A User Data Location Control Model for Cloud Services

Kaniz Fatema[1], Philip D. Healy[1], Vincent C. Emeakaroha[1], John P. Morrison[1] and Theo Lynn[2]
[1]*Irish Centre for Cloud Computing & Commerce, University College Cork, Cork, Ireland*
[2]*Irish Centre for Cloud Computing & Commerce, Dublin City University, Dublin, Ireland*

Keywords: Authorization System, Access Control, Data Location, XACML, Cloud Computing.

Abstract: A data location control model for Cloud services is presented that uses an authorization system as its core control element. The model is intended for use by enterprises that collect personal data from end users that can potentially be stored and processed at multiple geographic locations. By adhering to the model's authorization decisions, the enterprise can address end users' concerns about the location of their data by incorporating their preferences about the location of their personal data into an authorization policy. The model also ensures that the end users have visibility into the location of their data and are informed when the location of their data changes. A prototype of the model has been implemented that provides the data owner with an interface that allows their location preferences to be expressed. These preferences are stored internally as XACML policy documents. Thereafter, movements or remote duplications of the data must be authorized by submitting requests to an ISO/IEC 10181-3:1996 compliant policy enforcement point. End users can, at any time, view up-to-date information on the locations where their data is stored via a web interface. Furthermore, XACML obligations are used to ensure that end users are informed whenever the location of their data changes.

1 INTRODUCTION

Cloud Computing offers a new style of computing that allows consumers to pay only for the services used and frees them from the management overhead of the underlying infrastructure. Although Cloud Computing has gained significant traction in recent years, surveys have consistently shown that consumers' concerns around security and loss of control over data are hindering adoption (Subashini and Kavitha, 2011; Chen and Zhao, 2012). Additionally, the physical location of data can have an impact on its vulnerability to disclosure and can have implications for service quality and legal consequences (Albeshri et al., 2012; Gondree and Peterson, 2013). Many regulations such as HIPAA and the EU Data Protection Directive impose restrictions on the movement of data between geographical locations. Data location therefore represents a sensitive issue for enterprises that offer cloud services. Existing and potential customers seek assurance that the enterprise will act as faithful stewards of information entrusted to them, and the enterprise itself wishes to avoid falling foul of data protection rules and other regulations.

However, these concerns must be balanced against the enterprise's desire to maintain redundancy and operational efficiency. There are a number of valid reasons for copying data to geographically dispersed locations. These may include:

- *Risk Mitigation* Copying data to more than one physical location provides a hedge against localised catastrophic events such as fires and natural disasters.
- *Operational Expenditure* Due to the geographically variable nature of overheads, such as energy costs, it may be cheaper to store and/or process data at a location other than where it was stored originally.
- *Storage Capacity* If a data centre has limited capacity it might be helpful to offload some storage to another location.
- *Maintenance* Data may sometimes need to be moved temporarily in order to facilitate data centre maintenance, upgrade or relocation.
- *Localised Caching* Content delivery networks, such as Akamai and Amazon CloudFront, replicate data to edge servers in order to improve users' quality of experience.

Enterprises that offer Cloud services must therefore balance the benefits of migrating data against concerns relating to trustworthiness in the eyes of users and regulatory compliance.

Users' concerns about how their data is disseminated and used can be addressed by allowing them to specify policies about how they wish their data to be used and assuring them of the enforcement of this policy. In the context of data location, this assurance can be provided in two ways (a) with a transparency mechanism that allows users to view the locations where their data is being stored and (b) with a notification mechanism that informs them when there is a change to the *status quo*. If the users' policies are to carry any weight then they must be consulted before operations are performed that would result in movement or remote duplication so that violations can be avoided.

A means of addressing this challenge is to implement an authorization system that inputs queries relating to data relocation and returns results that indicate the permissibility of the proposed actions. Of course, such a system is of little value unless it is appropriately integrated into the service provider's internal processes and mechanisms. As such, there is a residual trustworthiness issue in that the users must trust that this integration has taken place and that the decisions of the authorization system are being honoured. These issues could be addressed through independent verification mechanisms such as trustmarks, audits and assurance services (Lynn et al., 2013).

Although the basic concept of an authorization system (di Vimercati et al., 2005; ISO, 1996) is to protect access to secured resources, the functionality of existing authorization systems can be extended to perform other operations such as enforcing obligations (Park and Sandhu, 2004; Demchenko et al., 2008) and verifying credentials (Chadwick et al., 2008). In this paper, we propose a Cloud data location assurance control model that is based on authorization systems. The ISO/IEC 10181-3:1996 standard access control framework is used as the technological foundation. The following issues are addressed:

1. Capturing users' location preferences as policies that can be automatically consulted
2. Providing an authorization mechanism for data movement decisions that takes user-specific policies into account
3. Providing users with visibility into the location of their data
4. Ensuring that users are informed when the location of their data changes

The remainder of this paper is organized as follows: Section 2 presents an analysis of related work. Section 3 provides background information on authorization systems, XACML policies and their evaluation process and how they can play a role in Cloud data management. Section 4 presents the design of the proposed location control model while Section 5 provides some technical details along with the result of validation tests. Finally, Section 6 concludes the paper and suggests future research directions.

2 RELATED WORK

Geolocation is a technique that can be used *post facto* to determine where data is being served from, and hence whether the data is being hosted from an undesirable location. Most Geolocation techniques make use of network latency as a proxy for distance (Katz-Bassett et al., 2006). The techniques include: *geo ping*, in which a server is pinged from a number of landmarks to identify how close the server is to each landmark and hence determine the location; *constraint-based geolocation*, which uses a triangulation technique to determine the position of a server that is possibly between landmarks; and *topology-based geolocation*, which considers trace route, topology and per hop latency. These techniques are used to estimate the geolocation of a Cloud data centre based on network latency and comparing it using heuristics to expected latencies from known distances (Benson et al., 2011; Ries et al., 2011; Peterson et al., 2011). Ries *et al.* proposed a geolocation approach based on network coordinate systems to identify location policy violations (Ries et al., 2011). They found the accuracy of the geolocation-based approaches to be only 52% at best. Albeshri *et al.* proposed an architecture for data geolocation based on a Proof of Storage (POS) protocol and timing-based distance bounding protocol (Albeshri et al., 2012). The POS protocol is used to provide proof to the client that the server is holding the data and that it has not been modified. The delay in response is used to determine the probable location of the data. A similar approach has been proposed (Peterson et al., 2011; Gondree and Peterson, 2013) that uses a challenge-response protocol based provable data possession technique combined with a geolocation-based solution. However, determining the location of data based on response time has been shown to be inaccurate (Ries et al., 2011). Given the inaccuracy and *post facto* nature of geolocation techniques, we decided against using this approach for our purposes. However, geolocation could have a role as an independent audit mechanism for confirming that users' location preferences are being honoured.

