models [16] are general purpose and do not address the specific aspects of contexts. Meanwhile, the lack of comprehensive ongoing enforcement mechanism and post decision checking limit the application of usage control and its practical utilization. Katt et al. [11] proposed an extension to the usage control model in order to enhance the continuity of enforcement model by configuring a meta-model. This model consists of two sub-modules: attribute decision function and obligation decision function. The first is responsible for authorizing a subject to do some actions and the second checks the obligation that the subject must fulfill.

Using state machines to model obligation enforcement was widely explored in literature so that authors in [11,25] applied formally state machine to model the obligation. Li et al. [13] modeled an obligation as a state machine interacting with PEP (Policy Enforcement Point) and the outside world through events. Meanwhile, using logic rules is another usual method to formalize obligation enforcement [5]. Our proposed enforcement model is different in terms of considering the duration of time for obligation enforcement. We propose that a SIP session be used to constrain the enforcement duration of an obligation.

Liscano et al. [14] leveraged Session Initiation Protocol (SIP) to manage the service discovery process in order to extend access to particular local services to remote participants. Karopoulos et al. [9] applied SIP architecture to handle the continuity of authorization based on usage control model. They proposed an authorization support for SIP by integrating the whole aspects of usage control along with the obligation in SIP system [10].

We also leveraged the SIP protocol to propose an architecture to approve the obligation fulfillment. However, our work is different in terms of modeling and expressing the access control obligation. We modeled the obligation based on the behavior of the subject who is responsible to perform the obligation. To express such a model of obligation, we utilized a behavior pattern language that enables the policy administrator to define a variety of subject's behavior.

3 SIP_Based Enforcement of Obligation

The traditional access control is unable to deal with the changes of attribute values of subject, object and environment after the access has been granted. However, values of attributes, that already participated in making an authorization decision, usually changes in dynamic environment. Condition and obligation are

two factors that play significant role in making authorization decision in modern access control systems. Conditions are related to the stateless features of attributes whereas obligations are concerned with commitments of the involved entities. Fulfillment or violation of an obligation often affects the stateful features of attributes.

In terms of approving the fulfillment of an obligation, a predicate represents the obligation in order to verify whether a certain activity has been fulfilled or not. The evaluation of obligation's fulfillment is applied to the access request before, during, and after usage decision. Pre obligation is a predicate that checks if specific activities have been executed or not before usage exercise based on the recorded activities of system's entities. Ongoing is a predicate that should be satisfied periodically during the usage of granted access. If the evaluation of the predicate is postponed after making decision for the current request and used for future usage decision, it is called post obligation. From the continuity perspective, the fulfillment of ongoing obligation has the main effect on the continuity of the granted access. Therefore, we present a mechanism for checking the adherence to the agreed commitments in functional and ongoing obligations that are expressed by access control policy language.

We need to manage an obligation session that allows the system to monitor the behavior of a user to comply the defined obligation for accessing a particular resource or when the access is in progress. We chose SIP from different session control protocols because of its capabilities for event notification, presence and instant messaging service as well as mature features such as:

- SIP allows two or more participants to establish a session without relying on the content of the media streams. The media streams can be audio, video or any other Internet based communication mechanism such as distributed games, shared applications, and shared text editors [19].
- The protocol was standardized by the Internet Engineering Task Force (IETF) and was implemented by a number of vendors. SIP allows the end-points to create, modify, and terminate any kind of media sessions, such as VoIP calls, multimedia conferences, or data communications in distributed systems.
- The SIP protocol supports an intermediate network element called the Back-to-Back User Agent (B2BUA) which operates between two communicating User Agents (UAs). The main advantage of using B2BUA is that it can provide a session management and full control over the session.
- SIP uses the Session Description Protocol (SDP) to help end users to negotiate the characteristics of the session. After session establishment, the protocol behaves as a Peer-to-Peer (P2P) one and allows users send messages directly to each other.
- Utilizing SDP inside of SIP provides a powerful capability to extend new services such as acquiring information about end-points, identifying the originator of a session, and identifying the content-type of a message in a session initiation request.

Increasing the processing load for parsing SIP messages is the main drawback of using standard SIP protocol. Particularly, in constrained environments, processing the large size of text-based SIP messages makes trouble in terms of memory and power consumption. However, recent researches [3,15] to introduce a lightweight SIP targeted to constrained environments encourage us to utilize the capabilities of it for session management in our approach.

