# A Distributed Access Control Architecture for Cloud Computing

**Abdulrahman A. Almutairi and Muhammad I. Sarfraz**, Purdue University

**Saleh Basalamah**, Umm Al-Qura University

**Walid G. Aref and Arif Ghafoor**, Purdue University

*// A novel distributed architecture incorporates principles from security management and software engineering to address cloud computing's security challenges. //*

**THE GROWING POPULARITY** of cloud computing draws attention to its security challenges, which are particularly exacerbated due to resource sharing.[1] Cloud computing's multitenancy and virtualization features pose unique security and access control challenges due to sharing of physical resources among potential untrusted tenants, resulting in an increased risk of side-channel attacks.[2] Additionally, the interference of multitenancy computation can result in unauthorized information flow. Heterogeneity of services in cloud computing environments demands varying degrees of granularity in access control mechanisms. Therefore, an inadequate or unreliable authorization mechanism can significantly increase the risk of unauthorized use of cloud resources and services. In addition to preventing such attacks, a fine-grained authorization mechanism can assist in implementing standard security measures. Such access control challenges and the complexities associated with their management call for a sophisticated security architecture that not only adequately captures access management requirements but also ensures secure interoperation across multiple clouds.

We present a distributed access control architecture for multitenant and virtualized environments. The design of this architecture is based on the principles from security management and software engineering. From a security management perspective, the goal is to meet cloud users' access control requirements. From a software engineering perspective, the goal is to generate detailed specifications of such requirements.

Several researchers have previously addressed access control issues for cloud computing. Daniel Nurmi and his colleagues provided an authorization system to control the execution of virtual machines (VMs) to ensure that only administrators and owners could access them.[3] Stefan Berger and his colleagues promoted an authorization model based on both role-based access control (RBAC) and security labels to control access to shared data, VMs, and network resources.[4] Jose Alcaraz Calero and his colleagues presented a centralized authorization system that provides a federated path-based access control mechanism.[5] What distinguishes our work is that we present an architecture that can be implemented using an XML-based formalism.[6] We also address the problems of side-channel attacks and noninterference in the presence of multitenancy and resource virtualization. Accordingly,

0740-7459/12/$31.00 © 2012 IEEE

we present an access control architecture that addresses these challenges.

## Authorization Requirements

In order to build a secure and trusted distributed cloud computing infrastructure, the cloud architecture's designer must address several authorization requirements.

### Multitenancy and Virtualization

Side-channel attacks and interference among different policy domains pose daunting challenges in distributed clouds. Side-channel attacks are based on information obtained from physical implementation (for example, via time- or bandwidth-monitoring attacks). Side-channel attacks arise due to lack of authorization mechanisms for sharing physical resources. The interference among tenants exists primarily because of covert channels with flawed access control policies that allow unauthorized information flow.[7]

### Decentralized Administration

Decentralized administration is characterized by the principle of local autonomy, which implies that each service model retains administrative control over its resources. This is in contrast to a centralized administration approach, which implies loss of autonomy in controlling resources; it's not a desirable system feature when dealing with several independent clouds. Moreover, the need for a fine-grained access control can enact substantial requirements in designing an access control policy employing a large number of authorization rules. These rules can grow significantly with an increase in the granularity of resources, as well as with the number of users and services supported by the cloud. A centralized design based on the integration of all global rules can pose significant challenges.

### Secure Distributed Collaboration

To support a decentralized environment, the cloud infrastructure should allow both horizontal and vertical policy interoperation for service delivery. Due to the heterogeneous nature of the cloud, resource and service policies might use different models requiring seamless interoperation among policies. These policies must be correctly specified, verified, and enforced. A service-level agreement (SLA) can provide secure collaboration and assure that services are provided according to preestablished rules.

> Side-channel attacks and interference among different policy domains pose daunting challenges in distributed clouds.

### Credential Federation

Because a user might invoke services across multiple clouds, access control policies must support a mechanism to transfer a customer's credentials across layers to access services and resources. This requirement includes a provision for a decentralized single-sign-on mechanism within the authorization model, which can enable persistent authorization for customers in terms of their identity and entitlement across multiple clouds.[6]

### Constraint Specification

The collaborative nature of a cloud computing environment requires the specification of semantic and contextual constraints to ensure adequate protection of services and resources, especially for mobile services. Semantic constraints (for example, separation of duties) and contextual constraints (such as temporal or environmental constraints included in an access request) must be evaluated when determining access to services and resources.[8] Semantic and contextual constraints are specified in the access control policy.

## Designing a Distributed Cloud Architecture

The nature of assuring resource sharing across multiple clouds depends on the collaborative environment. Figure 1 shows three types of collaborations (federated, loosely coupled, and ad hoc) that can fulfill the aforementioned authorization requirements.