The Data Location Assurance Service proposed by Noman *et al.* is based on an approach where the Cloud service provider sends regular updates to enterprises about the location of their data (Noman and Adams, 2012). The enterprise in turn provides location assurance to users by providing information on whether the data are in their preferred location in a simple yes/no format. This approach relies heavily on the trustworthiness of the Cloud service provider. Massonet *et al.* describe how negotiation can be used to integrate service providers' policies with those of infrastructure providers in a federated Cloud environment (Massonet et al., 2011). The infrastucture provider's security policy states the availability of logging facilies, logging type and monitoring, while the servuce provider's policy states its security requirements. A virtual machine is migrated to a site only if its security policy matches with the security requirements of the service provider. An audit log is created from the collaboration of between service provider and infrastructure provider. The location of data is identified through the presence of data centre's certificate in the audit log. Neither of these systems allow for detailed location preferences, such as multiple preferred geographic regions, to be collected from end users and enforced automatically.

Numerous implementations of policy-based authorization have been proposed in order to protect access to sensitive data in the Cloud (Iskander et al., 2011; Basescu et al., 2011; Chadwick and Fatema, 2012). A distributed authorization system architecture for Cloud services was proposed by Almutairi *et al.* where each Cloud layer has a virtual resource manager that protects access to the virtualized resources of its layer using an access control module (Almutairi et al., 2012). Ries *et al.* proposed a policy-based approach to location issues (Ries et al., 2011). However, they did not elaborate on how to incorporate location into policy or how to enforce policies.

A number of policy languages – such as P3P (Cranor, 2003), EPAL (Ashley et al., 2003), PERMIS (Chadwick et al., 2008), FlexDDPL (Spillner and Schill, 2012) and XACML (Godik et al., 2002) – have become available for various purposes. P3P (Cranor, 2003) defines a machine interpretable format for websites to express their privacy practices. The notice and consent model of P3P allows websites to describe their privacy policies and users can read these policies and can choose to interact with the websites and thus provide their consent to the policies. P3P provides: human readable versions of the policies; automated comparison of users preferences with policies; automated reports in the event of a mismatch; and incorporation of users preferences when viewing and changing policies. A drawback of P3P is that it simply checks whether the users preferences match with the organisations stated privacy policies; it does not ensure that the organisations actually enforce their stated privacy policies. FlexDDPL defines a policy language for data distribution which specifies who under what conditions is allowed to store what data, how and where. FlexDDPL policies can be expressed independently from storage controller and gateway implementations and be mapped onto those as part of administration process. However, how these two independent policies can be combined and how the conflicts between those can be resolved is not defined. Moreover their current implementation does not include location.

On the other hand, XACML (an OASIS standard) provides both a policy language and an access control decision request/response language and also specifies a policy evaluation engine. Chadwick *et al.* have modified the XACML model to allow multiple policy languages and multiple policies of independent authorities to be supported by the model, making it suitable for use in Cloud scenarios (Chadwick and Fatema, 2012). The current implementation of it supports both XACML and PERMIS policy languages and is available under an open source license. However it still uses the XACML request and response language and keeps the policy decision points (see below) for each policy language unchanged. We chose to adopt XACML for our location control model due to its standardised format and built-in enforcement mechanism. It is also the most widely used language for defining security policy (Turkmen and Crispo, 2008).

3 BACKGROUND INFORMATION

In this section, we present some background information on authorization systems and how they can be integrated with Cloud deployments. Furthermore, we describe a use case model that highlights the usability of the proposed Cloud data location control mechanism. We also specify the scope and constraints of our solution.

3.1 Authorization Systems

Access control/authorization is a process of determining whether a request to access (*e.g.* read, write, delete, copy, etc.) a resource object (*e.g.* file, database, program, system component) by a subject (*e.g.* user, system or process) should be granted or denied. The access control/authorization system can grant or deny the request based on whether or not certain policy constrains have been satisfied. ISO/IEC

10181- 3:1996 (ISO, 1996) defines a generic Access Control Framework (ACF) as shown in Figure 1. It consists of four components: (i) initiators (subjects), (ii) targets (resource objects), (iii) Access control Enforcement Functions (AEFs) – commonly known as Policy Enforcement Points (PEPs) – and (iv) Access control Decision Functions (ADFs) – commonly known as Policy Decision Points (PDPs). The initiators submit access requests (also known as user requests) that specify the operation to be performed on the target. The AEF transforms the request into one or more decision requests (also known as authorization queries) and sends these to the ADF. The ADF decides whether a decision request should be granted or denied based on the provided policies and sends the decision back to the AEF.

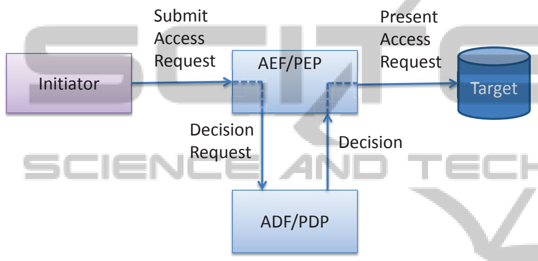

Figure 1: ISO/IEC 10181-3 access control framework (ACF).

An access decision can contain obligations, such as sending e-mail to the data subject or recording the permitted access in a log. These obligations are enforced by the PEP while executing the access decision. Authorization systems may play an important role in Cloud data management since Cloud service delivery can involve significant amounts of data transmission and storage. SaaS providers offering services, such as web applications, may accumulate a large amount of personal data like names, addresses, medical records, purchase history and so on during the execution of the service. The management, processing and storage of such data while considering customers' data location choices is challenging. In this paper, we aim to develop a mechanism for SaaS providers, who might be using services from IaaS and PaaS layers, to incorporate the location choices of their customers while managing and storing data in Clouds.