3.1 A Behavior-Based Model for Obligation

We define a behavior as a sequence of events that are issued by a subject. An event is represented by a tuple of attributes that describe the characteristics of the interaction between a subject and an object. We assume that events possess the same type in terms of the number of attributes and the domain values of attributes. A behavior consists of two parts "static" and "dynamic".

The static part defines the structure of the behavior and includes the number of events and the event type. In fact, it represents the common characteristics between the events. The dynamic part defines the semantics of the behavior by making a correlation between the events in the behavior. Such correlations are described by a set of constraints that must be satisfied while conducting the behavior. These constraints manage the sequence and association among the events. In other words, the semantics of the behavior is represented by a set of constraints that provide association among the events or specify particular conditions for the behavior.

In access control perspective, the obligation is an operation that should be performed before, after, or during the access. We consider the following features that can specify different aspects of the obligation:

Type: obligations are categorized into three types based on when they become effective. (i) Pre-obligation specifies actions that should be fulfilled before the access of resources is allowed; (ii) ongoing-obligation specifies actions need to be fulfilled during enforcing the access (the usage of resources); and (iii) post-obligation specifies actions required to be performed after accessing to resources.

Subject: the subject of an obligation is a person (in the case of user obligation) or a component of the system (in the case of system obligation) that fulfills the obligation. PEP is responsible to approve the fulfillment of the obligation.

Object: the obligation object presents to whom the obligation must be applied. The obligation object may not be the same object of the access request. The object is a finite set of resources or state of circumstances.

Operation: the subject must perform a mandatory operation. Practically, the operation is relevant to access the resources or its execution affects the state of circumstances. The word "operation" comes from the definition of obligation in XACML standard [1] and we define it as a set of planned actions to achieve a particular purpose. Performing the operation by the subject generates a sequence of events that we call it as the behavior of the subject. Actually, the concept of the obligation dictates the behavior of the subject in a particular situation.

Violation: obligations may include requirements to specify that the fulfillment of the obligation should satisfy particular stateful constraints such as durability, deadline, location, etc.

According to the aforementioned features, we consider formally the obligation as the behavior of the subject who is in charge of performing the obligatory actions and present it as a tuple *obl(w, s, b, o, v)*. *w* represents the obligation fulfillment time and PEP handles when the obligation should be fulfilled. The subject *s* should conduct the behavior *b* on the object *o* in order to satisfy the stateful conditions *v*. The defined model for obligation helps to express the complex obligation specifications in terms of defining a sequence of actions and corresponding conditions.

Specifying precisely the behavior of a user is a challenge. A user should issue a particular behavior to fulfill the obligation so that the characteristics of such a behavior can be represented by a Behavior Pattern Language (BPL) introduced in [20]. The BPL code consists of several parts to represent a typical behavior pattern. The declaration is used to describe different entities existing in the behavior. These entities include: attributes, events, sequence of events and some parameters that specify the structure of the behavior. In other words, the skeleton of a behavior will be defined in the declaration section, then the pertinent operations and constraints will be called to establish the meaning of the behavior. The methods in the operation section are used to define the relation between events. The relation can be defined through applying different statements and operators to manipulate attributes and attribute values. If someone intends to define particular restrictions for the operation, the relevant assertions will be defined in the constraint section.

Such a language allows the access control policy administrator to illustrate primitive events and their attributes, semantics, constraints, and behavior patterns in a detailed and precise manner. The elements of the defined tuple for the obligation can be specified by BPL. However, the obligation specified by BPL should be interpreted and converted to the XML (Extensible Markup Language) format to be compatible with XACML.

3.2 Obligation Handling in XACML

XACML is a standard developed by the OASIS organization [1] for expressing syntax and semantics of access control policies. It provides a language, a reference architecture and the processing model to describe how to evaluate an access request. Briefly, the XACML promises to be an expressive and flexible policy language and architecture for heterogeneous and dynamic environment.

The XACML Technical Committee provides a Policy Schema for the XACML policies so that at the root of all XACML policies there are tags <Policy> or <PolicySet>. A policy set can contain other policies or policy sets. A policy consists of a target, which denotes the target of the policy, a set of rules elements which represent the authorization rules, and a rule combining algorithm. The target specifies the subjects, resources, actions and environments on which a policy can be applied. A rule includes three main components: the Target,

the Condition, and the Effect of the rule which can be either Permit or Deny. The Target specifies for which kind of access requests the rule can be applied. The Condition elements denotes predicates in order to evaluate the attributes. A rule can include optionally an ObligationExpression element to specify action that should be performed after enforcing the authorization decision. A rule is applicable to an access request if the target of the access request matches the target of the rule and if all the conditions included in the rule are satisfied. Meanwhile, the predefined combining algorithms are used to resolve the conflict among applicable rules during the decision making.