### Federated Collaboration

Federated collaboration is characterized by a high degree of mutual dependence and trust among collaborating clouds and supports a long-term interoperation. To be secure, this collaboration requires a global metapolicy that's consistent with local policies of the collaborating clouds. A policy-composition framework (top block of Figure 1) is necessary if a global metapolicy needs to be generated by integrating the policies of individual clouds.[8]

### Loosely Coupled Collaboration

In a loosely coupled collaborative environment, local policies govern interactions among multiple clouds. In contrast to a federated collaboration,

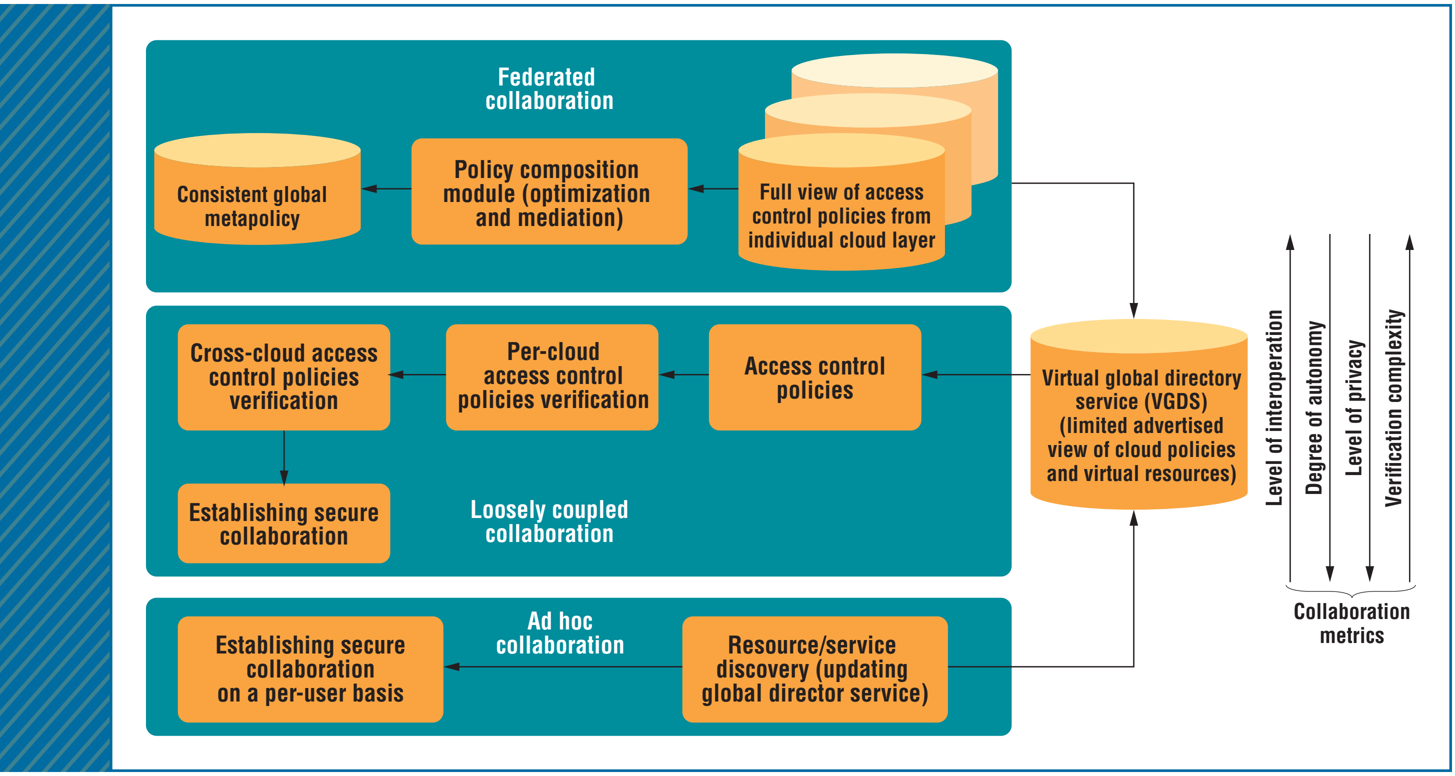

Federated collaboration
Consistent global metapolicy
Policy composition module (optimization and mediation)
Full view of access control policies from individual cloud layer
Cross-cloud access control policies verification
Per-cloud access control policies verification
Access control policies
Establishing secure collaboration
Loosely coupled collaboration
Virtual global directory service (VGDS) (limited advertised view of cloud policies and virtual resources)
Level of interoperation
Degree of autonomy
Level of privacy
Verification complexity
Collaboration metrics
Ad hoc collaboration
Establishing secure collaboration on a per-user basis
Resource/service discovery (updating global director service)

**FIGURE 1.** Characterization of collaboration in a mutlicloud environment. In a distributed environment, we can build a security architecture based on the design of these collaborations. Their comparison is based on degree of interoperation, autonomy, privacy, and verification complexity. The architecture we present in this article is based on federated and loosely coupled collaborations.

this collaboration is more flexible and autonomous in terms of access policies and resource management. Two collaborating clouds can virtualize their resources and allow autonomous sharing of resources. The information about the virtualized shareable resources and services of each cloud is stored in a virtual global directory service (VGDS), which is manifested across service-level agreement (SLAs). The middle block of Figure 1 shows the verification for conformance of individual clouds' security and privacy policies for loosely coupled collaboration.

### Ad Hoc Collaboration

In ad hoc collaboration, a user is only aware of a few remote sharable services. Because a priori information about an application's overall service requirements might not be available to the user or cloud at the start of a session, a cloud might deny access to its resources. To ensure secure interoperation via discovered resources and services in a dynamic interoperation environment where clouds can join and leave in an ad hoc manner, appropriate authentication and authorization mechanisms need to be developed.

### Evaluation

Several metrics can be used to evaluate these collaborations, including

- *degree of interoperation*, which indicates the level of service and resource sharing among multiple clouds;
- *autonomy*, which refers to a cloud's ability to perform its local operations without any interference from cross-cloud accesses;
- *degree of privacy*, which specifies the extent of information a cloud provider discloses about its internal policies and local constraints; and
- *verification complexity*, which quantifies the complexity associated with verifying the correctness of the overall constraints while integrating multiple policies.