3.2 XACML Policy Structure and Evaluation

This section provides a brief description of XACML policy structure and its evaluation strategy.

3.2.1 XACML Policy Structure

XACML policies are constructed with the following components: policy set (PS), policy (P), rule (R), target (t), policy combining algorithm (PCA) and rule combining algorithm (RCA). A target t is composed of four types of attributes: subjects (S), resources (Rs), actions (A) and environments (En). A rule is defined as a tuple, $r = (id, t, e, c)$, where id is a rule ID, t is a rule target, $e \in \{\text{Permit}, \text{Deny}\}$ is an effect and c is a condition which is an optional element and is evaluated to a Boolean value. A policy is defined as a tuple, $p = (id, t, R, RCA, O)$, where id is a policy ID, t is a policy target, $R = \{r_1, r_2, \ldots, r_n\}$ is the set of rules and RCA is the rule combining algorithm and O is an optional set of obligations.

A policy set is defined as tuple $PS = (id, t, P, PAC, O)$, where id is the policy set ID, t is the policy set target, $P = \{p_1, p_2, \ldots, p_n\}$ is the set of policies, PCA is the policy combining algorithm and O is an optional set of obligations. An access request in XACML v2 is defined as a set of attributes, $R_q = (s, rs, a, en)$, where s is a set of subject attributes, rs is a set of resource attributes, a is a set of action attributes and en is a set of environment or context attributes. Each attribute set describes the corresponding object. The request means that the subject described by s is asking to be perform the action described by a on the resource described by rs in the presence of the context described by en.

3.2.2 Evaluation of a Request

An authorisation decision is performed by the XACML PDP by matching the values of the request with the values in the policies. A *target* element of XACML is evaluated as Match if all the elements specified in the target match with the values presented in the request context. If any one of the elements specified in the *target* is *indeterminate*, then the *target* is evaluated as *indeterminate*. Otherwise, the *target* is evaluated as *no-match*. The *condition* element represents a Boolean expression. When the *target* element of an XACML policy evaluates to Match and the *condition* element evaluates to True, the *rule* evaluates to Effect and when the *target* element of an XACML policy evaluates to Match and the *condition* element evaluates to False the rule evaluates to NotApplicable (Anderson, 2005). The standard policy combining algorithms of XACML v2 (Anderson, 2005) and v3 (Rissanen, 2013) are defined as:

1. Deny-overrides (Ordered and Unordered in v3),
2. Permit-overrides (Ordered and Unordered in v3),
3. First-applicable and

4. Only-one-applicable.

In the case of the deny-overrides algorithm, if a single policy element evaluates to “Deny”, then, regardless of the evaluation result of the other policy elements, the combined result is “Deny”. For ordered deny-overrides the behaviour of the algorithm is the same except that the order in which the collection of policies is evaluated will match the order as listed in the policy set. Similarly, in the case of the permit-overrides algorithm, if a single “Permit” result is encountered, regardless of the evaluation result of the other policy elements, the combined result is “Permit” and for the ordered permit-overrides the behaviour of the algorithm is the same except that the order in which the collection of policies is evaluated will match the order as listed in the policy set. In the case of the “First-applicable” combining algorithm, the first decision encountered by the policy element in the list becomes the final decision accompanied by its obligations (if any). The only-one-applicable policy-combining algorithm ensures that only one policy is applicable by virtue of its target. The result of the combining algorithm is the result of evaluating the single applicable policy. If more than one policy is applicable, then the result is “Indeterminate”. In the XACML v3 specification some other combining algorithms are also defined, such as deny-unless-permit (which returns “Deny” only if no “Permit” result is encountered) and permit-unless-deny (which returns “Permit only” if no “Deny” result is encountered).

3.3 Use Case

To contextualize our approach in this paper, we present a use case that exemplifies the scope of our approach. Figure 2 shows an overview of our system model and how it integrates with Cloud deployments. The model considers interactions among three different groups of stakeholders: (i) Cloud service providers, (ii) enterprises and (iii) end users.

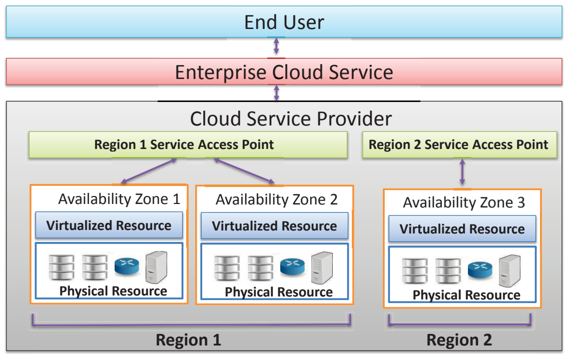

Figure 2: Location aware Cloud usage model.

Cloud Service Providers – For the purposes of this use case, we assume that the Cloud service providers are either IaaS or PaaS providers who offer virtualized resources/environment to the enterprises as services. The services are offered in separate regions, as shown in Figure 2. They deploy resource instances and data according to the instructions of their customers and do not move data without being instructed by the customers. For example, Amazon EC2 provisioning currently works in this manner. Each region is identified by a unique name and is accessed via specified access points that act as a gateways to the service. Each region may consist of a number of availability zones, as shown in Figure 2, where the instances and data can be copied to provide better availability in the event of a failure. Customers can choose one or more availability zones where their data and service instances can be replicated.

Enterprises are the customers of IaaS and PaaS services. They compose their SaaS services over the IaaS or PaaS services to offer them to the end users. These services may be designed to collect and process the end users’ personal data. Depending on their organizational needs, an enterprise can choose to put their services in more than one regions and a number of availability zones in each region. Enterprises manage the customers’ data and are responsible for instructing the Cloud providers correctly about which data is to be stored in which region.

End users/data owners – End users are the persons who are consuming the service. Enterprises may collect personal data from end users which are later stored and processed in the Cloud environment. The end users who are sharing their personal data with the enterprise are referred to as *data owners*.