XACML allows to express fine-grained access control policies, but it is not expressive enough to define different kinds of obligation particularly ongoing obligation that can be used to control the continuity of a granted access. In general, XACML does not possess specific constructs to address the continuity of policy enforcement. Specifying obligations using a behavior-based language and leveraging SIP protocol are our approach to enhance the expressiveness and capability of the XACML.

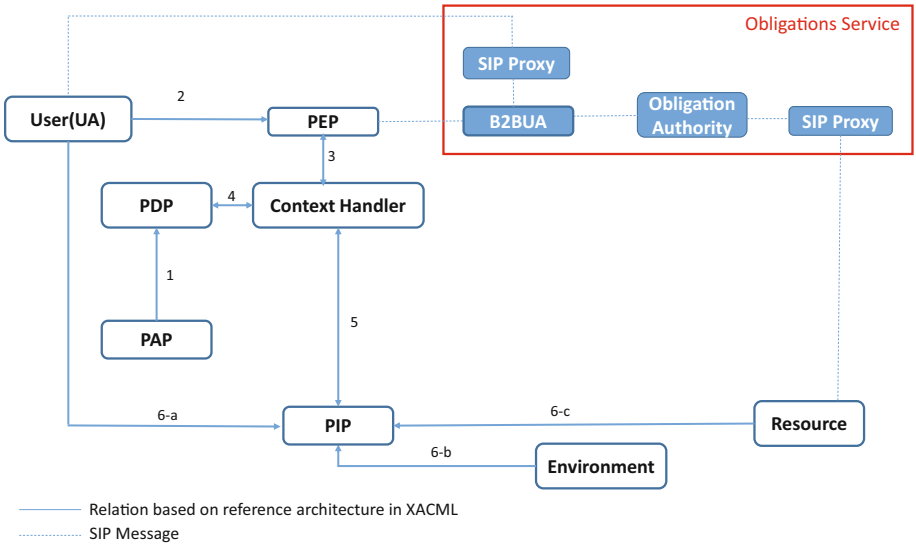


Fig. 1. Extending the obligation service in XACML reference architecture to handle the pre and ongoing obligations

3.3 Wrapping up Obligations in SIP Message

Session Initiation Protocol (SIP) is an application layer protocol that is used to set up a media session and help end users to exchange data. To establish a session, the requester sends the INVITE request that contains a number of header fields. Header fields are named attributes that provide additional information about a

message. The body of a SIP message contains a description of the session that is encoded by a protocol called Session Description Protocol (SDP). The TRYING response indicates that the INVITE has been received and the proxy is working on her behalf to route the INVITE to the destination. If the destination decides to answer the request, it will send OK response to the requester. Any of the two users can send a BYE message directly to the other in order to terminate the session.

SIP protocol provides a logical entity called Back-to-Back User Agent (B2BUA) that operates between two communicating User Agents (UAs) and controls all signaling exchanged between them. In other words, it is a concatenation of a User Agent Client (UAC) and User Agent Server (UAS). It receives request from the calling SIP user agent (client), and forwards them to the called SIP (server). Therefore, B2BUA as an intermediary prevents the end points from direct exchanging of messages. Applying the B2BUA provides a facility for communication management for the whole duration and full control over the connection.

As can be seen in Fig. 1, a Policy Administration Point (PAP) determines which policies should be defined and deployed to a policy repository. Later on, such predefined authorization rules inside policies can be used by Policy Decision Point (PDP) to make a decision for an incoming access request. A User Agent (UA) generates the request in order to achieve the permission for access to a particular resource. Policy Enforcement Point (PEP) receives the request. PEP transmits the original request to the Context Handler where an XACML request context is constructed. PDP receives the XACML request in order to make a decision based on policy rules. For this purpose, PDP requests any additional subject, resource, action, environment and other categories attributes from the context handler. Policy Information Point (PIP) is responsible to provide values for such additional attributes and returns the attribute values to the PDP through the Context Handler. PDP evaluates policies based on provided attribute values to make the decision for granting or denying the request. The authorization decision along with obligations will be sent to the PEP through the Context Handler for enforcement. In summary, the procedure to evaluate an access request based on policy rules is as follows:

1. Flow 1: PAP writes policies and corresponding obligations and make them available to the PDP
2. Flow 2: The access requester sends a request for access to the PEP
3. Flow 3: The PEP sends the request in its native format to the context handler
4. Flow 4: The context handler constructs an XACML request context and sends to PDP
5. Flow 4: The PDP requests attribute values and any additional attributes from the context handler
6. Flow 5: The context handler communicates with PIP to obtain the attribute values
7. Flow 6-a, 6-b, 6-c: The PIP obtains attributes of the subject, resource, action, and environment

8. Flow 5: PIP returns attributes to the PDP through the context handler
9. Flow 4: The PDP evaluates the policy
10. Flow 3: The PDP returns the authorization decision along with applicable obligation to the PEP via context handler

In the XACML workflow, there is no information for obligation enforcement and the role of obligation for the continuity of granted access. We leveraged SIP to provide a communication paradigm between entities in a distributed access control system in order to handle the obligation. For this purpose, we add SIP components such as SIP proxy, B2BUA as well as an Obligation Authority (OA) inside of "Obligations Service" provided by XACML architecture. We consider the OA as a third party (or resource owner) that obliges the user to follow a behavior before access to resources or when the access is still in progress. We describe the interactions between SIP and access control components to handle pre-obligation and ongoing obligation as follows:

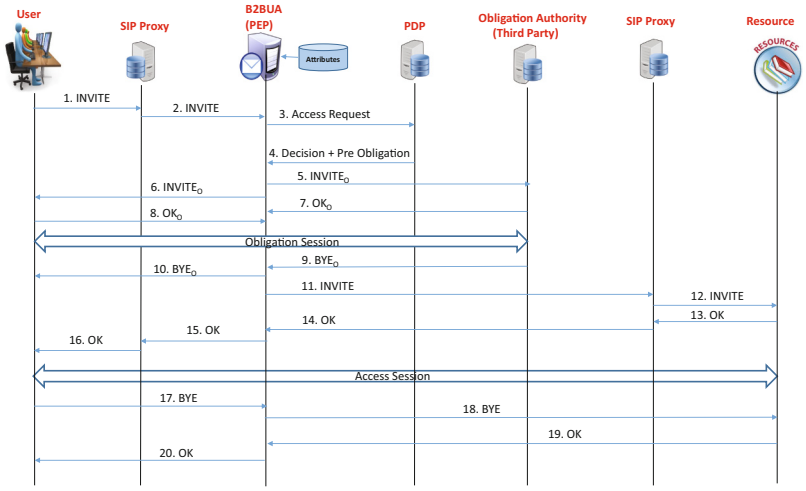


Fig. 2. Applying a SIP session to handle the pre-obligation

Pre-obligation: In some situations, the fulfillment of an obligation is a prerequisite to allow the user to access a resource. Figure 2 shows the sequence diagram in the case of pre-obligation so that the complete execution of an obligation is needed before permitting the user request. We use a B2BUA to achieve a full control over the SIP session. We got the idea from [10] to integrate the B2BUA with the PEP for decreasing the complexity of implementation. Meanwhile, handling obligation outside the PEP will greatly complicate the interface between them and decrease the performance. B2BUA is a logical SIP entity that provides a proper opportunity to implement the logics inside of the PEP for managing the obligation life-cycle. When a user sends a request to access a particular resource,

the request will be wrapped up with SIP message called INVITE and will be sent to B2BUA through SIP proxy. PEP extracts the request and sends to PDP to make decision for authorization based on predefined policies and obligations. PDP sends the final decision along with the applicable obligation to the PEP. In this case, we assume that the obligation is a pre-obligation that should be enforced before granting the permission to the requester. Therefore, the B2BUA suspends the request and sends a SIP invite message ($INVITE_O$) to both Obligation Authority(OA) and requester (User) to start an obligation session. When the user performs the obligation, the OA approves the fulfillment by sending a BYE message (BYE_O) to B2BUA. Then, B2BUA resumes the user's access request and sends it to the resource for establishment an access session.