Figure 1 shows the tradeoffs among collaboration types and these metrics; the collaboration metrics' arrows point toward higher values. For example, ad hoc collaboration supports a higher level of privacy than federated or loosely coupled collaborations do.

## A Distributed Cloud Security Architecture

The proposed distributed architecture that addresses and incorporates the aforementioned authorization requirements can be built using three types of components: a virtual resource manager (VRM), a distributed access control module (ACM; Figure 2), and an SLA (Figure 3). The proposed architecture (Figure 4) uses the RBAC model, which is recognized for its support for simplified administration and scalability.[6] However, the design of this architecture is generic enough to support other access control policies, such as discretionary access control and multilevel security.

### VRM Design Specification

The heterogeneity and granularity of virtual resources in a cloud environment call for a VRM at each layer of the cloud, as depicted in Figure 4. The VRM is responsible for providing and deploying virtual resources. Consequently, it maintains a list of required virtual resources with their configuration, including both local and remote resources through VGDS—the one shown in Figure 1. SLAs provide access to remote resources, whereas the VRM is responsible for monitoring deployed resources and might allocate or release them to ensure SLA compliance, including guarantees for quality of service. To manage the scalability issue in cloud computing in term of users and resources, the VRM uses a distributed architecture.[3]

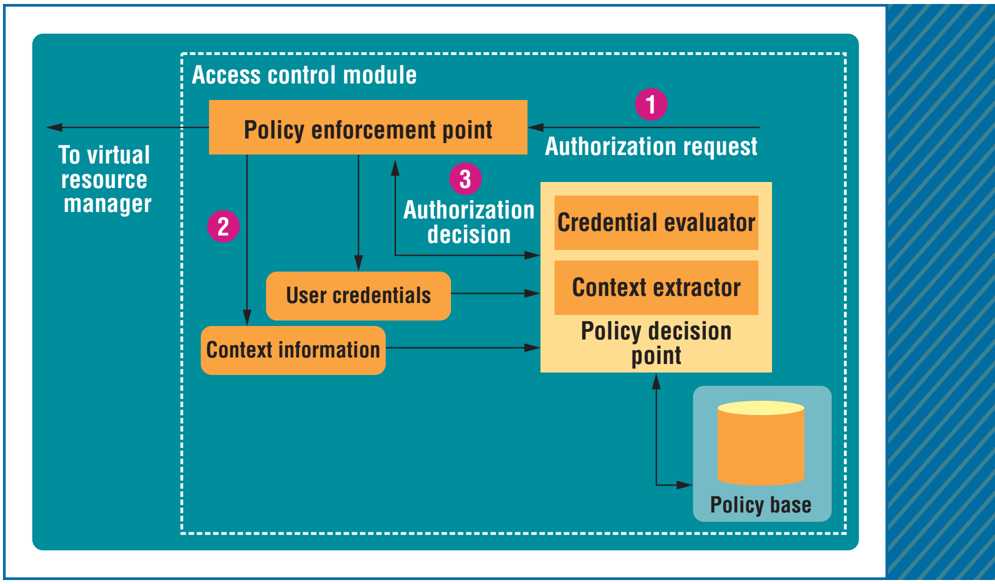


**FIGURE 2.** Access control module architecture. This component can be used to build the proposed distributed architecture.

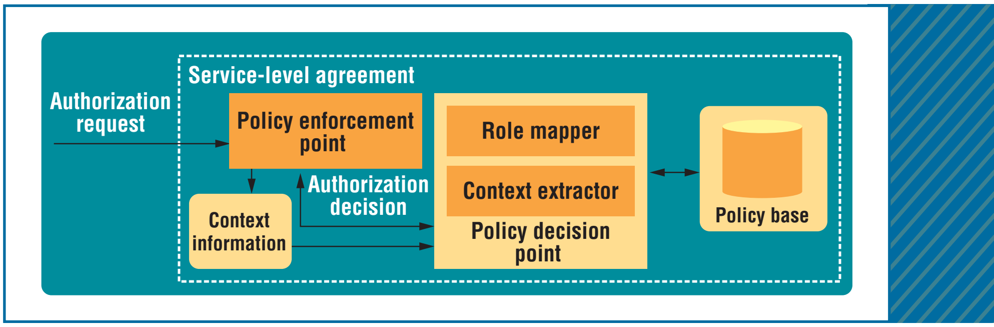


**FIGURE 3.** Service-level agreement (SLA) architecture. This component can be used to build the proposed distributed architecture.

### ACM Design Specification

An ACM resides at each layer to enforce the access control policy at its resident layer. As shown in Figure 2, the main components of an ACM include

- a policy decision point,
- a policy enforcement point (PEP), and
- a policy base.

The authorization request (Figure 2, step 1) submitted to the PEP includes the requesting subject, the requested service or resource, and the type of permissions requested for that service or resource (such as read or write privileges). The request might also include the credentials needed for authentication and authorization. The PEP extracts the authentication credentials and the context information from the authorization request and forwards them to the credential evaluator and context evaluator (Figure 2, step 2). The PEP receives the decision about granting the request (Figure 2, step 3) and either grants or denies the user's authorization request.