This model assumes data to be of any format which should have two properties: i) it should be identified by one uniquely identifiable name, *DataID*, and ii) it is expected to be treated as one unit while actions (*e.g.* copy, read, write) are performed on it. The idea of having a unique identifier within a Cloud is not an impractical assumption as each Cloud user has a unique account which can be used or combined to form the unique identifier. In federated scenarios involving multiple participants a number of Cloud and identifier linking mechanisms can be used to create a globally unique identifier.

The granularity of data that is protected by the policy depends on the enterprise’s application scenario. It can be applied to a file or a set of files grouped under one folder, or a database entry. The unique identifier of the data, DataID, and the identifier of the policy, PolicyID, that is applicable for protecting the data is linked together by the model. This linking will allow the policy that is relevant for that data to be evaluated while a request for performing an action (*e.g.* copy, read, write) on a data is received. As policy is linked to a DataID regardless of the format of data the model will protect the access of any type of data based on the relevant policy.

We use a loyalty card program, as offered by many retail chains, to exemplify this use case. During the sign up process, the customer gives some personal information such as her name, address and contact information (e-mail/telephone number). The customer uses the card while purchasing products from the retailer and may be offered discounts based on the purchase history. Each loyalty card can be represented electronically as an XML infoset containing a unique Loyalty Card number (which can be used as DataID)and other elements, such as a personal information element containing name, address, age; and a purchase history element, which can grow over time based on the number of purchases. To improve availability and quality of experience for their customers, the retailers use multiple Cloud services with different location and regions. The retailer accesses the services and data in these regions via the access points offered by the Cloud Service provider.

The data owners are given the opportunity by the retailer to choose one of the locations as a primary location for storing their data initially. They can also choose other locations where their data can be moved if necessary. They can also activate an option to receive e-mail when the location changes. The retailer ensures that the Cloud providers deploy the data to the correct region according to the preference of the data owner. In case of data movement, the data owner receives a notification if required by her policy.

4 DATA LOCATION CONTROL

The core of our approach is an enterprise Cloud service manager (ECSM) that manages access to all Cloud data and services using an authorization system. The authorization system evaluates two types of policies for making an access decision upon an access request. The first is the organizational administrative policy that specifies the enterprise roles and their attributed actions on services or data (*e.g.*, a marketing officer can execute a service, read the data and so on) or the roles of other external services to perform actions on their protected data. This administrative policy is executed for all access requests. The second type of policy consulted is the data owner's specific policy. All the data owners' policies are stored in the data owners' policy store.

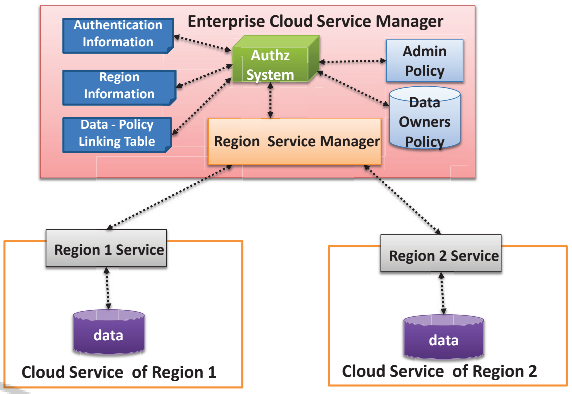
Enterprise Cloud Service Manager
Authentication Information
Region Information
Data - Policy Linking Table
Authz System
Admin Policy
Data Owners Policy
Region Service Manager
Region 1 Service
data
Cloud Service of Region 1
Region 2 Service
data
Cloud Service of Region 2

Figure 3: The enterprise Cloud service manager (ECSM) manages services and data in different Cloud regions.

The enterprise service manager also includes a user authentication information table for managing user authentications, a region information table that details the regions where each data set identified by a DataID is stored and a data-policy linking table that correlates each PolicyID to a DataID in order to retrieve the correct policy of the data owner for a specific data set. Figure 3 depicts how the region service manager receives instructions from the authorization system on where to store or retrieve data.

4.1 Intra-service Interaction Strategy

This section shows the interaction strategy to manage data locations within the same enterprise. The discussion is partitioned into four catagories: inputting data, accessing data, updating data location and checking location.

Data Input – To use the enterprise Cloud service, the end users first need to register for it. During the registration process, the user specifies authentication information (typically a user name and password or credentials) that is stored by the service provider to identify the user. The authentication information provided by each user is associated with a *DataID* that identifies the data of this user. In our example use case scenario it would be the loyalty card number. The user provides a policy expressing her preferred primary location for storing the data. It also specifies alternative locations where data can be moved for different reasons, for example, to achieve better performance or cheaper cost. The data owner's policy is provided with a PolicyID, which is also linked to the DataID in the data-policy linking table as shown in Figure 3. Each DataID is linked to a region information table that contains the current locations of the data identified by the DataID. This information is updated whenever the data location changes.

Accessing Data – All the requests for accessing data (*e.g.* reading the data or processing the data) goes to the ECSM's authorization system. The ECSM first gets the data owner's PolicyID in the data-policy linking table and retrieves the policy from the data owners' policy store. The administrative policy and the data owner's policy are evaluated against the access request. If the policies evaluations are successful, the requested access is granted. The enterprise authorization system can be configured with various static conflict resolution strategies (Godik et al., 2002) or can obtain the conflict resolution strategy dynamically (Mohan and Blough, 2010) to resolve the conflicts between the administrative policy and data owner's policy.

Updating Data Location – The enterprise Cloud service administrator continuously monitors the performance of the services. If there is a need to copy the data to another location, she places a permission request to the ECSM authorization system. The ECSM accesses the data owner's policy and the administrative policy to evaluate them as described previously. If the policy allows for copying to the new location, then the data are copied and the region information table is updated to reflect these changes. If the user requires notification upon data location change, then the authorization system enforces this through the obligation enforcer. If the copy request is rejected then the data location remains unchanged.

Checking Location – When a user wants to query the location of her data, she first logs in to the service with her access credentials so that the service can identify the data she owns. To allow the user access the region information, either the administrator policy or the data owner policy has to allow it as the authorization system evaluates these policies to determine the access right. If any of the policies return "Grant" then the user is given the resquested information.