Ongoing Obligation: Ongoing-obligations need to be fulfilled in parallel with the requested access. It means that ongoing-obligations should be executed when the user performs actions on a granted resource. The user may be permitted to access the resource in the beginning, but if the user fails to fulfill the ongoing-obligation, the access will be revoked. Figure 3 shows that the user sends the access request wrapped up by SIP invite message to the B2BUA. Similar to the pre-obligation, PDP evaluates the access request to provide authorization decision plus applicable obligation. However, if the obligation is ongoing-obligation then the B2BUA establishes an obligation session between the user and the OA which should remain active for the whole period of the access. After obligation session establishment, B2BUA conducts the suspended INVITE message to resource through SIP proxy in order to establish an access session. During the access session, if B2BUA receives a BYE_O message from the user to stop the obligation session or B2BUA recognizes changes in specific attribute values for the purpose of terminating the obligation session, it sends a BYE_O message to the OA to terminate the obligation session. Then, it sends immediately a BYE message to the resource and the user to terminate the access session.

4 Case Studies

We describe two application scenarios based on examples presented in [16] in order to illustrate the functionality of our proposed approach. Similar situations to each of these scenarios were applied in literature to focus on particular aspect of the access control. For example, the situation in Scenario 1 was also used in [12] to present the need for extending role-based access control model to support context-based access control requirements for pervasive computing systems. Scenario 2 was widely used as a reference example to apply obligation for handling the usage control [10,13]. Each of such scenarios can be realized by using different aspect of access control such as authorization, obligation, or condition predicates. We look at the obligation perspective of such scenarios to propose a solution that ensures a continuous access control while the subject's behavior changes.

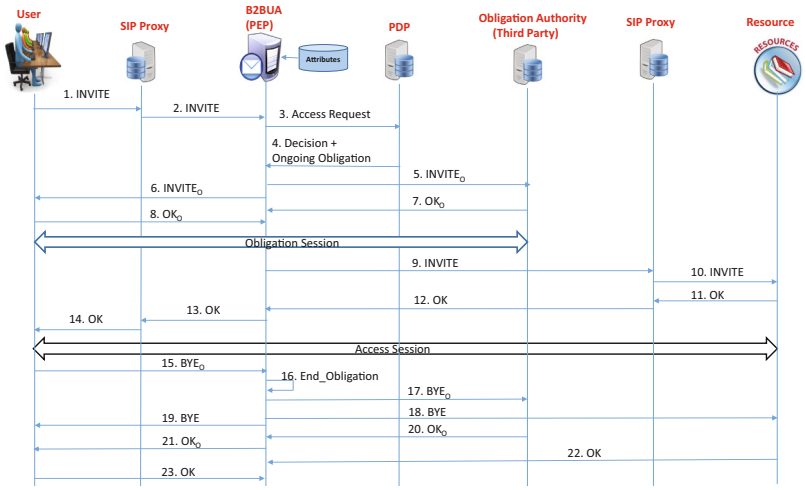


Fig. 3. Applying a SIP session to handle the ongoing obligation

In this paper, we concentrate on the ongoing obligation because the fulfillment of this kind of obligation should be continuously checked after access permission. The continuity of access to a service or resource is a challenge due to the dynamic nature of pervasive environments. In access control, continuity means that a control policy should be enforced not only before an access, but also during the period of the access. The dynamic nature of context information in pervasive environment requires the capability of access control model for revocation or continuity of a granted access when certain context conditions fail to hold. Pre-obligation and post-obligation cannot affect on the continuity of access, although our proposed approach can be used for enforcement of such obligations.

Scenario 1: We focus on combining the smart devices and hospital information system deployed in healthcare centers. Such systems can be accessed remotely or locally by nurses and doctors. The system is able to assign doctors in the capacity of a doctor-on-duty role to a ward during specific time periods. In this role the system allows access to the health records of only those patients who are assigned to the doctor. The system also allows interns to attend in a ward for training purpose. The access permission will be valid until the doctor keeps the role of doctor-on-duty. This role is revoked when the doctor leaves the location, or after the end of her duty time. Doctors allow to create different kinds of diagnostic reports about patients so that an intern can access such reports only if the doctor is present in the ward. We consider a situation where an intern begins to access a report and during the access, the doctor leaves the ward. In such a case, the system needs a mechanism to ensure that the ongoing accessing should be revoked due to the absence of the doctor.

We now illustrate how the access control obligation for the scenario 1 can be modeled and enforced by using our approach. We assume that all doctors, nurses, interns and patients should register to the HIS system. Meanwhile, entrance doors of important places such as ward, operation room, and conference room have been equipped by RFID readers. Therefore, the system is able to track the individuals, because all personnels have to wear a badge embedded by an RFID tag. The requests of all registered users in different roles to access various resources are evaluated based on authorization rules defined in the policy repository. However, we utilize our approach to resolve some issues existing around fulfilling the obligations such as: how policy administrator can express the detail of the obligation and how policy engine enforces the obligation during access.