If the request contains an authenticating credential, the credential evaluator assigns a user a local role based on the user-to-role assignment rules stored in the RBAC policy base. The process of user-to-role assignment requires input from the context evaluator regarding contextual constraints. If the request contains an authorization credential, the credential evaluator assesses if the role corresponds to a local role. If not, the implication is that this is a single-sign-on request and requires role mapping by a relevant SLA. Subsequently, the user acquires the privileges

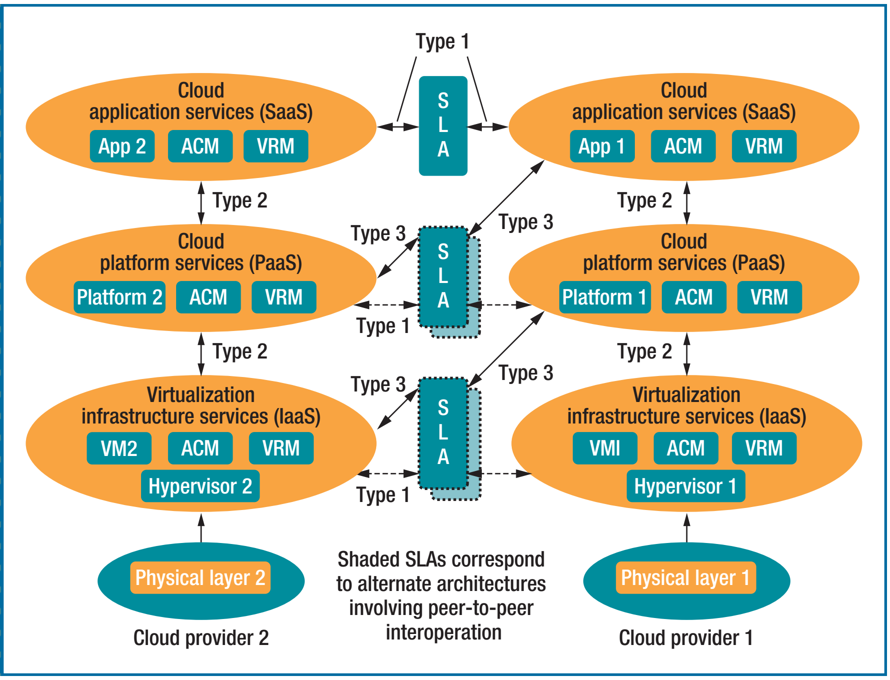

Type 1
S L A
Cloud application services (SaaS)
App 2
ACM
VRM
Cloud application services (SaaS)
App 1
ACM
VRM
Type 2
Type 3
Type 3
Type 2
Cloud platform services (PaaS)
Platform 2
ACM
VRM
S L A
Cloud platform services (PaaS)
Platform 1
ACM
VRM
Type 1
Type 2
Type 3
Type 3
Type 2
Virtualization infrastructure services (IaaS)
VM2
ACM
VRM
Hypervisor 2
S L A
Virtualization infrastructure services (IaaS)
VMI
ACM
VRM
Hypervisor 1
Type 1
Physical layer 2
Cloud provider 2
Shaded SLAs correspond to alternate architectures involving peer-to-peer interoperation
Physical layer 1
Cloud provider 1

**FIGURE 4.** Intercloud and intracloud interoperations for the distributed security architecture. Shaded SLAs correspond to alternate architectures involving peer-to-peer interoperation.

of the locally assigned role or of a mapped role in a remote cloud.[6]

## SLA Specification

To allow interoperation among autonomous policies manifested through ACMs, an SLA implements a mediated policy. For this purpose, an SLA performs role mapping, specifies isolation constraints for resource sharing to prevent side-channel attacks, and presents a virtualized view of resources at the levels for which the SLA is negotiated. In addition, an SLA usually includes quality-of-service parameters, as well as billing and auditing functions. Figure 3 depicts the authorization flow within an SLA.

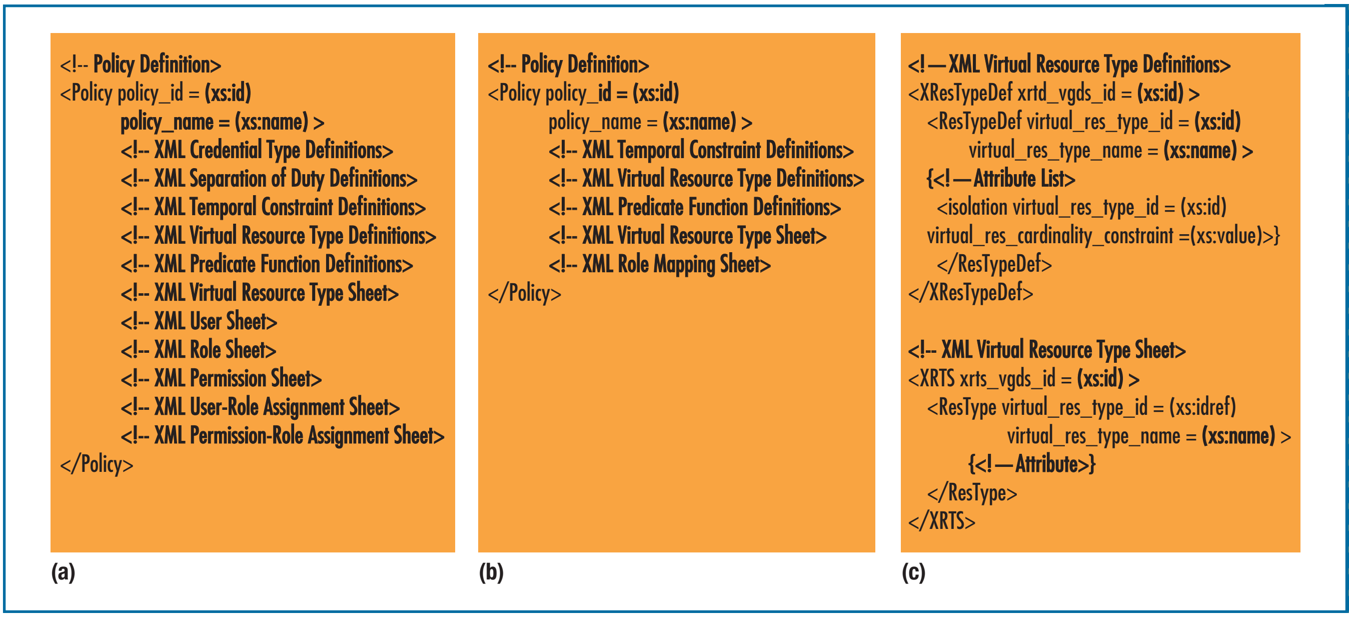
Role mapping is a function that maps a local role to a role in a remote cloud and grants access to all the mapped role's permissions. The mutually agreed upon mediated policy, which is generally a subset of the policies of the participating ACMs, enforces access control for distributed services or resources through this mapping. In addition, the SLA physically isolates resources to prevent side-channel attacks at the remote cloud.[2] Such isolation can prevent multiple VMs from residing on the same physical machine. Physical isolation can be explicitly enforced in the form of cardinality constraint rules in the RBAC policy.[6] By setting the cardinality constraint parameter to one, we can enforce such isolation.