4.2 Inter-cloud Data Outsourcing

To achieve better efficiency and scalability, enterprises may need to outsource the processing of end users' data. For example, the retailer in our use case scenario might outsource the processing of the purchasing history to an analytics service. To realise this strategy, the enterprise maintains an external cloud information table to store the information of the external Cloud services with copies of the the data for each DataID. This table is linked with the region information table that contains the information of regions where the enterprise host its own service and originally stores the data, as seen in Figure 4.

The enterprise's administrative policy expresses which external services are allowed to access their protected data and each data owner's policy specifies the allowed locations where his/her data can be copied. When an external service requests permission from the ECSM's authorization system to transferring data to a certain location of their Cloud, the administrative policy along with each data owner's policy are evaluated. If the administrative policy or the data owner policy does not allow data transfer to the specified locations, then the permission is denied. If the request is accepted, the data along with the policy of the data owner are sent to the external service. A Service Level Agreement (SLA) negotiation process may take place between the enterprise and the external service provider in order to ensure that the data owner policy is respected upon any access request to the data. The external service provider may use the same PolicyID and DataID or may assign new IDs according to its organizational format. When assigning new IDs, there must be a link to the old ones for traceability. The external service provider maintains the outsourced data location similar to the approach described in Section 4.1. However, in this case access is granted based on the data owner's policy since the external service provider might have a different administrative policy.

A scheme similar to that for viewing location information as mentioned in Section 4.1 is provided to allow data owners to view their data location entries in the external cloud information table. To view the data location information, there are two options. The end user (data owner) can access this information through the default enterprise originally hosting her data. The default enterprise is then responsible for acquiring this information from the external service provider. The external service provider's administrative policy can be written in a way to allow this access. This can be specified in the SLA document with the external service provider. Alternatively, since the external service provider evaluates the data owners policy upon receipt of an access request for a user's data, the data owner's policy can be used to allow the necessary access to the data location information. Figure 4 shows how data location can be controlled by the ECSM's authorization systems for two different service managers.

5 IMPLEMENTATION

This section presents the technical details required for a practical implementation of the proposed location control model. This section also includes a description of a prototype implementation and testbed that used to perform validation tests of the location control model.

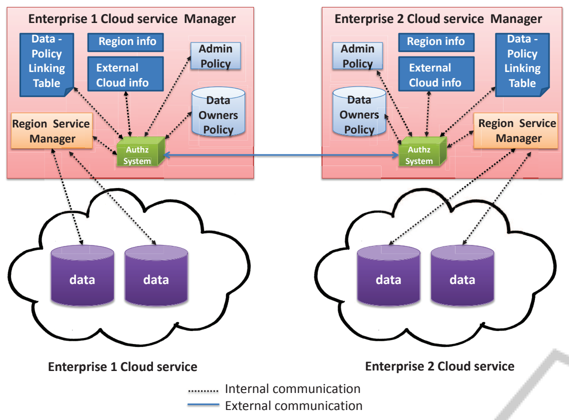

Figure 4: Controlling data location across multiple Clouds.

5.1 Data Presentation

Our model assumes that each end user's data is uniquely identified by a DataID. For our loyalty card use case scenario, XML is chosen as the representation format as it allows the data to be easily structured in separate elements which can be easily identified, modified and extended. However, alternate representation schemes, such as database tables or NoSQL entries could also be used. An example of the loyalty card XML format is given in Listing 1, which consists of two main elements: *PersonalInfo*, describing the personal identification information for a customer, and *PurchaseHistory* which encapsulates a list of the customer's purchases. Additional elements could be added based on the details of the specific loyalty card programme offered by the enterprise.

Listing 1: Loyalty Card Example.

```
<LoyaltyCard number="012345">
 <PersonalInfo>
    <Name>A B</name>
    <Address>IR</Address>
    <Age>30</Age>
    <Sex>Female</Sex>
 </PersonalInfo>
<PurchaseHistory>
  <Purchase>
    <Product>
      <ProductName>Bread</ProductName>
      <Brand>Acme</Brand>
      <Amount>3</Amount>
      <Price>15.50</Price>
      <DateOfPurchase>23.3.2013</
          DateOfPurchase>
      <PurchaseTime>15:00</PurchaseTime>
      <PurchaseStore>Cork</PurchaseStore>
    </Product>
  </Purchase>
</PurchaseHistory>
```

5.2 Policy Creation

Although the policies expressed as XML are human readable to an extent but are not particularly user-friendly, and it is unreasonable to expect end users to be able to write policies with these policy languages. Therefore, some form of user-friendly interface and translation mechanism is required. In our model, the data owner is presented with a web interface (implemented with PHP) for indicating location preferences. The interface works like a template for policy; the policy document is created automatically based on the locations that the user selects. The prototype implementation of the interface can be found in online[1]. After the customer's location preferences have been entered the resulting XACML request context containing user's policy (as PolicyContents element) is generated. This request context is then sent to the authorization system via a SOAP call. Depending on the enterprise's application scenario a default policy can be created when the user does not choose any location. In our prototype implementation the end user has to choose at least one location where she data to be stored initially and if the end user does not choose any optional locations then the created policy contains only the initial location.

The XACML policy language is used for the internal representation of policies as it is an OASIS standard and is the most widely used language for defining security policy (Turkmen and Crispo, 2008). XACML allows various types of attributes to be expressed, such as *Subject*, *Resource*, *Action* and *Environment*. The attributes presented in a policy are matched against the attributes presented in a request context by the PDP (see Section 3) when making an authorization decision. We have defined *Location* as an Environment attribute in our XACML policy which requires the request context to specify an appropriate value to be evaluated by the policy. An example of a policy document derived from a data owner's input is presented in Listing 2, allowing store/copy/transfer to Region 1 and Region 2 and obligating that an e-mail be sent to him/her when the location changes.