The obligation specifies that an intern can access diagnostic reports in the presence of a doctor. We use the BPL to express the behavior of the intern who is responsible to perform the obligation. In this case, there is a strong association between the behavior of intern and doctor based on the attribute of location. Code 1.1 illustrates the BPL code to represent the dependency of intern's behavior and doctor's behavior. We assume that the behavior of doctor and intern includes one event (line 6 and 7). The structure of the event is defined in line 4 so that it has six attributes. Line 8 defines the domain values for each attribute. We call two methods in line 9 and 10 to assign values to attributes of each event. In line 11, we emphasize that the occurrence of events should be in the same location.

When an intern sends a request, policy engine evaluates the request by applying policy rules and stateless conditions that are based on the information provided in the access request. The request will be allowed or denied based on the result of the evaluation. However, the authorization system must consider the location of doctor in the role of doctor-on-duty during the execution of the granted access. In this case, the location of the doctor-on-duty is a mutable attribute that presents stateful condition binding to the resource state. Therefore, continuous obligation enforcement is required by the authorization system.

According to Fig. 3, before starting the access session, B2BUA commences an obligation session between the intern and obligation authority (in this case electronic health record system). After establishment of the obligation session, B2BUA transmits the intern's request to the resource manager in order to create an access session. The values of mutable attributes of involved entities will be periodically collected by Policy Information Point (PIP). Once the location of the doctor changes, an "End_Obligation" message will be generated. Then, B2BUA sends a "Bye" message to both obligation authority and resource manager to terminate the obligation session and access session respectively.

Code 1.1. The BPL description of the obligatory behavior of the intern in scenario 1

```
PATTERN Intern_Behavior;
TYPE ongoing-obligation;
BEGIN
  DEFINE Event = <User,Role,Action,Resource,Location,Time>
  E[] = new Event(2);
  set1 = {E[1]};
  set2 = {E[1]};
  attVal = {<Role,{"intern", "doctor"}>,<Action,{"read","
      write"}>,<Location,{"ward","operating room" }>,<
      Resource,{"diagnostic report"}>};
  CALL Op1(set1,attVal);
  CALL Op2(set2,attVal);
  ASSERT EQUALITY(set1,set2,Location);
END

/* define operation 1-Behavior of intern*/
   DEF Op1 (set,attVal)
   Set[E1].Role=attVal[Role].value[0];
     Set[E1].Action=attVal[Action].value[0];
     Set[E1].Resource=attVal[Resource].value[0];
     Set[E1].Resource=attVal[Location].value[0];
   END
/* define operation 2-Behavior of doctor*/
   DEF Op2 (set,attVal)
     Set[E1].Role=attVal[Role].value[1];
     Set[E1].Resource=attVal[Resource].value[0];
   END
```

Scenario 2: We consider the case of two individuals, say Bob and Alice, who make a video conversation using free softphone applications. When the caller initiates a conversation, the application requires that an advertisement window is displayed on the caller's device while the call is in progress. The caller have to keep the advertisement window active during the conversation. Consequently, if the caller closes this window while the call is still in progress, the application policy term is violated and caller's access should be revoked.

Obligation in scenario 2 represents the typical example of a usage control that obliges the user to open and watch an advertisement during using a multimedia service. Code 1.2 expresses the mandatory behavior of a caller during using the calling service. Execution of the obligation occurs in parallel with using the calling service. The procedure to handle the obligation follows the process shown in Fig. 3. The caller sends an "Invite" massage (request) to set up a conversation. The B2BUA receives the decision along with the detail of the obligation from the PDP. Since the obligation is an ongoing obligation, the B2BUA creates an obligation session between the caller and the obligation authority (in this case an advertisement server). The obligation session should keep alive during the conversion between caller and callee. After establishment of the obligation

session, the B2BUA allows to create an access session between caller and callee to use the service. If the caller closes the advertisement window for any reason, the B2BUA realizes the termination of watching the advertisement and sends a "Bye" message to the obligation authority to stop obligation session, followed by a "Bye" message to callee to terminate the access session.