## RBAC Policy Specification for Proposed Architecture

We adopted an XML-based specification due to its compatibility with the emerging standards for cloud systems and security protocols, with the ultimate goal being that the proposed architecture should be interoperable with complementary security protocols for cloud systems. Figures 5a and 5b show the XML-based specifications of ACMs and SLAs, respectively. (The full details of RBAC XML declaration appear elsewhere.[6])

The ACM's XML user sheet defines the authenticating credentials and the XML role sheet defines the authorization credentials. The XML user-to-role assignment sheet defines user-to-role assignment rules, which can be based on attributes associated with users' credentials as defined in the XML user sheet. XML permission-to-role assignment sheets define permission-to-role assignment rules. Permission-to-role constraints can be based on attributes associated with a role's credential or the resource type as defined in XML virtual resource sheets (see Figure 5c). The constraints can be semantic (for instance, separation of duty) or temporal. To represent authorization requirements as a set of predicates, predicate function definitions sheets define the formal notion of predicate expression. A predicate function definition sheet can include mediated rules for intercloud resource sharing; a predicate expression can help evaluate sets of temporal or nontemporal constraints.[6]

A permission defined in the XML permission sheet comprises a specified operation on a given resource type. Thus, a role assigned a permission defined on a given resource type receives access to all instances of that resource type. XML allows access granularity at individual levels within a resource type to provide support for individual virtual resources—for example, as mentioned earlier, we can specify the physical isolation attribute of a virtual resource at the individual resource level in the form of a cardinality constraint to prevent side-channel attacks in the local cloud. Note that depending on if the requested resources are local or remote, the ACM decides whether or not to invoke SLA. The XML specification of the SLA depicted in Figure 5b provides a limited view of advertised virtual resources, role mapping, and cardinality constraints.



```
<!-- Policy Definition>
<Policy policy_id = (xs:id)
        policy_name = (xs:name) >
        <!-- XML Credential Type Definitions>
        <!-- XML Separation of Duty Definitions>
        <!-- XML Temporal Constraint Definitions>
        <!-- XML Virtual Resource Type Definitions>
        <!-- XML Predicate Function Definitions>
        <!-- XML Virtual Resource Type Sheet>
        <!-- XML User Sheet>
        <!-- XML Role Sheet>
        <!-- XML Permission Sheet>
        <!-- XML User-Role Assignment Sheet>
        <!-- XML Permission-Role Assignment Sheet>
</Policy>
```

(a)

```
<!-- Policy Definition>
<Policy policy_id = (xs:id)
        policy_name = (xs:name) >
        <!-- XML Temporal Constraint Definitions>
        <!-- XML Virtual Resource Type Definitions>
        <!-- XML Predicate Function Definitions>
        <!-- XML Virtual Resource Type Sheet>
        <!-- XML Role Mapping Sheet>
</Policy>
```

(b)

```
<!—XML Virtual Resource Type Definitions>
<XResTypeDef xrtd_vgds_id = (xs:id) >
  <ResTypeDef virtual_res_type_id = (xs:id)
        virtual_res_type_name = (xs:name) >
  {<!—Attribute List>
    <isolation virtual_res_type_id = (xs:id)
  virtual_res_cardinality_constraint =(xs:value)>}
    </ResTypeDef>
</XResTypeDef>

<!-- XML Virtual Resource Type Sheet>
<XRTS xrts_vgds_id = (xs:id) >
  <ResType virtual_res_type_id = (xs:idref)
            virtual_res_type_name = (xs:name) >
        {<!—Attribute>}
  </ResType>
</XRTS>
```

(c)

**FIGURE 5.** High-level XML declaration: (a) access control module, (b) mediated service-level agreement policy, and (c) virtual resource definition and sharing constraint (local and remote).

### Ensuring Noninterference

To avoid security risk due to potential interference as a result of multitenancy, we must abstract policies by participating ACMs and SLAs as an information flow model. Subsequently, this model can be verified to ensure the property of noninterference.[7] Such verification ensures that each domain remains unaffected by the actions of other domains. As side-channel attacks can be managed through cardinality constraints, unauthorized information flow can only occur when there's conflict among cloud policies. In conjunction with the data model, verification models[8] or verification tools (such as Alloy[9]) can detect conflicts among policies, which causes unauthorized information flow.