Listing 2: Data owner's policy example.

```
<PolicySet  PolicyCombiningAlgId="deny-
    overrides">
<Policy PolicyId="UserAccessPolicy1"
    RuleCombiningAlgId="deny-overrides">
<Target/>
 <Rule RuleId="Rule1" Effect="Permit">
```

[1]http://143.239.71.90:8013/data_location/User_registration.html

```
<Description>My data can be copied/
     stored/transferred in Region 1 and
     Region 2
</Description>
<Target/>
<Condition>
 <Apply FunctionId="and">
 <Apply FunctionId=
   "string-at-least-one-member-of">
 <ActionAttributeDesignator AttributeId=
  "action-id" DataType="string"/>
 <Apply FunctionId="string-bag">
  <AttributeValue DataType="string">
        COPY  </AttributeValue>
  <AttributeValue DataType="string">
        STORE  </AttributeValue>
  <AttributeValue DataType="string">
        TRANSFER  </AttributeValue>
 </Apply> </Apply>
 <Apply FunctionId=
  "string-at-least-one-member-of">
 <EnvironmentAttributeDesignator
     AttributeId="Location"
        DataType="string"/>
 <Apply FunctionId="string-bag">
    <AttributeValue DataType="string">
         Region 1  </AttributeValue>
    <AttributeValue DataType="string">
         Region 2  </AttributeValue>
  </Apply></Apply></Apply>
</Condition>
 </Rule>
<Obligations>
  <Obligation
  ObligationId="Email-to-AB@gmail.com"
  FulfillOn="Permit"/>
</Obligations>
 </Policy>
</PolicySet>
Note: Some prefixes from XACML elements
    are removed for readability.
```

5.3 Visibility for Data Owners

As described in Section 4, the location information for each user's data is referenced in the region information table of the ECSM. As the policies that are evaluated against an access request decide whether the access request should be granted or not, there needs to be a policy to allow the data owner to access the entries in the region information table for his/her data. This can be placed either inside the Cloud administrative policy set or the data owner's policy set. When it is stored in the Cloud Administrator's policy set, it has to be written in a generic way without requiring each individual user's credential to be specified in the policy. An example of such a policy is: If the credential of the requester matches the data owner's credential then allow access to the Region Information of that specific data. The result of such a generic policy is that there will exist only one policy for this purpose in the system. This policy will match the credential of the data owner, which is passed to the authorization system from the Authentication Information table, with the credential presented by the requester while making an access request to the service.

Alternatively, the policy to allow data owners to access the region information of their data can be placed in the data owners' policy sets. The policy would allow the requester presenting the appropriate credential to access region information for his/her data. A drawback with this approach is that the size of policy for each data owner will be increased which may lead to significant growth in the size of the overall policy store. On the other hand, a benefit of this approach is that it would ease the process of distributed enforcement of the data owner's policy in a remote Cloud as the data owner's policy will be passed to the remote Cloud service and no extra mechanism will be needed to identify the data owner.

With the first approach, the user credential and the administrative policy would also need to be passed to the remote service in order to allow the user to access location information from an external Cloud. The remote service would need to make sure that these user credentials are stored correctly and passed to the request contexts and would also need to integrate the received administrative policy with its own administrative policy. We therefore chose the second approach to ease the distributed enforcement of our location control model. Listing 3 shows an example policy which is placed inside the data owner's policy to allow access to the Region Information table. It specifies the UserName and Password that are to be matched to allow the access.

Listing 3: Data owner's policy example to permit access to region information.

```
<Policy PolicyId="UserAccessPolicy2"
    RuleCombiningAlgId="deny-overrides">
<Target/>
<Rule RuleId="Rule2" Effect="Permit">
<Description>Requester presenting the
    specified credential can access the
    region information </Description>
<Target/>
<Condition>
 <Apply FunctionId="and">
  <Apply FunctionId=
   "string-at-least-one-member-of">
   <SubjectAttributeDesignator
    AttributeId="UserName"
    DataType="string"/>
   <Apply FunctionId="string-bag">
    <AttributeValue DataType="string">
       Alice
    </AttributeValue>
  </Apply></Apply>
  <Apply FunctionId=
   "string-at-least-one-member-of">
    <SubjectAttributeDesignator
      AttributeId="Password"
      DataType="string"/>
    <Apply FunctionId="string-bag">
      <AttributeValue DataType="string">
         AlicePassword
      </AttributeValue>
  </Apply></Apply>
  <Apply FunctionId=
   "string-at-least-one-member-of">
    <ResourceAttributeDesignator
     AttributeId="ResourceType"
     DataType="string"/>
    <Apply FunctionId="string-bag">
      <AttributeValue DataType="string">
         RegionInformation
      </AttributeValue>
    </Apply></Apply>
 </Apply>
</Condition>
</Rule>
</Policy>
```

5.4 ECSM Implementation

The enterprise cloud service manager for the prototype was implemented using PHP. It in turn uses an open source authorization system that is implemented in Java as a web service and is available online[2]. This authorization system receives requests as *XACMLAuthzDecisionQuery* elements of the SAML profile of XACML and returns decisions in the form of *XACMLAuthzDecisionStatementType* elements. The system can store policy that comes with the request context and allows the policy of the data owner to be stored. It maintains the data-policy linking table for the provided DataID and PolicyID. It can return obligations if configured appropriately in the policy and can retrieve policy for a relevant DataID which is passed by the ECSM to remote authorization systems via SOAP calls. The remote authorization system receiving a request based on its policy either accepts the request and stores the policy or rejects the request. The authorization system can be configured to use various conflict resolution rules for combining the decisions of administrative policy and data owners' policy. The details of the conflict resolution rules can be found in (Fatema et al., 2011). In this instance, the deny-overrides conflict resolution rule is used. The rationale behind this choice is that it will ensure that if either the administrative policy or the data owner's policy returns a "Deny" the final decision will be a "Deny". The final decision will be a "Grant" only if one of the policies returns a "Grant" but no other policy has returned a "Deny".

We have implemented the Authentication Information table and Region Information table as MySQL tables. In the prototype implementation the Region Service Manager stores the data in different remote directories based on the specified region using the SSH SCP protocol to copy the data. An example of request context showing a copy request by an enterprise role Admin for a data to a certain location is presented in Listing 4 and the corresponding response from the authorization system is presented in Listing 5. For readability, the prefixes of some elements are removed.