Code 1.2. The BPL description of the obligatory behavior of the intern in scenario 2

```
1  PATTERN Caller_Behavior;
2  TYPE ongoing-obligation;
3  BEGIN
4     DEFINE Event = <User,Role,Action,Resource,Time>
5     E[] = new Event (3);
6     set = {E[1],E[2],E[3]};
7     attVal = {<Role,{"caller","calee"}>,<Action,{"call","
          open","watch"}>,<Resource,{"calling_service","
          advertizement"}>};
8     CALL Op1(set,attVal);
9  END
10
11 /* define operation */
12    DEF Op1 (set,attVal)
13       Set[E1].Role=attVal[Role].value[0];
14       Set[E1].Action=attVal[Action].value[0];
15       Set[E1].Resource=attVal[Resource].value[0];
16
17       Set[E2].Role=attVal[Role].value[0];
18       Set[E2].Action=attVal[Action].value[1];
19       Set[E2].Resource=attVal[Resource].value[1];
20
21       Set[E3].Role=attVal[Role].value[0];
22       Set[E3].Action=attVal[Action].value[2];
23       Set[E3].Resource=attVal[Resource].value[1];
24
25       ASSERT EQUALITY DURATION(E1,E3);
26    END
```

5 Conclusion

This paper addresses the fulfilment of access control obligation during the access to the particular resources. Handling different kinds of obligations goes beyond the capabilities of traditional access control systems. Due to the dynamic nature of pervasive computing entities behavior, protecting resources only by predefining a set of authorization rules is extremely challenging. Moreover, pervasive computing often relies on the heterogeneous environment that exacerbates the incompleteness of access control policy rules. Considering the obligation as a part of authorization policies increases the capability of the access control system to mitigate the misuse of resources. Therefore, obligations are required to force the

user to follow specific behaviors in order to access securely the resources and services in pervasive environments.

A model is required to present the obligation in order to dictate the mandatory operations that must be performed by the user before, after, and during the access. Furthermore, due to the incomplete nature of policy languages to cover all aspects of access needs, they cannot specify expressively the features of obligations. Therefore, we first defined a behavior as a sequence of events along with several constraints that show the correlation between events. Then, we modeled the obligation based on the obligatory behavior of a subject in access control system. We also introduced a behavior based language to express the structure and semantics of the mandatory behavior as an obligation.

From the continuity perspective, the fulfillment of ongoing obligation has the main effect on the continuity of the granted access. In the XACML workflow, there is no information for obligation enforcement and the role of obligation for the continuity of granted access. Consequently, we leveraged the capabilities of SIP protocol to handle the obligation. We have developed a structure for obligation service based on XACML reference structure and SIP entities for the purpose of continuity or revocation of granted access.

We aim at extending our approach by provisioning a behavior analysis and testing model in order to find the instances of the user's behavior leading to an obligation violation. Such an extension helps us to cover the fulfillment of the post obligation that cannot be approved appropriately by our current approach due to the durability issue. Meanwhile, PEP needs to monitor the actions of the user to verify the violation of the ongoing obligation. For this purpose, utilizing the complex event processing technology to capture the dynamic nature of user's actions is our next step to accomplish the work.

References

1. eXtensible Access Control Markup Language (XACML), version 3.0, OASIS standard, January 2013. https://www.oasis-open.org/
2. Ardagna, C.A., Cremonini, M., Damiani, E., di Vimercati, S.D.C., Samarati, P.: Supporting location-based conditions in access control policies. In: ACM Symposium on Information, Computer and Communications Security, pp. 212–222. ACM (2006)
3. Cirani, S., Picone, M., Veltri, L.: A session initiation protocol for the internet of things. Sci. Int. J. Parallel Distrib. Comput. Scalable Comput. Pract. Experience **14**(4), 249–263 (2015). SCPE
4. Damiani, M.L., Bertino, E., Silvestri, C.: An approach to supporting continuity of usage location-based access control. In: 12th IEEE International Workshop on Future Trends of Distributed Computing Systems, pp. 199–205. IEEE (2008)
5. Elrakaiby, Y., Cuppens, F., Cuppens-Boulahia, N.: Formal enforcement and management of obligation policies. Data Knowl. Eng. **71**(1), 127–147 (2012)
6. Feltus, C., Petit, M., Sloman, M.: Enhancement of business IT alignment by including responsibility components in RBAC. In: CAiSE 2010 Workshop Busital 10, Hammamet, Tunisia, pp. 61–75 (2010)