## Distributed Authorization Process and Use Cases

Three types of interoperations related to authorization flow can occur at various layers of the distributed architecture, as illustrated in Figure 4. Type 1 depicts a horizontal (peer-to-peer) interoperation between the same levels of different cloud providers; type 2 represents a vertical interoperation between layers within the same cloud; and type 3 indicates a cross-layered interoperation between different clouds at different layers. Both type 1 and 3 interoperations require SLAs among the participating clouds. These three types of interoperation also establish distributed authorization mechanisms among ACMs.

For distributed authorization, VRMs use their peer-to-peer or cross-layered interoperations through VGDSs in order to provide the required resources. VGDSs have both the local virtual resource IDs and the paths of the physical resources they map to, as well as remote virtual resource IDs consistent with the SLAs that advertise these resources. Therefore, a VGDS can manifest either through peer-to-peer or cross-layered SLAs (shown in dotted SLA blocks at the PaaS and IaaS levels of Figure 4). Assessment of these architectural choices is an open problem.

For interoperations among ACMs, we envision loosely coupled collaboration consistent with type 1 and type 3 interoperations because individual clouds need to reveal only limited information about their services and policies. Federated cloud collaboration requires an extensive analysis prior to generating the global metapolicy, which can result in a high degree of complexity and rule explosion. Therefore, this approach isn't scalable for distributed collaboration. Also, generating a consistent global metapolicy could require extensive mediation to resolve conflicts among heterogeneous policies.[8] Similarly, ad hoc collaboration doesn't federate credentials across clouds because it lacks SLA support.

For type 2 interoperation, federated collaboration can be an appropriate approach because it requires only vertical integration of polices. Therefore, the high complexity for generating a global metapolicy within a cloud is justified because the cloud provider has access to all its local policies belonging to the three service models. However, the provider must address the challenge of conflict resolution and mediation for generating such a metapolicy. Figure 5a

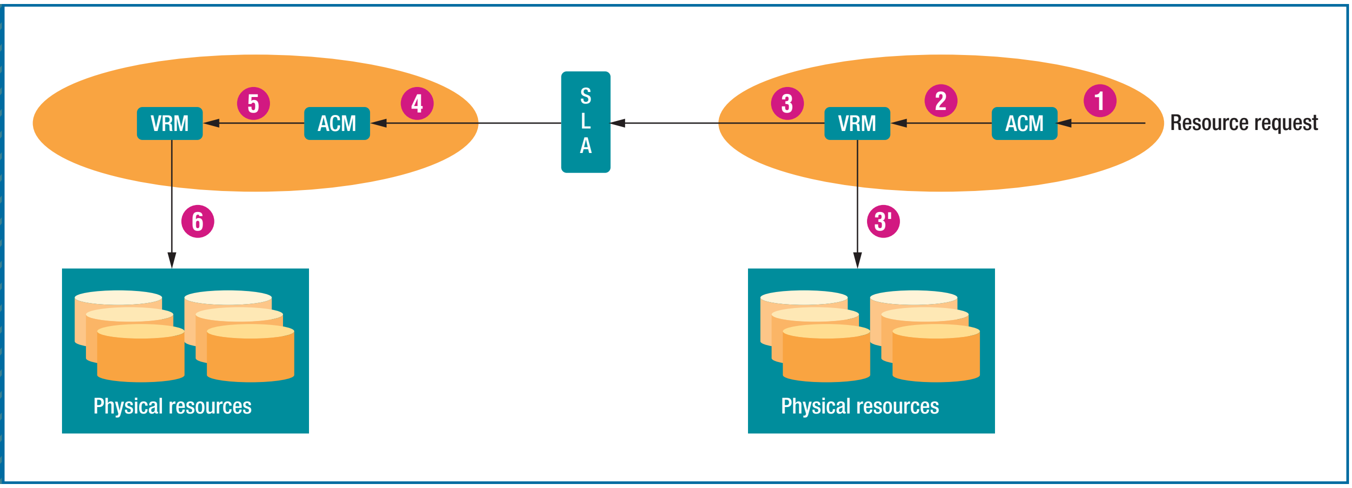


**FIGURE 6.** Flow of request via the access control module and virtual resource manager across multiple clouds.

shows an example of a high-level metapolicy specification; further details appear elsewhere.[6]

### Authorization Process

When a customer requests a service or virtual resource, the request goes to the local ACM (Figure 6, step 1). If the ACM grants this request, it routes the request to the local VRM (step 2). If the requested resources reside in the local cloud, the VRM (after consulting the VGDS) forwards the request to the local ACM of the lower level—for example, from SaaS to PaaS (step 3). Ultimately, the request goes to the infrastructure as a service (IaaS)-level VRM in order to deploy the required physical resources. If the required resources are in a remote cloud, the local VRM, after consulting the VGDS, issues a remote request to the appropriate SLA (step 3). The SLA, after performing its functions involving role mapping and evaluating the policy constraints, forwards the request to the remote ACM (step 4). After verifying its own constraints (including cardinality constraints), the ACM informs its local VRM to allocate the desired resources (step 5). Finally, the VRM identifies and configures the local physical resources (step 6).