[2]http://sec.cs.kent.ac.uk/permis/downloads/Level3/standalone.shtml

Listing 4: Copy request by Admin.

```
<soapenv:Envelope xmlns:soapenv="http://
    schemas.xmlsoap.org/soap/envelope/">
<soapenv:Header/>
<soapenv:Body>
<XACMLAuthzDecisionQuery>
<xacml-context:Request>
<xacml-context:Subject>
<xacml-context:Attribute
  AttributeId="Role"
DataType="string">
<xacml-context:AttributeValue>
      Admin
</xacml-context:AttributeValue>
</xacml-context:Attribute>
</xacml-context:Subject>
<xacml-context:Resource>
<xacml-context:Attribute
  AttributeId="rid"
DataType="string">
<xacml-context:AttributeValue>
      LC001
</xacml-context:AttributeValue>
</xacml-context:Attribute>
</xacml-context:Resource>
<xacml-context:Action>
<xacml-context:Attribute
   AttributeId="action:action-id"
DataType="string">
<xacml-context:AttributeValue>
        COPY
</xacml-context:AttributeValue>
</xacml-context:Attribute>
</xacml-context:Action>
<xacml-context:Environment>
<xacml-context:Attribute
   AttributeId="Location"
DataType="string">
<xacml-context:AttributeValue>
        Region1
</xacml-context:AttributeValue>
</xacml-context:Attribute>
</xacml-context:Environment>
</xacml-context:Request>
</XACMLAuthzDecisionQuery>
</soapenv:Body>
</soapenv:Envelope>
```

Listing 5: Response to copy request.

```
<soapenv:Envelope xmlns:soapenv="http://
    schemas.xmlsoap.org/soap/envelope/">
<soapenv:Body>
 <xacml-context:Response xmlns:xacml-
     context="urn:oasis:names:tc:xacml:2
```

```
      .0:context:schema:os">
    <xacml-context:Result
    ResourceId="ou=some,o=service,c=gb">
       <xacml-context:Decision>
             Permit
       </xacml-context:Decision>
        <xacml-context:Status>
           <xacml-context:StatusCode
           Value="
               urn:oasis:names:tc:xacml:1
               .0:status:ok"/>
        </xacml-context:Status>
        <xacml:Obligations xmlns:xacml=
          "urn:oasis:names:tc:xacml:2.0
              :policy:schema:os">
        <xacml:Obligation ObligationId=
          "Email-to-AB@gmail.com"
           FulfillOn="Permit"/>
        </xacml:Obligations>
      </xacml-context:Result>
     </xacml-context:Response>
    </soapenv:Body>
 </soapenv:Envelope>
```

5.5 Validation

Validation testing of the prototype was performed in order to confirm the correctness of the following functionality:

1. Data owners' location preferences are correctly translated into policy.
2. Data and policy are both stored correctly.
3. When a copy request is sent by an administrator to a region that is accepted by the data owner's policy, a "Grant" response is returned and an obligation to e-mail the data owner is created. On receiving a "Grant" response the data is retrieved from its current region and sent to the approved region and the region information table is updated accordingly.
4. When a copy request is sent by an administrator to a region that is *not* permitted by the data owner's policy, a "Deny" response is returned and the data is not copied in that case.
5. Requests by a data owner for region information with valid credentials are accepted and the data owner is given the region information only for her data.
6. When a transfer request is sent by an administrator to a remote Cloud service in a region that is accepted by the data owner's policy, a "Grant" response, along with the data owner's policy is received from the authorization system.
7. Remote services correctly store the data along with the data owner's policy after data has been transferred.

An additional objective was to determine the performance overhead of the system.

The validation work was performed using an instance of the prototype implementation deployed on a virtual machine running in a vSphere private cloud with ESXi host. The virtual machine was configured with one virtual core Intel Xeon E5620 CPU running at 2.4 GHz with 2GB of memory. Another virtual machine with an identical configuration was used to mimic a remote storage location. All of the functionality described above was found to operate correctly. By averaging over the response times of 100 requests in total it was found that it takes approximately 0.04*s* in order to process a transaction that (a) identifies the correct data location and actions to perform on that data; and (b) queries the authorization system, obtains a response and updates the database accordingly; which indicates that at least 25 requests can be handled per sec by the current implementation of the model. In contrast, a 0.2*s* overhead was observed for connecting to the remote storage via SSH and transferring a file containing an XML representation of a loyalty card of size less than 1KB via SCP. Therefore, for this particular scenario the overhead of using the location control model was found to be negligible compared to the overall cost of performing remote data transfers.

6 CONCLUSION AND FUTURE WORK

The data location control model presented here allows enterprises to manage location of end users' data based on their preferences. It also empowers users with the ability to get up-to-date location information for their data and the assurance that they will be notified when their data changes location. A drawback of this approach is that it relies on the trustworthiness of the provider who must be trusted to integrating requests to the authorization system into their software and procedures and honour the resulting output. Hence, end users cannot verify that location information provided to them is actually true. However, these issues could be addressed by trustmarks and other third-party verification methods. Crytographic techniques could also be used to enhance the trustworthiness of the model. For example, the data could be encrypted using a key that can must be obtained through a request to the authorization system.

For convenience, our prototype stored the data to be protected as files in XML format. However, the model could also be used to protect access to collections of files, such as medical records composed of a variety of files in various formats. Future work will focus on improving the granularity of policies to allow for selective disclosure of data. This granularity could be extended to database entries where the one single row can be identified by DataID and selective disclosure can be provided for various items represented by columns of that row. A private cloud was used as a testbed to perform the validation of the system. Future work will examine the issues around running the authorization system on public clouds. For example, in a federated cloud scenario, how well do the region boundaries of each provider correlate with those of the others?

Future work will also focus on how to integrate location awareness into open cloud platforms such as OpenStack. For OpenStack, obvious integration points include: Horizon, the web-based dashboard that controls all OpenStack components; Keystone, which manages the authentication and authorization of services and users; Swift, which provides object store funtionality; and Cinder, which provides block storage. OpenStack supports the concepts of geographically dispersed regions with separate endpoints (Jackson, 2012), providing a good fit with the data control model described in Section 4. Under this scenario, one Keystone and Horizon is shared between the regions to provide a common access control and dashboard, while distributed Swift and Cinder components allow for complete separation of storage by region. The object storage functionality provided by Swift provides an easier starting point for the integration of location control compared to Cinder as it deals with named, atomic units of data. A first step will to be to allow location preferences to be associated with users and Swift objects, and to provide an API that allows developers to check the permissibility of copying objects from one region to another.