7. Gomez, L., Trabelsi, S.: Obligation based access control. In: Meersman, R. (ed.) On the Move to Meaningful Internet Systems: OTM 2014 Workshops. LNCS, vol. 8842, pp. 108–116. Springer, Heidelberg (2014). doi:10.1007/978-3-662-45550-0_15
8. Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., et al.: SIP: Session initiation protocol. RFC 3261 (2002). https://www.ietf.org/rfc/rfc3261.txt
9. Karopoulos, G., Mori, P., Martinelli, F.: Continuous authorizations in SIP with usage control. In: 20th Euromicro International Conference on Parallel, Distributed and Network-based Processing, pp. 283–287. IEEE (2012)
10. Karopoulos, G., Mori, P., Martinelli, F.: Usage control in SIP-based multimedia delivery. Comput. Secur. **39**, 406–418 (2013). Elsevier
11. Katt, B., Zhang, X., Breu, R., Hafner, M., Seifert, J.P.: A general obligation model and continuity-enhanced policy enforcement engine for usage control. In: The 13th ACM symposium on Access Control Models and Technologies, pp. 123–132. ACM (2008)
12. Kulkarni, D., Tripathi, A.: Context-aware role-based access control in pervasive computing systems. In: 13th ACM symposium on Access Control Models and Technologies, pp. 113–122. ACM (2008)
13. Li, N., Chen, H., Bertino, E.: On practical specification and enforcement of obligations. In: The second ACM conference on Data and Application Security and Privacy, pp. 71–82. ACM (2012)
14. Liscano, R., Dersingh, A., Jost, A.G., Hu, H.: Discovering and managing access to private services in collaborative sessions. IEEE Trans. Syst. Man Cybern. Part A: Syst. Hum. **36**(6), 1086–1097 (2006). IEEE
15. Miskovic, V., Babic, D.: An architecture for pervasive healthcare system based on the IP multimedia subsystem and body sensor network. Facta Univ. Ser. Electron. Energetics **28**(3), 439–456 (2015)
16. Park, J., Sandhu, R.: The UCON$_{ABC}$ usage control model. ACM Trans. Inf. Syst. Secur. **7**(1), 128–174 (2004)
17. Ray, I., Toahchoodee, M.: A spatio-temporal role-based access control model. In: Barker, S., Ahn, G.-J. (eds.) DBSec 2007. LNCS, vol. 4602, pp. 211–226. Springer, Heidelberg (2007). doi:10.1007/978-3-540-73538-0_16
18. Sampemane, G., Naldur, P., Campbellg, R.H.: Access control for active spaces. In: 18th Annual Computer Security Applications Conference. ACM (2002)
19. Schulzrinne, H., Wedlund, E.: Application-layer mobility using SIP. Mob. Comput. Commun. Rev. **4**(3), 47–57 (2000). ACM
20. Sharghi, H., Sartipi, K.: An expressive event-based language for representing user behavior patterns. J. Intell. Inf. Syst. 1–25 (2017). doi:10.1007/s10844-017-0456-5
21. Strembeck, M., Neumann, G.: An integrated approach to engineer and enforce context constraints in RBAC environments. ACM Trans. Inf. Syst. Secur. **7**(3), 392–427 (2004). ACM
22. Toahchoodee, M.: Access control models for pervasive computing environments. Ph.D. thesis, Colorado State University, Fort Collins, Colorado (2010)
23. Toahchoodee, M., Abdunabi, R., Ray, I., Ray, I.: A trust-based access control model for pervasive computing applications. In: Gudes, E., Vaidya, J. (eds.) DBSec 2009. LNCS, vol. 5645, pp. 307–314. Springer, Heidelberg (2009). doi:10.1007/978-3-642-03007-9_22
24. Ulltveit-Moe, N., Oleshchuk, V.: Enforcing mobile security with location-aware role-based access control. Secur. Commun. Netw. **9**(5), 429–439 (2016). Wiley
25. Zhang, X., Parisi-Presicce, F., Sandhu, R., Park, J.: Formal model and policy specification of usage control. ACM Trans. Inf. Syst. Secur. **8**(4), 351–387 (2005)



http://www.springer.com/978-3-319-59040-0

E-Technologies: Embracing the Internet of Things
7th International Conference, MCETECH 2017, Ottawa, ON, Canada, May 17-19, 2017, Proceedings
Aïmeur, E.; Ruhi, U.; Weiss, M. (Eds.)
2017, XII, 319 p. 80 illus., Softcover
ISBN: 978-3-319-59040-0