### Use Cases

This authorization process is a generic representation of a set of use cases. To specify these cases, we adopt Alcaraz Calero and colleagues' authorization model[4] by extending it to support multitenancy and virtualization in a distributed environment. Figure 7 illustrates two classes of scenarios covering all possible interactions within and across multiple clouds. These scenarios involve the three types of interoperations discussed earlier in this article. Assuming an RBAC model, the authorization request can be represented using a four-tuple expression (subject, permission, interface, object [attributes]), which can be interpreted in the following way: the subject (as a role) asking for a permission to be performed over the object (virtual resource or service) with its attributes (such as isolation constraint) and that object's interface type. We assume the authorization request is time stamped to accommodate temporal contextual constraint. From an RBAC perspective, the subject is represented as a role. In addition, users of the XML user sheet specified in Figure 5a, which identifies user-to-role assignments, can assume their respective roles. Along with this assignment, the proposed four-tuple can fully specify an authorization request.

When user X initiates the authorization process to access an application (app) at the SaaS level of its local cloud ($\mathsf{SaaS}_{\mathsf{CP1}}$), the corresponding ACM's PEP needs to authenticate the user prior to assigning a local role (for example, **Rx**) based on its credentials. If X requires a remote resource, the participating SLA assigns it a mapped role (say, **Ry**).

The local SaaS verifies this request, represented as $\mathsf{R}_{\mathsf{X}}$, **execute**, $\mathsf{SaaS}_{\mathsf{CP1}}$, **app**, for authorization. Consequently, one of the following scenarios can occur.

**Scenario A.** Figure 7a depicts this scenario. We assume the requested resources are locally available, resulting in type 2 collaboration within the local cloud. Accordingly, the SaaS's local VRM identifies virtual resources—for example, computation instance ($\mathsf{CompInst}_{\mathsf{x}}$) and storage ($\mathsf{Store}_{\mathsf{x}}$). Assuming that the local policy verifies the authorization request, the VRM, after consulting with the VGDS, requests the two desired resources through the following two authorization requests: $\mathsf{R}_{\mathsf{X}}$, **execute**, $\mathsf{PaaS}_{\mathsf{CP1}}$, $\mathsf{CompInst}_{\mathsf{X}}\mathsf{(isolation=1)}$ and $\mathsf{R}_{\mathsf{X}}$, **execute**, $\mathsf{IaaS}_{\mathsf{CP1}}$, $\mathsf{Store}_{\mathsf{X}}$. Here, we assume X is requesting fully isolated computation resources to avoid side-channel attacks.

**Scenario B.** Figure 7b shows four scenarios depicting ACM interaction across multiple clouds at different levels:

1. The service requested (app) by X consists of two services, app1 and app2 (local and remote, respectively), causing interoperation between SaaS ACMs in different clouds. In this case, we assume a peer-to-peer interoperation (type 1). Consequently, the VRM in the local SaaS of CP1 forwards the request $R_y$, execute, $\text{SaaS}_{\text{CP2}}$, app2 to the remote SaaS's ACM of CP2 through the relevant SLA (depicted in Figure 6). Because app1 and app2 use virtualized resources in their local clouds, the remaining authorization process within each cloud is similar to scenario A.
2. In scenario B.2, the local SaaS needs to access virtual resources managed by CP2's PaaS and IaaS. Assuming cross-layered SLA architecture, the local SaaS's VRM generates the authorization request $R_y$, execute, $\text{PaaS}_{\text{CP2}}$, $\text{CompInst}_x$(isolation = 1), which is then forwarded to CP2's PaaS's ACM through the SLA. The remaining authorization process for acquiring virtualized resources within the remote cloud is similar to scenario A.
3. Scenario B.3 is identical to scenario B.2, except the local cloud needs virtual resources, which are maintained by a remote IaaS. Accordingly, the local PaaS's VRM generates the authorization request $R_y$, execute, $\text{IaaS}_{\text{CP2}}$, $\text{VM}_x$(isolation = 1) and forwards it to the remote IaaS's ACM through a cross-layered SLA.
4. In scenario B.4, an intermediate cloud must process the authorization request with further rerouting to a remote cloud (CP3) where the physical infrastructure is located.[1] In this case, SaaS, PaaS, and IaaS belong to distinct clouds. The authorization requests Ry, execute, PaaSCP2,CompInstX(isolation = 1) and Rz, execute, IaaSCP3,VMX( isolation = 1) are generated in succession to the corresponding ACMs after the VRMs invoke the SLAs.

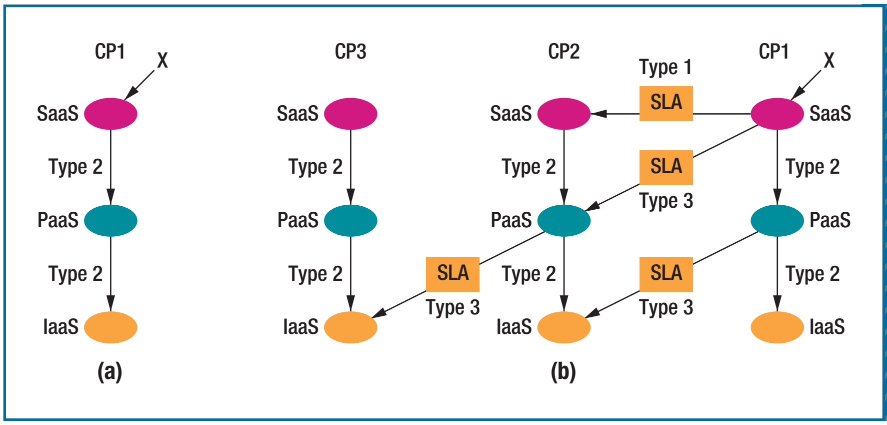


**FIGURE 7.** Scenario-based policy interoperation. (a) Secure interoperation within a local cloud to acquire resources that are locally available. (b) Secure interoperation involving SLAs at different levels to acquire resources among multiple clouds.