ACKNOWLEDGEMENTS

The research work described in this paper was supported by the Irish Centre for Cloud Computing and Commerce, an Irish national Technology Centre funded by Enterprise Ireland and the Irish Industrial Development Authority.

REFERENCES

Albeshri, A., Boyd, C., and Nieto, J. (2012). Geoproof: Proofs of geographic location for cloud computing environment. In *Distributed Computing Systems Workshops (ICDCSW), 2012 32nd International Conference on*, pages 506–514.

Almutairi, A., Sarfraz, M., Basalamah, S., Aref, W., and Ghafoor, A. (2012). A distributed access control architecture for cloud computing. *Software, IEEE*, 29(2):36–44.

Anderson, A. (2005). OASIS extensible access control markup language (XACML) version 2.0. *OASIS Standard*, 1.

Ashley, P., Hada, S., Karjoth, G., Powers, C., and Schunter, M. (2003). Enterprise privacy authorization language (epal 1.2). *Submission to W3C*.

Basescu, C., Carpen-Amarie, A., Leordeanu, C., Costan, A., and Antoniu, G. (2011). Managing data access on clouds: A generic framework for enforcing security policies. In *Advanced Information Networking and Applications (AINA), 2011 IEEE International Conference on*, pages 459–466.

Benson, K., Dowsley, R., and Shacham, H. (2011). Do you know where your cloud files are? In *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*, pages 73–82. ACM.

Chadwick, D., Zhao, G., Otenko, S., Laborde, R., Su, L., and Nguyen, T. A. (2008). PERMIS: a modular authorization infrastructure. *Concurrency and Computation: Practice and Experience*, 20(11):1341–1357.

Chadwick, D. W. and Fatema, K. (2012). A privacy preserving authorisation system for the cloud. *Journal of Computer and System Sciences*, 78(5):1359–1373.

Chen, D. and Zhao, H. (2012). Data security and privacy protection issues in cloud computing. In *Computer Science and Electronics Engineering (ICCSEE), 2012 International Conference on*, volume 1, pages 647–651. IEEE.

Cranor, L. F. (2003). P3P: Making privacy policies more useful. *Security & Privacy, IEEE*, 1(6):50–55.

Demchenko, Y., Koeroo, O., de Laat, C., and Sagehaug, H. (2008). Extending XACML authorisation model to support policy obligations handling in distributed application. In *Proceedings of the 6th international workshop on Middleware for grid computing*, page 5. ACM.

di Vimercati, S. D. C., Samarati, P., and Jajodia, S. (2005). Policies, models, and languages for access control. In *Databases in Networked Information Systems*, pages 225–237. Springer.

Fatema, K., Chadwick, D. W., and Lievens, S. (2011). A multi-privacy policy enforcement system. In *Privacy and Identity Management for Life*, pages 297–310. Springer.

Godik, S., Anderson, A., Parducci, B., Humenn, P., and Vajjhala, S. (2002). Oasis extensible access control 2 markup language (xacml) 3. Technical report, Tech. rep., OASIS.

Gondree, M. and Peterson, Z. N. (2013). Geolocation of data in the cloud. In *Proceedings of the third ACM conference on Data and application security and privacy*, pages 25–36. ACM.

Iskander, M. K., Wilkinson, D. W., Lee, A. J., and Chrysanthis, P. K. (2011). Enforcing policy and data consistency of cloud transactions. In *Distributed Computing Systems Workshops (ICDCSW), 2011 31st International Conference on*, pages 253–262. IEEE.

ISO (1996). Information technology – open systems interconnection – security frameworks for open systems: Access control framework.

Jackson, K. (2012). *OpenStack Cloud Computing Cookbook*. Packt.

Katz-Bassett, E., John, J. P., Krishnamurthy, A., Wetherall, D., Anderson, T., and Chawathe, Y. (2006). Towards IP geolocation using delay and topology measurements. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 71–84. ACM.

Lynn, T., Healy, P., McClatchey, R., Morrison, J., Pahl, C., and Lee, B. (2013). The case for cloud service trustmarks and assurance-as-a-service. In *Intl. Conference on Cloud Computing and Services Science Closer'13*.

Massonet, P., Naqvi, S., Ponsard, C., Latanicki, J., Rochwerger, B., and Villari, M. (2011). A monitoring and audit logging architecture for data location compliance in federated cloud infrastructures. In *Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on*, pages 1510–1517.

Mohan, A. and Blough, D. M. (2010). An attribute-based authorization policy framework with dynamic conflict resolution. In *Proceedings of the 9th Symposium on Identity and Trust on the Internet*, pages 37–50. ACM.

Noman, A. and Adams, C. (2012). DLAS: Data location assurance service for cloud computing environments. In *Privacy, Security and Trust (PST), 2012 Tenth Annual International Conference on*, pages 225–228. IEEE.

Park, J. and Sandhu, R. (2004). The UCON ABC usage control model. *ACM Transactions on Information and System Security (TISSEC)*, 7(1):128–174.

Peterson, Z. N., Gondree, M., and Beverly, R. (2011). A position paper on data sovereignty: The importance of geolocating data in the cloud. In *Proceedings of the 8th USENIX conference on Networked systems design and implementation*.

Ries, T., Fusenig, V., Vilbois, C., and Engel, T. (2011). Verification of data location in cloud networking. In *Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference on*, pages 439–444. IEEE.

Rissanen, E. (2013). OASIS extensible access control markup(XACML) version 3.0. *OASIS Standard*, 1.

Spillner, J. and Schill, A. (2012). Flexible data distribution policy language and gateway architecture. In *Cloud Computing and Communications (LATINCLOUD), 2012 IEEE Latin America Conference on*, pages 1–6. IEEE.

Subashini, S. and Kavitha, V. (2011). A survey on security issues in service delivery models of cloud computing. *Journal of Network and Computer Applications*, 34(1):1–11.

Turkmen, F. and Crispo, B. (2008). Performance evaluation of XACML PDP implementations. In *Proceedings of the 2008 ACM workshop on Secure web services*, pages 37–44. ACM.