These use cases represent high-level design requirements for the proposed architecture and cover all possible authorization flow processes that can be used to design and develop the distributed architecture. Currently, development for a prototype of this architecture is underway; it uses the Microsoft Azure platform to develop a health surveillance and rapid response infrastructure with the capability of collecting and analyzing real-time epidemic data from various hospitals. This cloud computing environment consists of compute clusters, reliable data storage, and software services. The stakeholders include researchers, physicians, and government public health management personnel in the chain of reporting. The services provided to stakeholders include visual analytics, statistical data analysis, and scenario simulations.[10]

The architecture we present in this article represents a precise but comprehensive authorization design for access management. Using an XML-based declaration of the access control policy for this architecture is a step toward its implementation. However, we must address several open challenges in order to implement a fully secure and trusted cloud environment. These include design of an authentication mechanism, cryptography and key management, mediation for conflict resolution of heterogeneous policies, software design for virtualized resources, integrating information flow verification tools to ensure noninterference, and architectural choices for SLAs. We plan to address these challenges in our future work.

## Acknowledgments

The research in this article is partially funded by the US National Science Foundation under grant IIS-0964639.

## References

1. H. Takabi, J.B.D. Joshi, and G.-J. Ahn, "Security and Privacy Challenges in Cloud Computing Environments," *IEEE Security & Privacy*, vol. 8, no. 6, 2010, pp. 24–31.
2. T. Ristenpart et al., "Hey, You, Get off of My Cloud: Exploring Information Leakage in Third-Party Compute Clouds," *Proc. 16th ACM Conf. Computer and Communications Security* (CCS 09), ACM, 2009, pp. 199–212.

3. D. Nurmi et al., “The Eucalyptus Open-Source Cloud-Computing System,” *Proc. 9th IEEE/ACM Int’l Symp. Cluster Computing and the Grid* (CCGRID 09), IEEE CS, 2009, pp. 124–131.
4. S. Berger et al., “Security for the Cloud Infrastructure: Trusted Virtual Data Center Implementation,” *IBM J. Research and Development,* vol. 53, no. 4, 2009, pp. 560–571.
5. J.M. Alcaraz Calero et al., “Toward a Multitenancy Authorization System for Cloud Services,” *IEEE Security & Privacy,* vol. 8, no. 6, 2010, pp. 48–55.
6. R. Bhatti, E. Bertino, and A. Ghafoor, “X-Federate: A Policy Engineering Framework for Federated Access Management,” *IEEE Trans. Software Eng.*, vol. 32, no. 5, 2006, pp. 330–346.
7. J. Rushby, *Noninterference, Transitivity, and Channel-Control Security Policies*, tech. report CSL-92-02, Computer Science Lab, SRI Int’l, 1992.
8. B. Shafiq et al., “Secure Interoperation in a Multidomain Environment Employing RBAC Policies,” *IEEE Trans. Knowledge and Data Eng.,* vol. 17, no. 11, 2005, pp. 1557–1577.
9. D. Jackson, I. Schechter, and I. Shlyakhter, “ALCOA: The Alloy Constraint Analyzer,” *Proc. 22nd Int’l Conf. Software Eng.*, ACM, 2000, pp. 730–733.
10. S. Afzal, R. Maciejewski, and D.S. Ebert, “Visual Analytics Decision Support Environment for Epidemic Modeling and Response Evaluation,” *IEEE Conf. Visual Analytics Science and Technology* (VAST 11), IEEE CS, 2011, pp. 191–200.

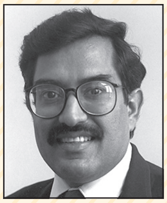
## ABOUT THE AUTHORS

**ABDULRAHMAN A. ALMUTAIRI** is a PhD student in the School of Electrical and Computer Engineering at Purdue University. His research interests include information security and privacy and cloud computing systems. Almutairi has an MS in electrical and computer engineering from Purdue University. He is a student member of IEEE. Contact him at aalmutai@purdue.edu.

**MUHAMMAD I. SARFRAZ** is a student at Purdue University. His research interests include distributed access control and information security and privacy. Sarfraz has a BSC in computer science from the King Fahd University of Petroleum and Minerals. He is a student member of IEEE. Contact him at msarfraz@purdue.edu.

**SALEH BASALAMAH** is an assistant professor at Umm Al-Qura University. His research interests include computer vision, multimedia, and information security. Saleh has a PhD in bioengineering from Imperial College London. He’s a member of IEEE and ACM. Contact him at smbasalamah@uqu.edu.sa.

**WALID G. AREF** is a professor of computer science at Purdue University. His research interests include extending the functionality of database systems in support of emerging applications, query processing, indexing, data mining, and geographic information systems. Aref has a PhD in computer science from the University of Maryland at College Park. He is a member of ACM, a senior member of IEEE, and current chair of the ACM Special Interest Group on Spatial Information. Contact him at aref@cs.purdue.edu.



**ARIF GHAFOOR** is a professor at the School of Electrical and Computer Engineering at Purdue University. His research interests include information security and distributed multimedia systems. Ghafoor has a PhD in electrical engineering from Columbia University. He is an IEEE Fellow. Contact him at ghafoor@purdue.edu.